Université des Sciences et des Technologies de Lille
Ecole Doctorale Sciences Pour L'ingénieur

Thèse de Doctorat

Spécialité **Informatique**

présentée par
Edouard Leurent

Safe and Efficient Reinforcement Learning for Behavioural Planning in Autonomous Driving

---

Apprentissage par renforcement sûr et efficace pour la prise de décision comportementale en conduite autonome

---

sous la direction d'**Odalric-Ambrym Maillard**
et de **Wilfrid Perruquetti**,
ainsi que l'encadrement de **Denis Efimov**
et de **Yann Blanco**.

Soutenue publiquement à **Villeneuve d'Ascq**, le **30 octobre 2020** devant le jury composé de

M. Lucian **Buşoniu**	Universitatea Tehnică din Cluj-Napoca	Rapporteur
M. Jorge **Villagra**	Universidad Politécnica de Madrid	Rapporteur
M^{me} Luce **Brotcorne**	Inria	Présidente
M. Marc **Deisenroth**	University College London	Examinateur
M. Denis **Efimov**	Inria	Encadrant
M. Odalric-Ambrym **Maillard**	Inria	Directeur de thèse

Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL),
UMR 9189 Équipe SequeL, 59650, Villeneuve d'Ascq, France

À mes grands-pères, Marc et Germain,
qui ont nourri mon goût pour les sciences.

Remerciements

Un concept central en apprentissage par renforcement est celui du *"credit assignment"* (« attribution du mérite »). Selon ce principe, lors de l'obtention d'une haute récompense, il convient de remonter l'historique des évènements survenus par le passé afin d'identifier ceux qui furent responsables de ce succès. Prêtons-nous à l'exercice.

Tout d'abord, je dois beaucoup à mes parents, ainsi qu'à mon frère et mes sœurs. Leur affection et encouragements constants au fil des années m'ont permis de m'engager avec confiance dans cette aventure.

Mais je n'aurais pas entrepris cette thèse sans l'exemple éclatant des doctorants de Parrot, Gauthier Rousseau et Clément Pinard, ni le concours de Jill-Jênn Vie, Edouard Oyallon, Alberto Bietti et Michal Valko qui ont tous conspiré à me mener au laboratoire SequeL. Je remercie également l'examen du permis de conduire, auquel mes échecs répétés m'ont permis de développer un intérêt égoïste mais pragmatique pour ce sujet de thèse en particulier.

J'adresse maintenant mes plus hautes *traces d'éligibilité* ainsi que mes plus chaleureux remerciements à mes encadrants. Odalric et Denis, j'admire profondément votre intégrité et votre rigueur scientifique, ainsi que l'étendue de vos connaissances, que vous savez mobiliser pour répondre à la moindre de mes interrogations avec une facilité déconcertante. Merci également pour votre ouverture d'esprit, dont témoigne particulièrement la rencontre de vos disciplines respectives et la confrontation fructueuse des points de vue et des méthodes qui en découle. Mais avant tout, je vous suis reconnaissant pour votre bienveillance, votre disponibilité et votre soutien appuyé lorsque j'en avais besoin. Yann, je te remercie pour avoir monté ce projet ambitieux, et pour m'avoir accordé une pleine liberté dans mes recherches, bien qu'elles se soient parfois écartées des préoccupations très concrètes de l'ingénierie Renault. Enfin, Wilfrid je te remercie pour tes précieux conseils toujours pertinents.

I am also very grateful to Lucian Busoniu and Jorge Villagra for their thoughtful reporting on this manuscript, as well as to all members of the jury, Luce Brotcorne, Marc Deisenroth and Rosane Ushirobira, for their valuable feedback, and for giving their time and energy to make it possible for my defence to take place amid the turmoil of a global pandemic despite being under lockdown.

J'en viens à mes coreligionnaires de la pause-café, avec qui j'ai pu décompresser, partager mes intérêts, mes joies et mes peines. A SequeL tout d'abord, je remercie particulièrement Mathieu et Xuedong, camarades de la première heure avec qui j'ai entamé d'inoubliables marches aléatoires dans les montagnes de Stellenbosch ; Nicolas et Omar, avec qui j'ai eu le privilège et le plaisir de collaborer ; Guillaume et Lilian dont l'éloignement rendait la visite occasionnelle d'autant plus festive ; Nathan et Dorian, dont l'appétence pour le débat n'a d'égale que leur propension à le trancher à coups d'étude ad-hoc ; Pierre Ménard dont la cinéphilie et l'hospitalité ont permis la renaissance en grande pompe du Cinéquel ; Pierre Schegg, compagnon d'armes avec qui nous défendons fièrement le bastion des roboticiens à SequeL, ainsi que Florian, Ronan, Merwan, Mahsa, Julien, Yannis, Reda, Romain, Jean, Sarah et Antoine. Sans oublier les chercheurs : merci Emilie, Jill-Jênn et Philippe pour vos contributions à la vie du laboratoire et à sa convivialité, ainsi que pour les collaborations et conversations scientifiques. À Renault maintenant, Jean, j'ai su dès notre rencontre à Munich que ton goût communicatif et intarissable pour la philosophie nous conduirait à de précieuses et mémorables conversations. Merci également à Lu, Clara, Edwin, Federico, Louis et Thomas pour nos fameux repas-mini-doc™. Je salue aussi les doctorants du CAOR, Philip, Marin et Florent, avec qui il est toujours agréable de discuter, aux Mines ou en conférence, de nos approches différentes à un sujet commun.

Mais il y a une vie en dehors du travail, et je dois ses meilleurs moments à mes amis, merci Luc, Pierre, Bertrand, Adrien, Benjamin, Mano, Quentin, Thomas D., Thomas L., Simon, Oriane, Ivain, Magali et Daniel.

Enfin, merci Ariane pour ton soutien indéfectible, pour toutes tes qualités que j'admire tant, et pour me donner envie d'être et de donner le meilleur de moi-même.

Résumé

Dans cette thèse de doctorat, nous étudions comment des véhicules autonomes peuvent apprendre à garantir la *sûreté* et à éviter les accidents, bien qu'ils partagent la route avec des conducteurs humains dont les comportements sont incertains. Pour prendre en compte cette incertitude, nous nous appuyons sur les observations en ligne de l'environnement pour construire une région de confiance autour de la dynamique du système, qui est ensuite propagée au cours du temps pour borner l'ensemble des trajectoires possibles des véhicules à proximité. Pour assurer la sûreté en présence de cette incertitude, nous avons recours à la prise de décision robuste, qui préconise de toujours considérer le pire cas. Cette approche garantit que la performance obtenue pendant la planification sera également atteinte sur le système réel, et nous montrons dans une analyse de bout en bout que la sous-optimalité qui en résulte est bornée. Nous en fournissons une implémentation efficace, basée sur des algorithmes de recherche arborescente.

Une seconde contribution est motivée par le constat que cette approche pessimiste tend à produire des comportements excessivement prudents : imaginez vouloir dépasser un véhicule, quelle certitude avez-vous que ce dernier ne changera pas de voie au tout dernier moment, provoquant un accident ? Ce type de raisonnement empêche les robots de conduire aisément parmi d'autres conducteurs, de s'insérer sur une autoroute ou de traverser une intersection, un phénomène connu sous le nom de « *robot figé* ». Ainsi, la présence d'incertitude induit un compromis entre deux objectifs contradictoires : sûreté et *efficacité*. Comment arbitrer ce conflit ? La question peut être temporairement contournée en réduisant au maximum l'incertitude. Par exemple, nous proposons une architecture de réseau de neurones basée sur de l'attention, qui tient compte des interactions entre véhicules pour améliorer ses prédictions. Mais pour aborder pleinement ce compromis, nous nous appuyons sur la prise de décision sous contrainte afin de considérer indépendamment les deux objectifs de sûreté et d'efficacité. Au lieu d'une unique politique de conduite, nous entrainons toute une gamme de comportements, variant du plus prudent au plus agressif. Ainsi, le concepteur du système dispose d'un curseur lui permettant d'ajuster en temps réel le niveau de risque assumé par le véhicule.

Abstract

In this Ph.D. thesis, we study how autonomous vehicles can learn to act *safely* and avoid accidents, despite sharing the road with human drivers whose behaviours are uncertain. To explicitly account for this uncertainty, informed by online observations of the environment, we construct a high-confidence region over the system dynamics, which we propagate through time to bound the possible trajectories of nearby traffic. To ensure safety under such uncertainty, we resort to robust decision-making and act by always considering the worst-case outcomes. This approach guarantees that the performance reached during planning is at least achieved for the true system, and we show by end-to-end analysis that the overall sub-optimality is bounded. Tractability is preserved at all stages, by leveraging sample-efficient tree-based planning algorithms.

Another contribution is motivated by the observation that this pessimistic approach tends to produce overly conservative behaviours: imagine you wish to overtake a vehicle, what certainty do you have that they will not change lane at the very last moment, causing an accident? Such reasoning makes it difficult for robots to drive amidst other drivers, merge into a highway, or cross an intersection –an issue colloquially known as the *"freezing robot problem"*. Thus, the presence of uncertainty induces a trade-off between two contradictory objectives: safety and *efficiency*. How does one arbitrate this conflict? The question can be temporarily circumvented by reducing uncertainty as much as possible. For instance, we propose an attention-based neural network architecture that better accounts for interactions between traffic participants to improve predictions. But to actively embrace this trade-off, we draw on constrained decision-making to consider both the task completion and safety objectives independently. Rather than a unique driving policy, we train a whole continuum of behaviours, ranging from conservative to aggressive. This provides the system designer with a slider allowing them to adjust the level of risk assumed by the vehicle in real-time.

Contents

List of Acronyms

A

ACC	Adaptive Cruise Control , 4
AD	Autonomous Driving , 4, 7, 10, 17, 19, 26–29, 31, 33, 35, 60, 78, 174, 175
ADAS	Advanced Driver-Assistance Systems , 4, 178
ADP	Approximate Dynamic Programming , 84
AEB	Autonomous Emergency Braking , 4
AES	Autonomous Emergency Steering , 4

B

BFTQ	Budgeted Fitted Q-Learning , 61, 67, 68, 70–73, 75–77, 233, 236
BMDP	Budgeted Markov Decision Process , 10, 59, 61–67, 71, 77, 78, 188, 232
BRUE	Best Recommendation with Uniform Estimation , 85, 117

C

CEM	Cross Entropy Method , 84
CMA-ES	Covariance Matrix Adaptation Evolution Strategy , 84
CMDP	Constrained Markov Decision Process , 32, 60–62, 64, 77, 172, 193, 232
CNN	Convolutional Neural Network , 52
CVaR	Conditional Value-at-Risk , 31, 32, 175

D

List of Acronyms

DP Dynamic Programming , 31, 66, 78, 84, 85

DQN Deep Q-Network , 47, 49, 52, 69, 164–167

F

FCN Fully-Connected Network , 51

FTQ Fitted Q-Learning , 67, 68, 71, 73, 76, 77, 233

H

HJI Hamilton-Jacobi-Isaacs , 32

I

i.i.d. independent and identically distributed , 20

IDM Intelligent Driver Model , 39, 51, 179

IRL Inverse Reinforcement Learning , 26, 27

K

KL-OLOP Kullback-Leibler OLOP , 10, 87, 89–92, 97, 98, 101, 117–119, 197, 207, 208, 212, 234, 237

L

LKA Lane-Keeping Assist , 4

LMI linear matrix inequality , 132, 140, 148, 156, 167

LPV Linear Parameter-Varying , 130, 132, 134, 135, 145, 148

LQ Linear Quadratic , 31, 128

LTI Linear Time-Invariant , 132, 133, 148

M

MAB Multi-Armed Bandits , 31, 86, 87

MCTS Monte-Carlo Tree Search , 7, 10, 19, 83–86, 102

MDP Markov Decision Process , 5, 7, 9, 22, 23, 27–29, 31, 32, 35, 40, 41, 45, 60, 64, 65, 71, 83–86, 102–104, 177, 206, 235

MDPGapE MDP Gap-based Estimation , 101, 117

ML Machine Learning

MLE Maximum Likelihood Estimation , 7

MOBIL Minimizing Overall Braking Induced by Lane change , 39, 179

MOMDP Multi-Objective Markov Decision Process , 60–62, 185, 232

MORL Multi-Objective Reinforcement Learning , 60

MPC Model Predictive Control , 11, 126–128, 143, 152–156, 167, 169, 175

N

NN Neural Network , 46, 49, 62, 67, 69, 79

O

OFU Optimism in the Face of Uncertainty , 86, 104, 128, 174

OLOP Open-Loop Optimistic Planning , 10, 83, 86, 87, 89–92, 98, 101, 118, 173, 197, 207, 237

OPC Optimistic Planning algorithm with Continuous actions

OPD Optimistic Planning of Deterministic Systems , 86, 98, 101, 102, 105–110, 112, 116–119, 158, 159, 162, 164, 202, 215, 216, 234, 236

P

PAC Probably Approximately Correct , 84

POMDP Partially Observable Markov Decision Process , 23, 24

PRM Probabilistic Roadmap , 19

R

RL Reinforcement Learning , 5–10, 17, 18, 26–28, 30, 31, 45, 46, 60–62, 65, 78, 83, 122, 123, 172, 174, 176, 177, 180, 182

RRT Rapidly-exploring Random Trees , 19

S

SOOP Simultaneous Optimistic Optimisation for Planning

List of Acronyms

StOP Stochastic Optimistic Planning , 86

U

UCT Upper Confidence bounds applied to Trees , 86, 117

V

VaR Value-at-Risk , 31, 175

List of Symbols

Mathematical notations

$\mathbb{N}$	set of integers
$[n]$	range of integers $\{1, \dots, n\}$
$\mathbb{R}_+$	set of positive reals $\{\tau \in \mathbb{R} : \tau \geq 0\}$
$\mathbb{R}$	set of real numbers
$\mathbb{N}_+$	set of positive integers $\mathbb{N} \cap \mathbb{R}_+$
$\mathcal{L}_\infty$	the set of all inputs u with the property $\|\|u\|\| < \infty$
$\|x\|$	Euclidean norm for a vector $x \in \mathbb{R}^n$
$\|u\|$	L_∞ norm $\|\|u\|\|_{[t_0,t_1]}$ with $t_1 = +\infty$
$\|u\|_{[t_0,t_1]}$	L_∞ norm on $[t_0, t_1)$ of a measurable and locally essentially bounded input $u : \mathbb{R}_+ \to \mathbb{R}$
$\|z\|$	absolute value $\|z\| = z^+ + z^-$
z^-	negative part $z^- = z^+ - z$
z^+	positive part $\max(z, 0)$
e_i	normal basis vectors $[0 \dots 0\ 1\ \dots 0]^\top$ in $\mathbb{R}^n$ for $i = \overline{1, n}$, where 1 appears in the i^{th} position
I_n	the identity matrix with dimension $n \times n$
$E_{n\times m},\ E_p$	the matrices with all elements equal 1 with dimensions $n \times m$ and $p \times 1$, respectively
$M^\top$	transpose of a matrix M
$\|A\|_2$	the induced L_2 matrix norm $\max_{i\in[n]} \lambda_i(A^\top A)$

List of Symbols

$\|A\|_{\max}$	the elementwise maximum norm $\|A\|_{max} = \max_{i\in[n],j\in[n]} \|A_{i,j}\|$, it is not sub-multiplicative
$\lambda(A)$	the vector of eigenvalues of a matrix $A \in \mathbb{R}^{n\times n}$
$P \prec 0$ $(P \succ 0)$	a symmetric matrix $P \in \mathbb{R}^{n\times n}$ is negative (positive) definite
$x_1 \leq x_2$	for two matrices $A_1, A_2 \in \mathbb{R}^{n\times n}$, (including vectors), the relation $A_1 \leq A_2$ is understood elementwise
$o(\cdot), \mathcal{O}(\cdot), \Omega(\cdot)$	Landau notations for positive functions: $f(x) = o(g(x)$ means that $g(x) \neq 0$ and $f(x)/g(x) \to 0$ for $x \to \infty$, $f(x) = \mathcal{O}(g(x))$ means that there exists $x_0, K > 0$ such that $f(x) \leq Kg(x)$ from $x \geq x_0$, and $f(x) = \Omega(g(x))$ means $g(x) = \mathcal{O}(f(x))$
$\mathbb{E}$	expectation under a probabilistic model
$\mathbb{V}$	variance under a probabilistic model
$\mathcal{M}(\mathcal{X})$	set of probability measures on a measurable space $\mathcal{X}$, 5
$\mathcal{B}$	Binomial distribution
δ	Dirac distribution , 46
$\mathcal{N}$	Normal distribution
$\mathcal{U}(\mathcal{X})$	uniform distribution on a measurable space $\mathcal{X}$, 68

Markov Decision Processes

$\mathcal{S}$	set of states $s \in \mathcal{S}$, 5
$\mathcal{A}$	set of actions $a \in \mathcal{A}$
$R(s,a)$	reward function $R : s, a \to R(s,a) \in [0,1]$, 5
$P(s' \mid s,a)$	transition distribution $s' \sim P(s' \mid s,a)$, 5
γ	discount factor in $[0,1)$, 5
π	policy , 5
$\pi^\star$	optimal policy , 6
G	discounted return for the reward signal , 5
V	state value function (* for optimal value, $^\pi$ for policy value) , 6
Q	state-action value function (* for optimal value, $^\pi$ for policy value) , 6
$\mathcal{T}$	Bellman operator (* for optimality, $^\pi$ for evaluation) , 47
r_n	simple regret of an algorithm after n samples , 6

$\mathcal{D}$	dataset , 7

Budgeted Reinforcement Learning

$\overline{\mathcal{S}}$	set of augmented states , 62
$\overline{\mathcal{A}}$	set of augmented actions , 62
$\mathscr{B}$	set of admissible budgets , 62
$\overline{\Pi}$	set of budgeted policies , 62
C	cost function , 60
β	budget , 60
β_a	budget allocated to an action , 62
$\overline{P}$	augmented transition function , 62
$\overline{R}$	augmented reward function , 62
$\overline{\pi}$	budgeted policy , 62
$\overline{\pi}^\star$	optimal budgeted policy , 68
G_c	discounted return for the cost signal
$\overline{G}$	augmented return , 63
V_c	cost value function , 63
V_r	reward value function , 63
$\overline{V}$	augmented value function , 63
Q_c	cost Q-function , 63
Q_r	reward Q-function , 63
$\overline{Q}$	augmented Q-function , 63
$\overline{\mathcal{T}}$	augmented Bellman operator (* for optimality, $^\pi$ for evaluation) , 63

Tree-based Planning

$\mathcal{A}^*$	set of a finite words a, representing sequences of actions $(a_1, \dots, a_h) \in \mathcal{A}^h$, for $h \in \mathbb{N}$
$\mathcal{A}^\infty$	the set of infinite sequences of actions $(a_1, \dots)$
$ab \in \mathcal{A}^*$	the concatenation of two finite sequences $a \in \mathcal{A}^*$ and $b \in \mathcal{A}^*$
$\emptyset$	the empty sequence of actions

List of Symbols

$a_{1:t}$	the prefix $(a_1, \dots, a_t) \in \mathcal{A}^t$ of length $t \leq h$ of a word $a \in \mathcal{A}^h$
$a\mathcal{A}^*$, $a\mathcal{A}^\infty$	the set of finite and infinite suffixes of a, respectively: $a\mathcal{A}^* = \{c \in \mathcal{A}^\star : \exists b \in \mathcal{A}^* \text{ such that } c = ab\}$ and $a\mathcal{A}^\infty$ defined likewise
$\mathcal{T}$	the look-ahead tree
$\nu(a)$, $\mu(a)$	the distribution and mean of the reward obtained at the last step after executing a sequence of actions $a \in \mathcal{A}^*$
U_a, L_a	upper and lower bounds on the value $V(a)$ of a sequence of actions $a \in \mathcal{A}^*$
κ	effective branching factor of a planning tree $\mathcal{T}$, in $[1, K]$

Linear Systems

$\mathbb{X}$	constraint set for safe states $x(t) \in \mathbb{X} \subset \mathbb{R}^p$, 126
$\mathbb{U}$	constraint set for safe controls $u(t) \in \mathbb{U} \subset \mathbb{R}^q$, 126
$\mathrm{d}t$	time step at which MPC controls are applied , 122
ϕ	features for a parametrized , 126
θ	parameters $\theta \in \mathbb{R}^d$ for a model
$A(\theta)$	structured state matrix $A(\theta) \in \mathbb{R}^{p\times p}$, depending on unknown parameters θ , 126
δ	confidence level for statistical estimates, in $(0, 1]$, 124
$\mathcal{C}_{[N],\delta}$	high-confidence set for the estimation of θ, such that $\mathbb{P}(\theta \in \mathcal{C}_{[N],\delta}) \geq 1 - \delta$. , 124
$\underline{x}(t)$	upper-bound of the state interval $\overline{x}(t) \geq x(t),\ \forall t$, 125
$\underline{x}(t)$	lower-bound of the state interval $\underline{x}(t) \leq x(t),\ \forall t$, 125
$G_{N,\lambda}$	Gramian matrix for the regularised least-square estimation of θ, with N samples and penalty λ. , 129
$\theta_{N,\lambda}$	regularised least-square estimate of θ, with N samples and penalty λ. , 129
$\underline{R}$	pessimistic reward function, evaluated on the worst-case reachable states , 157
N	number of transition samples
K	number of planning iterations

Chapter 1

Introduction

We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time.

T. S. Eliot, *Little Gidding*.

1.1 Context and scope

1.1.1 How should a driving robot make decisions?

In the first few weeks of my Ph.D., I observed that layman interlocutors, when confronted with this question on the occasion of a social dinner, have a general tendency to conjure up disaster scenarios involving imminent accidents with unavoidable casualties. This reflex is likely to stem from the popularisation of the Trolley Problem (Foot, 1967), a famous thought experiment in moral philosophy, depicted in Figure 1.1, in which a runaway trolley is headed straight toward five people tied up on the main track and unable to move. When pulled, a lever switches the trolley to a side track occupied by one person: what should you do? Answering this general question of what we *ought* to do in any situation, what is a *right* or *wrong* decision, is the focus of the field of normative ethics. This dilemma illustrates a clash between two schools of thought: utilitarianism and deontological ethics. According to utilitarians, the rightfulness of an action should be evaluated based on its consequences, and actions maximising a *utility* –the happiness and well-being for the affected individuals– should be preferred. Conversely, deontologists evaluate the morality of actions *per se*, according to a series of rules, rather than based on their consequences. Although this problem was initially introduced as a thought experiment, its transposition to the context of autonomous driving and arguably more realistic

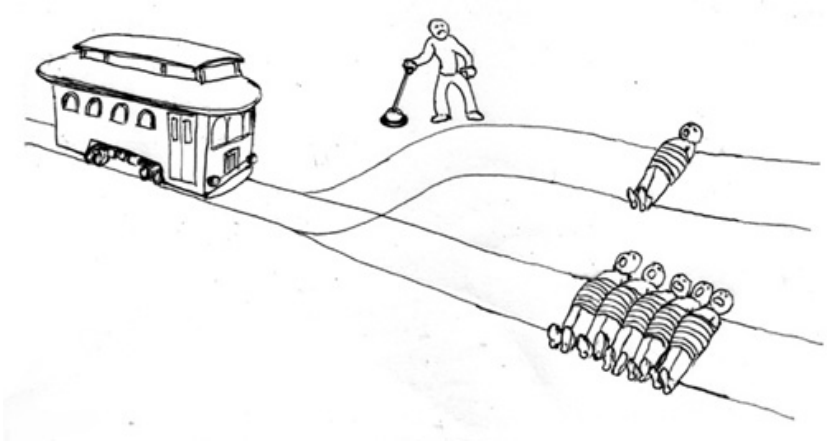


Figure 1.1 – The Trolley Problem (Foot, 1967). Illustration by Jesse J. Prinz.

scenarios made it heavily cited in discussions regarding safety (*e.g.* Lin, 2015; Bonnefon, Shariff, and Rahwan, 2016; Gogoll and Müller, 2017). In early 2017, MIT's Media Lab launched the *Moral Machine* platform (Awad et al., 2018), in which they invited members of the public to select the morally acceptable decision out of several options available to an autonomous vehicle. The authors argued that the recovered global preference would provide *"essential topics to be considered by policymakers"*, and Noothigattu et al. (2018) proposed an implementation of a system aggregating these preferences, trained on the collected data. However, the relevance of this analogy to inform engineering and policy has been called into question. Thus, De Freitas et al. (2019) point out that such dilemmas are unlikely to occur on real roads, hard to detect by perception systems and to act upon by control systems, and that they are distracting researchers from the more appropriate goal of how to avoid accidents altogether. Indeed, when we drive, we seldom find ourselves in such extreme situations but rather constantly ponder over less tragic considerations: Where does this vehicle intend to go? Do I have the time to proceed or should I yield? What is the appropriate speed to drive at right now? The object of this thesis is to artificially reproduce this cognitive process of how to avoid accidents while driving, which is more a technical matter than an ethical one. Still, the Trolley Problem, though unrealistic, reveals and raises a number of legitimate questions. Should we base driving decisions on a set of rules? Can these rules be learned, *e.g.* by imitating human drivers? Should we instead make decisions by comparing the utility of possible outcomes, like utilitarians advocate? And if so, how do we choose a good utility? This last interrogation becomes particularly sensitive if we add uncertainty to the Trolley Problem. Facing a potential collision, driving slowly decreases the risk of accident at the expense of efficiency, which can ultimately have an economical impact. What is the *right* level of caution to take? This question directly translates as that of the *value of life*, which has been taken up by economists for decades (Abraham and Thédié, 1960; Drèze, 1962; Schelling, Bailey, and Fromm, 1968; Banzhaf, 2014; Tirole and Rendall, 2017; Charpentier and Cherrier, 2019) and has countless practical implications for public policies,

including recent debates on lowering the speed limits on highways and trunk roads, but also on the appropriate lockdown durations during a pandemic. It would be illusory to pretend that the practical implications of the Trolley Problem can simply be swept aside and entirely replaced by technicality. Throughout this manuscript, we will see that ethical concerns still underpin most assumptions and design choices of safety-critical software.

1.1.2 Nuts and bolts of self-driving software

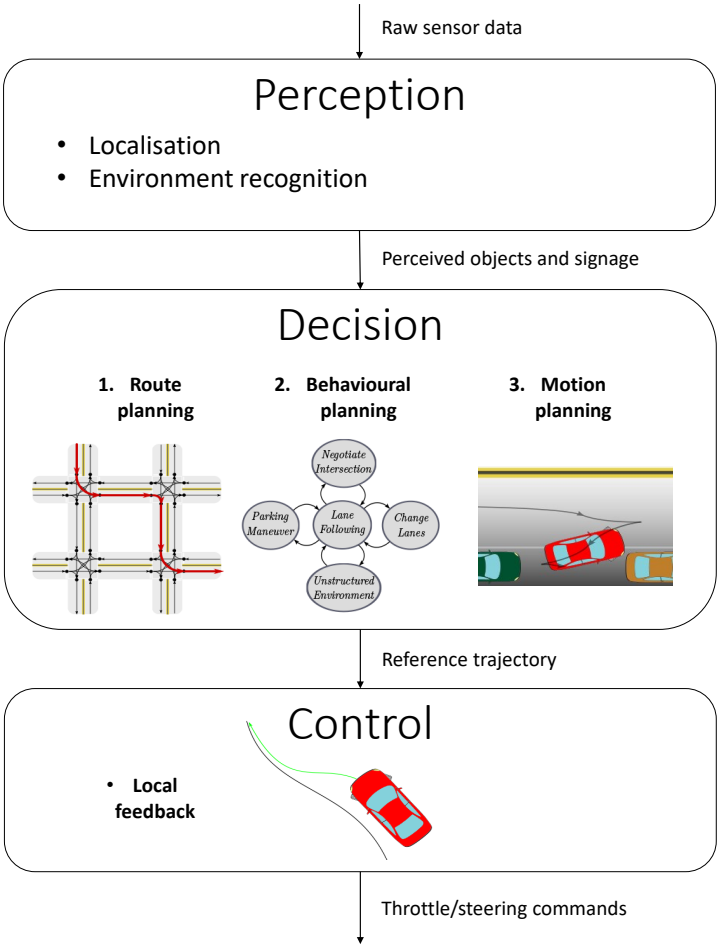


Figure 1.2 – The architecture of a typical self-driving software

Historically, autonomous vehicles have been developed following a traditional robotics pipeline, illustrated in Figure 1.2. This architecture decomposes the task of driving as a series of three functions: *Perception*, *Decision*, and *Control* (also called the Sense-Plan-Act paradigm, or Navigation, Guidance and Control in aerospace engineering). The Perception module takes raw sensor data as input and produces a high-level reconstruction of the scene. The Decision module then determines the desired trajectory of the vehicle, based on the current situation.

Finally, the Control module manipulates forces, by way of steering and throttle controls, to track the desired trajectory. In the context of Autonomous Driving, the Decision module is often implemented with a hierarchical structure whose layers work at different timescales. First, a *Route Planning* layer searches for the shortest route in a road network from the current location to the desired destination. Second, the *Behavioural Planning* layer specifies a coarse driving behaviour through short-term goals or semantic decisions, such as changing lane, slowing down at an intersection, or yielding to a vehicle. This layer is thus responsible for following the planned route while adapting to the current state of the traffic in real-time. Third, the *Motion Planning* layer generates a continuous, feasible trajectory that implements the desired behaviour while ensuring comfort and safety.

Great strides have been made in the two end-of-pipe tasks: Perception has benefited from the substantial progress in the field of computer vision due to the recent advent of deep learning (surveyed in Janai et al., 2020), and many Control schemes (surveyed in Polack, 2018) have been developed for ground vehicles. In the Decision module, Route Planning is virtually solved and already provided by services such as Open Street Maps, and there exist a vast body of Motion Planning algorithms, discussed in Chapter 2. All these building blocks are widely used for industrial applications, including Advanced Driver-Assistance Systems (ADAS) functions such as Lane-Keeping Assist (LKA), Adaptive Cruise Control (ACC), Autonomous Emergency Braking (AEB) or Autonomous Emergency Steering (AES); and in academic research challenges. Ultimately, we claim that Behavioural Planning remains the only neglected link in the chain. Indeed, most of these applications focused so far on simple settings with little complexity: ADAS systems are mostly tailored for highway driving and struggle whenever required to interact with other drivers, *e.g.* for merging into traffic[1]. Similarly, most academic challenges focused on highway driving, with the exception of the DARPA Urban Challenge, which required more advanced interactions with other vehicles. Nevertheless, even this event still constituted a controlled environment, simple enough that all participants could rely on rule-based systems for behavioural planning (Buehler, Iagnemma, and Sanjiv Singh, 2009), such as finite state machines whose transitions are triggered by handcrafted criteria (*e.g.* Baker and Dolan, 2008). Unfortunately, there is little hope that this approach can scale to complex scenes since responses tailored for specific use-cases cannot be easily merged.

1.1.3 Scope and Challenges of this Thesis

In the light of the above, this thesis is dedicated to addressing a weak link in the Autonomous Driving (AD) chain: *Behavioural Planning*. We ask the following question: assuming that we had access to a ground truth perception and a perfectly accurate control system, what steps would remain to achieve fully autonomous driving?

[1] This difficulty, which motivated this thesis, was reported by engineers from the ADAS team at Renault.

Humans in the loop Unless restrained to dedicated infrastructure, Autonomous Vehicles will have to share the road with human drivers. This introduces a great deal of uncertainty in the decision problem. Indeed, while the location and velocity of a vehicle can be perceived, the mind of its driver remains impenetrable. Even though the present state is known, the future becomes uncertain: where are they headed? Are they paying attention to their surroundings? In that regard, it seems impossible to manually model all the factors involved in the human decision-making process. However, human drivers do not drive erratically either, and their behaviour is highly structured: humans drivers tend to follow the lanes, avoid collisions with other vehicles, and generally respect road signage. In other words, human drivers are *predictable*. This motivates the idea of learning from data, and hope for a better comprehensiveness than handcrafted decision systems.

Learning to act The skill of driving a car involves taking a series of decisions, where early stages influence the resulting outcomes and subsequent reasoning at late stages. This aspect is known as sequential (or multistage) decision-making. Let us start by introducing some useful notations. At each time step t, the system is described by its *state* s_t that belongs to a measurable[2] state space $\mathcal{S}$. Then, the agent can take an *action* a_t within a measurable action space $\mathcal{A}$, before transitioning to a next state $s_{t+1} \in \mathcal{S}$, drawn from a conditional distribution $P(s_{t+1} \mid s_t, a_t)$ that we call the *system dynamics* $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$, where $\mathcal{M}(\mathcal{X})$ denotes the set of probability measures of a measurable set $\mathcal{X}$. The agent actions can themselves be drawn from a distribution $\pi(a_t \mid s_t)$, called the *policy* $\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}$.

Reinforcement Learning (RL) is a general framework for learning-based sequential decision-making. It is formulated as an optimal control problem: the policy π is chosen to maximise an objective function. It is generally formalised as a Markov Decision Process (MDP), *i.e.* a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ in which at each step t, the agent receives a bounded reward $R(s_t, a_t)$, where $R \in [0, 1]^{\mathcal{S}\times\mathcal{A}}$ is a deterministic reward function and $\gamma \in [0, 1)$ is a discount factor. Adequate long-term behaviour of policies π is fostered by considering their *return*.

Definition 1.1 (Policy return)**.** *The return G^π of a policy π is a random variable defined as the discounted sum of rewards*

$$G^\pi = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

accumulated along a trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$ induced by the policy $a_t \sim \pi(a_t|s_t)$ and system dynamics $s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$.

The performance of a policy π is then evaluated through its *value* function.

[2] A measurable space is a set with a σ-algebra, that allows to define random variables. For example, this set can finite ($[N]$), countable ($\mathbb{N}$), or continuous ($\mathbb{R}$).

Definition 1.2 (Value functions)**.** *The state value $V^\pi(s)$ of a policy π is the expected return of the policy when starting in a state s*

$$V^\pi(s) \triangleq \mathbb{E}\left[G^\pi \mid s_0 = s\right].$$

Similarly, the state-action value $Q^\pi(s,a)$ of a policy π is the expected return of the policy when starting in the state s and taking the action a

$$Q^\pi(s,a) \triangleq \mathbb{E}\left[G^\pi \mid s_0 = s, a_0 = a\right].$$

This allows to define the goal of Reinforcement Learning: finding an *optimal* policy $\pi^\star$.

Definition 1.3 (Optimality)**.** *A policy $\pi^\star$ is said to be optimal if it maximises the value functions V^π and Q^π in every state and action. We also define the optimal value functions $V^\star$ and $Q^\star$ as*

$$\forall s \in \mathcal{S}, \qquad V^\star(s) \triangleq Q^{\pi^\star}(s) = \max_\pi V^\pi(s);$$
$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \qquad Q^\star(s,a) \triangleq Q^{\pi^\star}(s,a) = \max_\pi Q^\pi(s,a).$$

Sample efficiency Several performance measures have been introduced to evaluate RL algorithms. In this thesis, we consider the goal of finding a near-optimal policy as fast as possible. The *fixed-confidence* setting evaluates the smallest sample complexity, *i.e.* number of interactions, required to find a near-optimal policy $\pi^\star$ with high probability. Alternatively, in the *fixed-budget* setting, the *simple regret* r_n of an algorithm measures the *expected* sub-optimality of the recommended policy $\hat{\pi}_n$ after a fixed number n of interactions

$$r_n(s) \triangleq \mathop{\mathbb{E}}_{\hat{\pi}_n}\left[V^\star(s) - V^{\hat{\pi}_n}(s)\right].$$

The goal of this thesis is to provide *sample-efficient* algorithms for learning a driving policy. In the particular context of Behavioural Planning for Autonomous Driving, this goal will be articulated around a few main questions and challenges.

Model-free *vs.* model-based Reinforcement Learning algorithms can be grouped into two main families. To find an optimal policy $\pi^\star$, model-based Reinforcement Learning algorithms

first attempt to estimate the MDP parameters $\hat{P}$ and $\hat{R}$ based on a history of transitions $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}$, using for instance Maximum Likelihood Estimation (MLE) in a hypothesis class of dynamics and reward functions:

$$\max_{\hat{P}} \prod_t \hat{P}(s_{t+1} \mid s_t, a_t) \quad \text{and} \quad \min_{\hat{R}} \sum_t \|R(s_t, a_t) - \hat{R}(s_t, a_t)\|_2^2.$$

This allows to *plan* in the estimated MDP $(\mathcal{S}, \mathcal{A}, \hat{P}, \hat{R}, \gamma)$, *i.e.* compute the associated optimal policy $\hat{\pi}^\star$. This can be achieved using planning algorithms such as Dynamic Programming or Linear Programming. Conversely, model-free RL algorithms do not estimate the underlying MDP and aim instead to optimise a policy π directly. Policy-based methods evaluate the value Q^π of the current policy π, so that it can be locally improved *e.g.* by gradient ascent. Value-based method bypass this alternation of evaluation and improvement steps by directly learning the optimal value function $Q^\star$.

The question of which approach is appropriate depends on the underlying problem. Indeed, model-based techniques are relevant when the dynamics are simple but the optimal policy is complex. For instance, the case of Computer Go was tackled in AlphaGo (Silver, Huang, et al., 2016; Silver, Schrittwieser, et al., 2017; Silver, Hubert, et al., 2018) with a Monte-Carlo Tree Search (MCTS) planning module, which leveraged the knowledge of the Go dynamics (placing pawns on the board) to sample sequences of plies in a game tree. On the contrary, the model-free approach is useful when the system dynamics are complex, but the optimal policy is simple. Thus, a swimming robot would require massive fluid dynamics simulation to accurately predict the effect of moving its fins, while a simple periodic gait could suffice to propel it forward in the water. Which brings us to the question: which scenario does AD fall into? Unfortunately, the answer is not so clear-cut. On the one hand, the motion planning literature has historically been heavily relying on kinematics and dynamics models to plan trajectories, as detailed in Chapter 2. Reliable priors are available to describe the physics of a vehicle, but not so much for the actual driving policy. On the other hand, automotive companies such as Mobileye reportedly[3] advocated that the task of predicting a driving scene is actually more difficult than that of driving. In fact, the case of Autonomous Driving is peculiar in that the two problems of prediction and control are somewhat equivalent, due to the presence of other drivers: a good trajectory predictor can be used to predict the action that a human would take in place of the ego-vehicle, and a good driving policy can be applied to each agent in the scene to produce reasonable predictions of their trajectory. Hence, both approaches seem equally relevant and will be considered in this thesis.

Social interactions and coupled dynamics To ensure safety while driving, traditional motion planning techniques rely on conservative independence assumptions on the behaviour of other

[3]These comments are reported from discussions between Renault and Mobileye.

vehicles. This makes them suffer from an effect colloquially known as the *"freezing robot problem"* (Trautman and Krause, 2010): due to massive uncertainty, these systems tend to struggle in situations that require interacting –or negotiating– with other vehicles, such as unprotected left-turns, intersections, or highway merges (see *e.g.* these two videos where a Waymo car fails to merge). Thus, for an autonomous vehicle to efficiently integrate with the traffic flow, it must anticipate the effect of its own actions on the behaviour of other agents. This skill is known as *socially-aware* decision-making. Unfortunately, interactions between vehicles translate into complex and coupled traffic dynamics where local deviations must be propagated from one vehicle to the next, which can result in a quick and chaotic build-up of uncertainty. We will need to contain this escalation to prevent instability in our predictions.

Ensuring safety The complexity of the task of driving leads us to consider learning algorithms. However, these methods typically raise a legitimate concern among car manufacturers: how can we guarantee the *safety* of such systems, in the presence of uncertainty? In this thesis, we will strive to formalise this concern by formulating different notions of *risk*, as functions of the distribution of outcomes induced by a policy. We will study *Safe* Reinforcement Learning algorithms, that seek to explicitly estimate and control the level of risk taken by a policy.

Balancing safety and efficiency However, protecting against risk often comes at the price of efficiency in achieving a goal. Consider for a moment a situation where a vehicle is driving slowly in front of you on the road, and so you decide to overtake them. How do you know, at that very moment, that the driver has seen you and is not going to change lane at the last moment, causing an accident? In fact, you don't, at least not with certainty. In this situation, the only way to guarantee safety is to refrain from overtaking. By pushing this simple argument through to its conclusion, it becomes apparent that the only way to *fully* ensure safety is to stay in the garage. You are thus facing an irreducible dilemma: the two objectives of driving fast and safely are contradictory. This opposition induces a trade-off that we typically observe with human drivers, especially in situations of negotiations: some people adopt *aggressive* behaviours and try to force their way through traffic, while others act more *defensively*, favouring safety. This ambiguity is difficult to capture manually in an objective function and soon leads to the pitfall of reward engineering. To provide a principled control over this trade-off, the learning algorithms that we consider will be required to respect an *adjustable* level of risk.

1.2 Outline and Contributions

The ultimate goal of this thesis could be summarised in the following question: *"how can an algorithm learn to drive and avoid accidents?"*. The first step in such an endeavour must necessarily

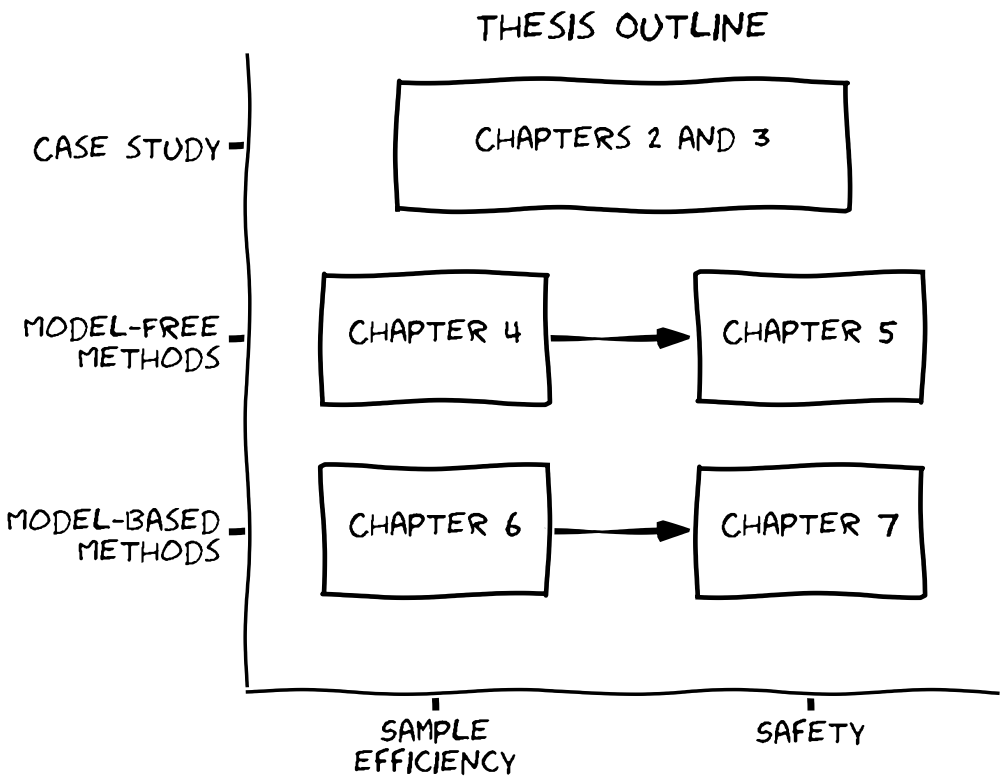


Figure 1.3 – This thesis is structured around two disjunctions: model-free *vs.* model-based on the one hand, and sample-efficiency *vs.* safety on the other hand.

be to formalise more precisely the meaning of this ill-posed formula, which we try to do in **Part I**. It is only natural that we begin this effort by turning to the standard model for sequential decision making: the Markov Decision Process. At first glance, this framework shines with its simplicity and elegance, but also its apparent generality and representation power. Yet, as we embark on the ambitious task of casting the blurry problem of autonomous driving into this rigid mould, we highlight in **Chapter 2** how reductive each step of the formalisation is, how approximations and assumptions always hide behind each symbol and each equation. This observation is supported by the numerous variations of the framework developed by the research community, in as many attempts to address these concerns. Such limitations are as varied as partial observability, temporal abstraction, the reward hypothesis, transfer from simulation to real-world and safety; and we relate these research directions to specific works in the autonomous driving literature.

In order to progress, we put aside some of these questions in **Chapter 3** and commit to an (observable) state space, a (hierarchical) action space, a (quasi-linear) system dynamics and a (dense) reward function that we deem suitable for a large class of behavioural planning tasks. This allows us to refocus on two fundamental issues: *sample-efficient* and *safe* Reinforcement

Learning. In the sequel we tackle them through the perspective of the two main approaches to Reinforcement Learning aforementioned: first model-free, and then model-based algorithms. This organisation is depicted in Figure 1.3.

Part II is dedicated to the study of how model-free methods can be applied for efficient and safe Autonomous Driving. In **Chapter 4**, we question the choice of state representation and model architecture in relation to their associated sample-efficiency. In particular, we identify desirable properties and inductive biases that the policy should enjoy, such as *permutation invariance* with respect to vehicles in the scene. We propose an attention-based architecture that fulfils our criteria, and compare it to standard representations and model architectures that have been used for behavioural planning tasks.

In **Chapter 5**, we consider a continuous notion of risk, defined as an expected discounted sum of a cost signal. This formulation allows highlighting a trade-off between two separate objectives: the traditional return associated with task completion, and the risk related to safety. In this multi-objective perspective, the Pareto frontier of non-dominated policies defines a spectrum of behaviours, from risk-averse on one side to risk-seeking on the other. In order to explicitly control the level of risk taken in real-time, we place ourselves within the Budgeted Markov Decision Process (BMDP) framework, in which the risk is constrained to lie below an –adjustable– threshold. So far, BMDPs could only be solved in the case of finite state spaces with known dynamics. This chapter extends the state-of-the-art to environments with continuous state space and unknown dynamics. We show that the solution to a BMDP is a fixed point of a novel Budgeted Bellman Optimality operator, which enables to estimate both the expected return and risk of an action, in a model-free fashion. This observation allows us to introduce natural extensions of Deep Reinforcement Learning algorithms to address large-scale BMDPs.

Part III is devoted to the study of model-based methods, that solve the Reinforcement Learning problem by planning with a learned generative model. In **Chapter 6**, we assume that a reliable generative model has already been learnt and focus on the sample-efficiency of the planning procedure specifically. More precisely, we look into the theoretical and practical aspects of planning algorithms under budget constraints. First, we consider the `Open-Loop Optimistic Planning (OLOP)` algorithm that enjoys good theoretical guarantees but is overly conservative in practice, as we show in numerical experiments. We propose a modified version of the algorithm with tighter upper-confidence bounds, `Kullback-Leibler OLOP (KL-OLOP)`, that leads to better practical performances while retaining the sample complexity bound. Second, we study a limitation of MCTS algorithms: they do not identify together two similar states reached via different trajectories and represented in separate branches of the tree. We propose a *graph-based* planning algorithm, which takes into account this state similarity, provide a regret bound that depends on an improved problem-dependent measure of difficulty, and illustrate its empirical benefits numerically.

In **Chapter 7**, we look back into the issue of *model bias*, which refers to the gap that exists between a learned model and the true system dynamics, and can dramatically degrade the performance of the planned trajectory. More specifically, we study the problem of *robust* and *adaptive* Model Predictive Control (MPC) of a linear system, with unknown parameters that are learned along the way (adaptive), in a critical setting where failures must be prevented (robust). To that end, instead of merely considering a point estimate of the dynamics, we leverage non-asymptotic linear regression to build an entire *confidence region* that contains the true dynamics with high probability. To effectively propagate this parametric uncertainty, we design a predictor that produces a tight interval hull bounding the system trajectories. Having observed the instability of traditional interval predictor techniques, we propose a new one whose stability is guaranteed by a Lyapunov function analysis and verification of linear matrix inequalities. These tools enable us to guarantee the system stabilisation and robust constraint satisfaction, through an MPC algorithm based on a stabilising control that uses the predicted interval. Finally, in order to go beyond stabilisation problems only, we tackle the minimax control of more general (non-convex) costs that naturally arise in many practical problems. To that end, we combine our results with the tree-based planning techniques of Chapter 6. By adapting the theoretical guarantees at each layer, we provide the first end-to-end regret analysis for this setting. Interestingly, our analysis naturally adapts to handle multiple models and combines with a data-driven robust model selection strategy, which enables to relax the modelling assumptions. We strive to preserve tractability at any stage of the method, that we illustrate numerically.

List of publications

Publications in international conferences with proceedings

- Edouard Leurent, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2020a). Robust-Adaptive Control of Linear Systems: beyond Quadratic Costs. In *Advances in Neural Information Processing Systems 33*. Virtual (used in Chapter 7)
- Edouard Leurent and Odalric-Ambrym Maillard (Nov. 2020a). Monte-Carlo Graph Search: the Value of Merging Similar States. In *Asian Conference on Machine Learning (ACML 2020)*. Ed. by Sinno Jialin Pan and Masashi Sugiyama. Bangkok, Thailand, pp. 577–592 (used in Chapter 6)
- Edouard Leurent, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2020b). Robust-Adaptive Interval Predictive Control for Linear Uncertain Systems. In *2020 IEEE 59th Conference on Decision and Control (CDC)*. Jeju Island, Republic of Korea (used in Chapter 7)
- Nicolas Carrara, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Odalric-Ambrym Maillard, and Olivier Pietquin (Dec. 2019). Budgeted Reinforcement Learning in Con-

tinuous State Space. In *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 9299–9309 (used in Chapter 5)

- Edouard Leurent, Denis Efimov, Tarek Raissi, and Wilfrid Perruquetti (Dec. 2019). Interval Prediction for Continuous-Time Systems with Parametric Uncertainties. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. Nice, France, pp. 7049–7054 (used in Chapter 7)

- Edouard Leurent and Odalric-Ambrym Maillard (Sept. 2020b). Practical Open-Loop Optimistic Planning. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet. Würzburg, Germany: Springer International Publishing, pp. 69–85 (used in Chapter 6)

Workshop presentations in international conferences

- Edouard Leurent and Jean Mercat (Dec. 2019). Social Attention for Autonomous Decision-Making in Dense Traffic. In *Machine Learning for Autonomous Driving Workshop at the Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019)*. Montreal, Canada (used in Chapter 4)

- Edouard Leurent, Yann Blanco, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2018). Approximate Robust Control of Uncertain Dynamical Systems. In *Machine Learning for Intelligent Transportation Systems Workshop at the Thirty-second Conference on Neural Information Processing Systems (NeurIPS 2018)*. Montreal, Canada (used in Chapter 7)

Software

- Edouard Leurent (2018). *An Environment for Autonomous Driving Decision-Making*. https://github.com/eleurent/highway-env. GitHub repository (used in Chapters 3 to 7)

Collaborations not presented in this thesis

- Pierre Ménard, Omar Darwiche Domingues, Anders Jonsson, Emilie Kaufmann, Edouard Leurent, and Michal Valko (July 2020). *Fast active learning for pure exploration in reinforcement learning*. Research Report. DeepMind

- Emilie Kaufmann, Pierre Ménard, Omar Darwiche Domingues, Anders Jonsson, Edouard Leurent, and Michal Valko (2020). *Adaptive Reward-Free Exploration*. Submitted to ALT 2021, under review.

- Anders Jonsson, Emilie Kaufmann, Pierre Ménard, Omar Darwiche Domingues, Edouard Leurent, and Michal Valko (Dec. 2020). Planning in Markov Decision Processes with Gap-Dependent Sample Complexity. In *Advances in Neural Information Processing Systems 33*. Virtual

Part I

Case Study: Learning to Drive

Chapter 2

Literature Review

Νὰ εὔχεσαι νά ῾ναι μακρὺς ὁ δρόμος.
.....
Σὲ πόλεις Αἰγυπτιακὲς πολλὲς νὰ πᾷς,
νὰ μάθεις καὶ νὰ μάθεις ἀπ᾽ τοὺς σπουδασμένους.
Κωνσταντίνος Καβάφης, Ἰθάκη.

This chapter provides an overview of the sequential decision-making literature in the specific context of Autonomous Driving. It is meant to summarise the main directions that researchers have taken to tackle this wide problem, and discuss the questions that one practitioner may ask themselves when trying to apply Reinforcement Learning to Autonomous Driving. Thus, I will prioritise breadth rather than depth, and remind the reader that each of these sections is not comprehensive and has been surveyed independently.

Contents

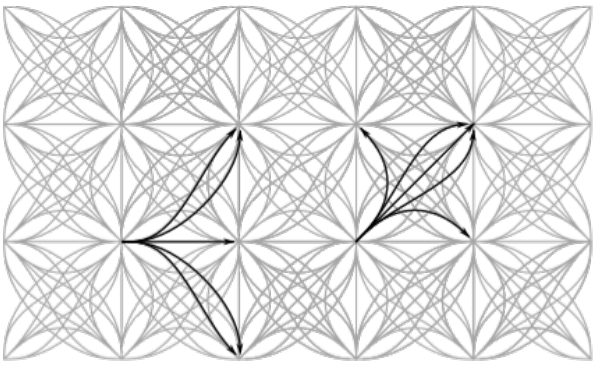


Figure 2.1 – A lattice structure connects a discrete set of states by feasible trajectories

2.1 Sequential decision-making

Section 1.1.3 presented the Reinforcement Learning framework, that formulates the learning procedure as an optimal control problem. In this section, I start by recalling other design principles that have been considered for coming up with a good driving policy π.

2.1.1 Motion Planning

The development of motion planning techniques for intelligent vehicles dates back to the late 80s, supported by international research projects such as Eureka (1987) of the Prometheus program, followed by the DARPA Grand and Urban Challenges (2004, 2007), and more recently the VIAC (2010), GCDC (2011) and Delphi (2015) challenges. In two surveys (González et al., 2016; Paden et al., 2016) studying the literature of this period, the authors identified three main approaches.

Search-based algorithms This method is based on a regular discrete partition of the state space $\mathcal{S}$ called a *lattice*, which must be connected by feasible trajectories (*e.g.* Pivtoraiko and Kelly, 2005). This framing reduces motion planning to the problem of finding a shortest path in a known graph. Then, traditional graph-search algorithms such as Dijkstra's algorithm (Dijkstra, 1959), $A^\star$ (Hart, Nilsson, and Raphael, 1968) or $D^\star$ (Stentz, 1994) can be used to compute the optimal trajectory. This technique has been applied by at least five different teams during the DARPA Urban Challenge for driving on structured roads and unstructured parking: Dijkstra for team Ben Franklin (Bohren et al., 2008) and VictorTango (Bacha et al., 2008), and $A^\star$ for Stanford University (Montemerlo et al., 2008) and KIT (Kammel et al., 2008), and $D^\star$ by the winning team from CMU (Urmson et al., 2008). Kinematics constraints can also be included in the configuration graph, as in (*e.g.* Latombe, 1991; T. Fraichard, 1993; Laumond et al., 1994).

Sampling-based algorithms The limitation of search-based algorithms lies in the difficulty of formulating a regular lattice structure covering the states space $\mathcal{S}$ with feasible transitions, and in the real-time constraint that may not be met by graph-search algorithms. To address them, sampling-based motion planners iteratively grow a set of reachable configurations by randomly sampling valid transitions. The most popular ones are Probabilistic Roadmap (PRM) (Kavraki et al., 1996), Rapidly-exploring Random Trees (RRT) (Lavalle, 1998; Karaman and Frazzoli, 2011) and MCTS algorithms (Coulom, 2007b; Kocsis and Szepesvári, 2006). These methods have been used in the context of Autonomous Driving in (*e.g.* Lamiraux and Lammond, 2001; Sánchez L., Zapata, and Arenas B., 2002; D. Lenz, Kessler, and Knoll, 2016; Paxton et al., 2017; Faust et al., 2018).

Optimisation-based algorithms The third approach consists in optimising a parametrised trajectory with respect to a real-valued objective function. The most popular instance is interpolation between the current and goal states, which has been applied to various classes of functions in the context of Autonomous Driving, such as lines and circles (Reeds and Shepp, 1990), clothoids (Funke et al., 2012), polynomials (W. Xu, Wei, et al., 2012), and Bézier curves (González et al., 2016; Artuñedo, Villagra, and Godoy, 2019).

These three approaches to motion planning thus rely on deterministic models of the vehicle dynamics. These models are often required to take a simple form so that the search or optimisation procedure can be solved efficiently, and other objects in the scene are often considered as static. In order to study more complex multi-agent interactions specifically, a collaborative approach to motion planning has been developed.

Cooperative planning The difficulty of predicting intricate interaction patterns between multiple agents can be bypassed in one particular setting: cooperative motion planning for multiple vehicles. Indeed, instead of predicting how vehicles react to one another, the behaviours of these vehicles are jointly optimised. As an effect, prediction outputs are replaced by input variables that can be chosen freely to maximise an objective function. Two main variations have been studied: coordination along fixed paths (Altché, Qian, and La Fortelle, 2016; Altché and La Fortelle, 2016; Altché, Qian, and de La Fortelle, 2017), and general unconstrained motion planning (LaValle and Hutchinson, 1998). However, this framework does not allow to represent human behaviours, or more generally any behaviour that is not explained by the objective function. In particular, that lack of communication between agents and the resulting uncertainty lead to suboptimal, uncertain and multimodal trajectories that are not handled by cooperative planning approaches.

2.1.2 Imitation Learning

An orthogonal strategy to motion planning techniques is to learn a reactive policy $\pi(a|s)$ under supervision of an expert π_E that produces a dataset $\mathcal{D}$ of demonstration trajectories. To that end, we optimise a parametrised policy π_θ to minimise a regression loss $\mathcal{L}$, such as the KL divergence to the distribution of expert action:

$$\min_\theta \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[\mathcal{L}\left(\pi_\theta(a|s), \pi_E(a|s)\right)\right]$$

This approach is also referred to as behavioural cloning, and is particularly suited when only low-level high-dimensional inputs are available, such as camera images, which prevents access to the dynamics model required by motion planning approaches. The first application of imitation learning to autonomous driving is the ALVINN (Autonomous Land Vehicle In a Neural Network) project (Pomerleau, 1989), where a 3-layer neural network was trained for the task of road following, as shown in Figure 2.2.

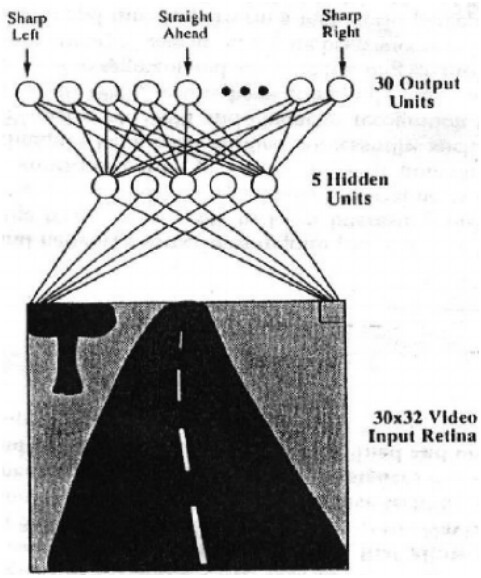


Figure 2.2 – The 3-layer architecture used in ALVINN (Pomerleau, 1989).

Compounding errors Unfortunately, the behavioural cloning paradigm is known to suffer from *compounding errors*: since the future states depend on previous predictions (actions), the assumption made in statistical learning that input variables are *independent and identically distributed (i.i.d.)* does not hold. Therefore, small mistakes will place the system into states that are outside the training data distribution, as illustrated in Figure 2.3, and resulting policies struggle to maintain high performances on long time horizons. This effect was identified and tackled in (S. Ross, G. J. Gordon, and J. A. Bagnell, 2011), who proposed to iteratively request

expert labels (actions) from the states encountered by the current trained policy, rather than from the initial expert distribution. However, this can only be accomplished at the cost of a significant labelling effort. In the context of a Lane Keeping application, Bojarski et al. (2016)

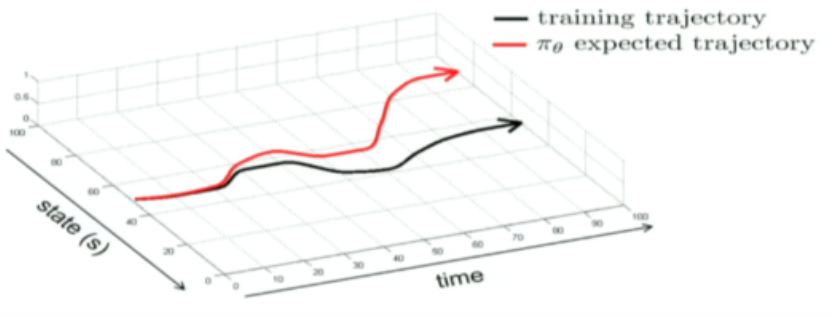


Figure 2.3 – As the agent deviates from the expert trajectories, the errors compound and push the agent further and further from the training distribution.

proposed instead to mitigate this issue during the data collection step by simulating deviations from the expert trajectories by means of two side cameras facing at the edge of the road. The corresponding synthetic expert controls were obtained by adding a constant adjustment term to steer the vehicle back on track. Researchers at Waymo reported using the same technique of synthesising perturbations for their `ChauffeurNet` imitation model (Bansal, Krizhevsky, and Ogale, 2018). The effect of compounding errors can also be delayed by further increasing the prediction performance, *e.g.* by considering temporal dependencies as in (Eraqi, Moustafa, and Honer, 2017; Huazhe Xu et al., 2017). Other techniques than maximum likelihood estimation can be used to train the models, such as Generative Adversarial Imitation Learning (J. Ho and Ermon, 2016), used for highway driving from range measurements in (Kuefler et al., 2017; Bhattacharyya et al., 2018).

Policy conditioning A limitation of imitation learning for autonomous driving is the fact that merely imitating human drivers by sampling likely trajectories is not sufficient: the sampling must also be conditioned on the current short-term destination specified by the Route planner. Codevilla et al. (2018) propose to achieve this by considering several policy heads for three possible behaviours when reaching an intersection: go straight, turn left or turn right. At test time, the appropriate policy is used at each intersection depending on the planned route. Rhinehart, McAllister, Kitani, et al. (2019) and Rhinehart, McAllister, and Levine (2020) present a model-based approach that consists in learning a probabilistic model $q(s)$ of expert trajectories used as a prior, and inferring the maximum a posteriori trajectory given a test-time goal likelihood $p(goal \mid s)$.

2.1.3 Reinforcement Learning

While the MDP framework undoubtedly constitutes a convenient theoretical framework for analysis, it may be too narrow a frame to accommodate the real world. In the sequel, by trying to cast the problem of Autonomous Driving as an MDP, we will identify multiple underlying assumptions that do not hold in practice, and relate them to existing research areas for which researchers proposed variants and solutions.

2.2 States and partial observability

(**a**) *"Area of potential danger"*. A partial observability stemming from sensor occlusion in a turn.

(**b**) *"Area of uncertainty"*. A partial observability stemming from unknown intentions of human agents.

Figure 2.4 – Sources of Partial Observability in Autonomous Driving. Illustrations from (Editions Nationales du Permis de Conduire, 2017).

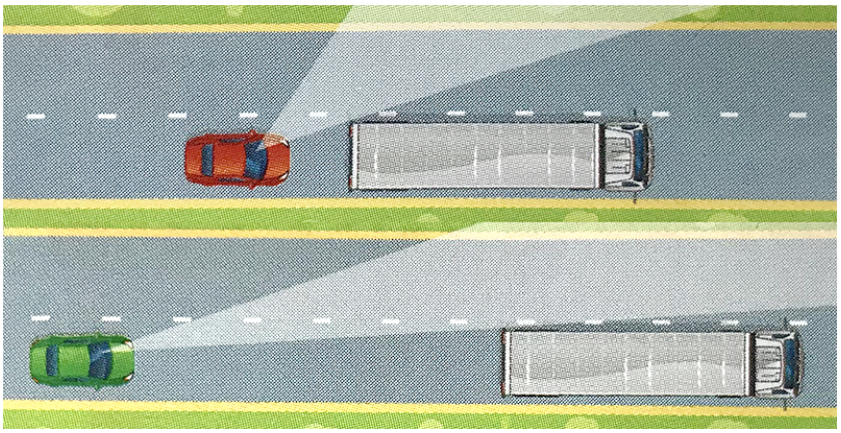


Figure 2.5 – Information-seeking behaviours: the tailgating vehicle (top) should slow down, although it might decrease its immediate rewards, to gain valuable information in return (bottom). Image from (Editions Nationales du Permis de Conduire, 2017).

In order to specify an MDP, the first step consists in defining the state space $\mathcal{S}$, with the underlying assumption that the agent will have access to its current state $s \in \mathcal{S}$. Yet in practice, information about the scene can only be obtained through sensors, which produce typically noisy measurements (Ulbrich and Maurer, 2013; Du et al., 2010; Artuñedo, Villagra, Godoy, and Castillo, 2020). Worse, parts of the state may be missing altogether, as is the case when a scene entity is occluded by an obstacle (*e.g.* Brechtel, Gindele, and Rüdiger Dillmann, 2013; M. Bouton, A. Nakhaei, et al., 2018; Sun et al., 2019), as shown in Figure 2.4a. To account for these difficulties, the concept of a Partially Observable Markov Decision Process (POMDP) was introduced in (Åström, 1965), extending the MDP framework with two additional quantities: an observation space Ω, and an measurement model O such that the observation $o \in \Omega$ is measured at state s' with the conditional probability $O(o \mid s', a)$. At each time step t, a belief $b_t \in \mathcal{M}(\mathcal{S})$ over the state $s_t \in \mathcal{S}$ is updated by performing Bayesian Filtering to compute the posterior state distribution:

$$b_{t+1}(s_{t+1}) = \frac{O(o_t|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} P(s_{t+1} \mid s_t, a_t) b_t(s_t)}{\sum_{s \in \mathcal{S}} O(o_t|s, a_t) \sum_{s_t \in \mathcal{S}} P(s \mid s_t, a_t) b_t(s_t)}.$$

By conditioning the policy $\pi(a_t|b_t)$ on the belief b_t, the POMDP framework allows to optimally balance between information gathering and task completion, as shown in Figure 2.5. However, there is a price to pay: the belief space $\mathcal{M}(\mathcal{S})$ is much larger than the state space $\mathcal{S}$ (*e.g.* continuous of dimension n when $\mathcal{S}$ is finite of size n), which drastically increases the planning complexity. Exact solutions exist when $\mathcal{S}$ is finite (Pineau, G. Gordon, and Thrun, 2003), and approximate ones when it is compact (Porta et al., 2006; Silver and Veness, 2010). Even the belief update is intractable in general, when $\mathcal{S}$ is compact. In the case of linear dynamics and Gaussian measurement noise, the belief update is known as Kalman Filtering (Kalman et al., 1960). It has been applied in (Bry and N. Roy, 2011; M. Bouton, Cosgun, and M. J. Kochenderfer, 2017;

Berg, Patil, and Alterovitz, 2017, *e.g.*), and in the context of quadratic costs –*i.e.* the standard LQG problem– which enables efficient computation of the optimal policy (see *e.g.* W. Xu, Pan, et al., 2014; Berg, Abbee, and Goldberg, 2011). The more complex observation model of sensor occlusions in continuous-space is handled in (Brechtel, Gindele, and Rüdiger Dillmann, 2013; Brechtel, Gindele, and Rudiger Dillmann, 2014; M. Bouton, A. Nakhaei, et al., 2018) where cautious driving policies are learned for crossing intersections; and in (Sun et al., 2019) where the observed behaviour of nearby vehicles is used to infer the presence of potentially occluded pedestrians. Furthermore, the POMDP framework has been used to account for uncertainty in the intentions of other agents, as illustrated in Figure 2.4b. In (Bandyopadhyay et al., 2013), the authors propose a Mixed-Observability framework (MOMDP) in which the other drivers' locations are observed but not their destinations. In (Barbier et al., 2018), the uncertainty lies in whether the other agents intend to give or take the right of way. The value of inferring these drivers intentions has been assessed in (Sunberg, C. J. Ho, and Mykel J. Kochenderfer, 2017), in comparison to MDP baselines where a static or maximum-likelihood behavioural model is assumed instead.

2.3 Actions and temporal abstraction

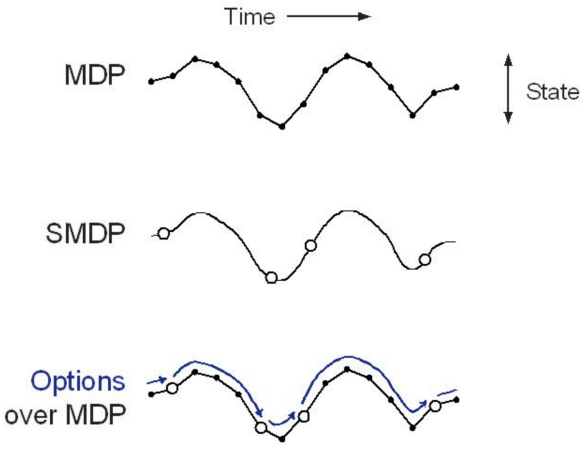


Figure 2.6 – Temporally extended sequences of actions can be used as skills, or *options*, and futher used by a meta-policy to plan over long time horizons (Sutton, Precup, and Satinder Singh, 1999).

The second step of when specifying an MDP is to choose an action space $\mathcal{A}$. Suppose you are taking your first driving lesson, your instructor will be providing you with very detailed instructions about actuation such as *"turn the steering wheel by a full turn"*, *"change gear"*, *"brake smoothly"*, etc. After a few lessons, however, the instructor will switch to more general instructions such as *"change lane to overtake this vehicle"*, *"yield to vehicles on the roundabout before merging"* or *"drive slower in this area"*. And when, having finally got your driver's license,

you start driving in an unfamiliar city, your friend in the passenger seat will merely give you directions: *"follow the signs for the train station, and turn into the third street on the left"*. In short, driving involves reasoning at several time scales, and the corresponding decisions have different granularities. This can hardly be expressed in the MDP framework, where a single action space $\mathcal{A}$ is considered. In particular, relying on shorter actions yields a smaller signal-to-noise ratio, which leads to slow learning when planning over a long time horizon (Shalev-Shwartz, Shammah, and Shashua, 2017). To address this issue, the concept of *temporal abstraction* was introduced: nuclear actions $a \in \mathcal{A}$ can be used to define temporally-extended *skills* $\pi_o : \mathcal{S} \to \mathcal{M}(\mathcal{A})$, also called *options*, which in turn can be used as meta-actions by a high-level *option policy* $\pi(o|s)$. This idea is also referred to as *Hierarchical Reinforcement Learning*, and was first analysed with the *semi*-Markov Decision Process (SMDP) extension (Sutton, Precup, and Satinder Singh, 1999), illustrated in Figure 2.6. In autonomous driving, this hierarchy is typically imposed by the pipeline presented in Chapter 1: Behavioural planning corresponds to the policy over options $\pi(o|s)$, while the low-level options π_o are achieved in the Motion planning layer. However, this requires manually specifying the interfaces at each layer. For instance, in (Barbier et al., 2018) a behavioural policy can only be trained after having defined a finite set of low-level skills, namely *Stop*, *Yield*, and *Pass*. Consequently, using a fixed set of options constrains the comprehensiveness of the set of option-policies, and may prevent recovering optimality if the architecture is not versatile enough. To bypass these limitations, Shalev-Shwartz, Shammah, and Shashua (2016) proposed to ensure sufficient comprehensiveness of the class of available options by generating new options on-the-fly as paths in an option-graph, shown in Figure 2.7. Other works attempt to learn low-level skills jointly with the meta-policy (Bacon, Harb, and Precup, 2017; Vezhnevets et al., 2017; Heess et al., 2016). In (Paxton et al., 2017), low-level skills such as *follow* and *overtake* are trained using Neural Networks and ad-hoc reward functions, while serving as meta-actions for a high-level tree-based planner.

2.4 Rewards and inverse reinforcement learning

Having defined the state and action spaces $\mathcal{S}$ and $\mathcal{A}$, we come to specify which state-action pairs are deemed *desirable*, through the definition of the reward function R. Paradoxically enough, humans know how to drive but not necessarily how to explicit the reasons for their actions, especially in the form of a fixed evaluable objective. A common approach to reward specification is colloquially known as *reward engineering*, in which the reward function is typically parametrised as a linear combination of features $R(s,a) = \sum_i \omega_i \phi_i(s,a)$. For example, such features may include the ego-vehicle speed, its longitudinal distance to a short-term goal, lateral distance to the lane centerline, or the presence of collisions. By handling more and more use-cases, the number of features to consider will rise quickly, some of which contradicting

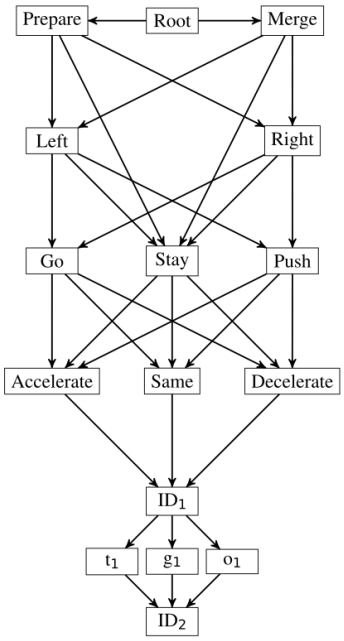


Figure 2.7 – A graph used to generate options (Shalev-Shwartz, Shammah, and Shashua, 2016).

each other. Then, the issue of how to properly choose the weights ω_i remains, and it can be increasingly hard to strike the right trade-off. Besides, this difficulty is further exacerbated by an ambiguity lying in the blurry boundary between the reward function $R(s, a)$ and the value function $V(s)$. For instance, are safety distances desirable *per se*, or only because respecting them means we are less likely to end up in an accident, which is the actual feature of interest here? Likewise, do road traffic regulation rules describe rewarding states or high-value states?

One practical solution to these concerns is to iteratively refine the reward function R until the corresponding optimal policy $\pi^\star$ matches the expected behaviour π_E of human drivers. The careful or well-versed reader will have noticed that this approach is directly opposed to the Reinforcement Learning problem, where the optimal behaviour $\pi^\star$ stems from the reward function R. Accordingly, the aptly named Inverse Reinforcement Learning (IRL) framework aims at finding a reward function that makes the expert behaviour appear uniquely (near)-optimal. At first glance, this problem seems related to Imitation Learning formulation of Section 2.1.2 in its attempt to reproduce expert behaviour. This intuition is supported by the fact that RL $\circ$ IRL is the dual problem of state-action occupancy matching with respect to expert trajectories (J. Ho and Ermon, 2016). Example applications of IRL to Autonomous Driving include the work of Kuderer, Gulati, and Burgard (2015) who learn the trade-off between comfort and efficacy in human lane-change trajectories.

In addition to finding a good candidate reward for the ego-vehicle behaviour, Inverse Reinforcement Learning can also be applied for the purpose of predicting how other agents

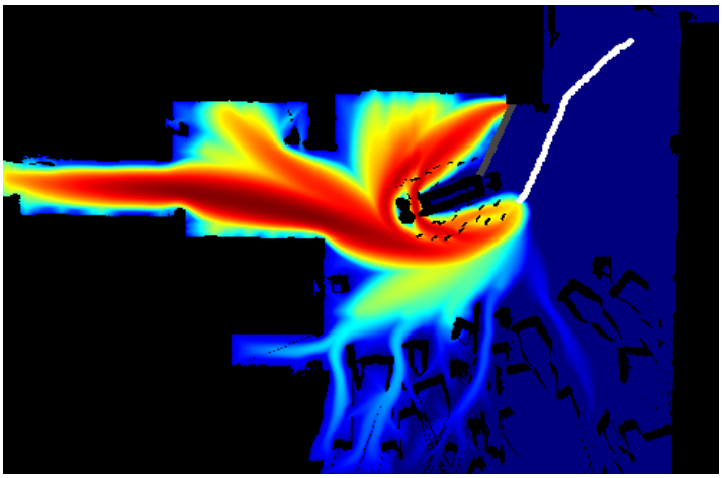


Figure 2.8 – Trajectory prediction for a pedestrian modelled as an optimal planner with a learned cost (Ziebart, Ratliff, et al., 2009).

in the scenes are likely to behave, by modelling them as rational agents trying to maximise an unknown reward function to be learned. In that sense, RL ∘ IRL is a form of model-based Reinforcement Learning. For instance, this approach has been used to model routing preferences of taxi-drivers (Ziebart, Maas, et al., 2008), to predict the future trajectory of pedestrians (Ziebart, Ratliff, et al., 2009) as shown in Figure 2.8, and the behaviour of human drivers at an intersection (Sun et al., 2019) or on a highway (Sadigh et al., 2016).

2.5 Dynamics, offline learning and transfer

We now come to the last element of the Markov Decision Process tuple: the dynamics $P(s' \mid s, a)$. Contrary to the state and actions which are the input and output interfaces of the agent, and to the reward function which is generally chosen by the system designer, there is usually no need to specify the system dynamics. Indeed, one of the most significant assets of Reinforcement Learning is its ability to handle unknown dynamics that are only accessed through interaction. Unfortunately, this assumption that the agent is allowed direct interaction with the true environment is both unacceptable and unrealistic in the context of Autonomous Driving. Indeed, the traditional *learning-by-acting* pipeline requires exploration, which would imply having autonomous vehicles violate the rules of the road and generate accidents, which is obviously out of the question. Besides, most Reinforcement Learning algorithms require a tremendous amount of interaction data to learn an optimal policy, including for MDPs such as Atari games which are arguably less diverse and complex than real-world driving scenes. Not to mention that, perhaps evidently but contrary to simulated games, *the real world can only run in real time*.

This issue can be addressed in two ways. A first solution is to consider the *offline* Reinforcement Learning problem (Levine, Kumar, et al., 2020), in which the agent can no longer interact

with the environment and is instead provided with a *static* dataset $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}$ of interaction data, and must learn the best policy it can with this dataset. These interaction data are collected by an exploration policy π_E, in our case, human driving. However, for the same reasons mentioned in Section 2.1.2, this limited data available induces a loss of optimality: any attempt to improve the policy may steer the agent in regions of the state space $\mathcal{S}$ that are not present in the dataset $\mathcal{D}$, typically accident states, off-road driving, *etc.* in the case of Autonomous Driving. Still, safe policy improvement guarantees can be derived under some conditions, *e.g.* for finite MDPs (Laroche, Trichelair, and Combes, 2019; Nadjahi, Laroche, and Tachet des Combes, 2020). This derivation often requires to constrain how much the learned policy π can differ from the exploration policy π_E, so as to bound the state distribution shift (Kakade and Langford, 2002; Schulman et al., 2015).

Figure 2.9 – Sim-to-real unsupervised transfer (M.-Y. Liu, Breuel, and Kautz, 2017) from synthetic images of the CYNTHIA dataset (Ros et al., 2016) to realistic images of the CITYSCAPES dataset (Cordts et al., 2016).

A second decision is to leverage *simulation*. The vast majority of Reinforcement Learning systems currently running are interacting simulated environments. Indeed, simulation provides many advantages: it is generally much cheaper than real experiments, it can run much faster, be easily parallelised, be reset to an initial state when it fails, and failures are not costly and can be experienced at will. Unfortunately, these benefits come at a price: there is always a *gap* between simulation and reality, an issue known as *model bias*. The problem of adapting a policy learned in one environment to another related one is known as *transfer*, and has been tackled by researchers in different ways. A general strategy is to use simulation as a pre-training and fine-tune to the policy in the target environment (Liang et al., 2019). This method allows to obtain satisfactory initial behaviour and cut down on training time. To reduce the gap between simulation and reality, three approaches can be taken:

(i) make the simulation as *realistic* as possible. For instance, Xinlei Pan and C. Lu (2017) translate the virtual images rendered by a simulator to realistic synthetic images, using

an Image-to-Image Translation Networks similar to that shown in Figure 2.9, so that the observations seen by the agent while training are closer to those it will observe when deployed in the real environment.

(ii) make the simulation more *diverse*, by partially randomising the observations and dynamics, so that the real world appears like yet another realisation of that diversity. This approach is known as *Domain Randomisation* and has been widely applied to robotic manipulation tasks (Tobin et al., 2017; OpenAI et al., 2019), and in the context of Autonomous Driving (Prakash et al., 2019; Pouyanfar et al., 2019).

(iii) make both the simulation and the real-world *abstract*, by mapping them to intermediate observation and action spaces so that the encapsulated policy is not directly exposed to raw perceptual inputs and low-level controls. For instance, semantically segmented images and waypoints are used as observations and actions in (Mueller et al., 2018).

2.6 Optimality criterion and safety

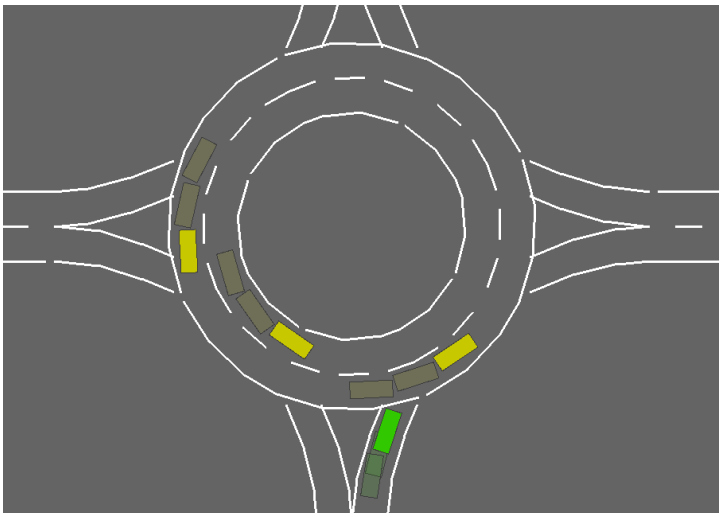


Figure 2.10 – A risky situation: should the vehicle merge into the roundabout?

Once the Markov Decision Process is fully specified, a policy π is said to be optimal if it maximises the *expected* return

$$\max_{\pi} \mathop{\mathbb{E}}_{\pi,P} \left[\sum_{t} \gamma^t R_t \right].$$

But is this an appropriate performance measure? Consider a driving decision –say merging into a roundabout as illustrated in Figure 2.10– that, given the uncertainty on the behaviour of other agents, can lead to a collision with a certain probability δ. When maximising the return in expectation, the agent is allowed to balance the rare accident penalty with the more likely

Decision	Merge		Wait
Outcome	Success	Failure	Boredom
Probability	$1-\delta$	δ	1
Reward	1	$-\frac{1-\delta}{\delta}$	0
$\mathbb{E}[R]$	0		0
$\mathbb{V}(R)$	$\frac{1-\delta}{\delta} \simeq \frac{1}{\delta}$		0
$\min R$	$-\frac{1-\delta}{\delta} \simeq -\frac{1}{\delta}$		0

Table 2.1 – A simple bandit problem associated with Figure 2.10. When trying to merge, an accident happens with probability δ and the agent suffers a high penalty $\frac{1-\delta}{\delta}$. If the agent decides to wait instead, it suffers a small penalty of 0 for the inconvenience. We show the associated expected reward, worst-case reward, and reward variance.

perspective of a successful merge. As observed by Shalev-Shwartz, Shammah, and Shashua (2017), assuming that rewards are normalised in $[0, 1]$ for standard behaviours, for the agent to care about avoiding accidents with probability $1-\delta$ requires the penalty associated with a collision to be in the order of $1/\delta$, as we illustrate in Table 2.1. This leads to a high variance in the observed rewards, which is likely to impede learning. Furthermore, the optimal agent is willing to gamble a collision now and then, if it allows for an increased efficiency the rest of the time. This question of whether and how to weigh human lives with economic benefits brings us straight back to the ethical considerations of Section 1.1.1.

This questioning has lead researchers and practitioners to consider alternative definitions of optimality, tailored for an increased awareness of events of small probability and high consequences. These approaches are grouped under the name of *Safe* Reinforcement Learning, and typically involve giving up some expected performance in favour of the lower tail of the return distribution. They were surveyed by García, Fern, and Fernández (2015), who identified four main paradigms: the first three involve a change of the optimisation criterion, while the fourth one defines a constraint on the exploration process.

2.6.1 Risk-sensitive Criterion

Risk-sensitive methods augment the traditional expected performance criterion with a measure of *variability*. For instance, in the above example of Table 2.1 the two decisions are equivalent in terms of expected return, but `Merge` has a significantly higher variance than `Wait`. A first instance of risk-sensitive objective is the *variance-penalized* criterion (Markowitz, 1959) which takes the form

$$\max_{\pi} \mathop{\mathbb{E}}_{\pi,P}\left[\sum_t \gamma^t R_t\right] - \alpha \mathop{\mathbb{V}}_{\pi,P}\left[\sum_t \gamma^t R_t\right],$$

where the weight $\alpha > 0$ is often tuned manually and allows to trade-off expected performance for consistency (higher α means lower variance in the outcomes).

Many other risk measures have been investigated, such as the Value-at-Risk (VaR), the Conditional Value-at-Risk (CVaR) and percentile performance, in both the Multi-Armed Bandits setting (Torossian, Aurélien Garivier, and Picheny, 2019) and the Reinforcement Learning setting (Moody and Saffell, 2001; Tamar, Di Castro, and Mannor, 2012; L.A. and Ghavamzadeh, 2013; Delage and Mannor, 2010). In their seminal paper, Artzner et al. (1999) identified a set of properties deemed necessary for a risk measure to be *coherent*, and policy gradient methods have been designed to optimise this class of measures (Tamar, Chow, et al., 2015). In the context of Autonomous Driving, the variance-penalised objective has been applied in *e.g.* (Naghshvar, Sadek, and Wiggers, 2018) to highway ramp scenarios with occlusions and limited sensor range.

2.6.2 Worst-case criterion

In model-based Reinforcement Learning, the model parameters are typically estimated from noisy interaction data, which can lead to modelling errors that may have fatal consequences in real, physical systems. To address this issue, robust control community has formulated to optimise the outcome of a policy π with respect to an *ambiguity set* $\mathcal{P}$ of likely dynamics:

$$\max_{\pi} \min_{P \in \mathcal{P}} \mathbb{E}_{\pi, P} \left[\sum_t \gamma^t R_t \right]$$

This problem was first studied by the control community, which focused on the minimax control of the $\mathcal{H}_\infty$-norm (Basar and Bernhard, 1996) and $\mathcal{H}_2$-norm (Berkenkamp and Schoellig, 2015) of linear systems. Minimax-control of quadratic costs –the so-called robust Linear Quadratic (LQ) problem– was later considered in (Abbasi-Yadkori and Szepesvári, 2011; Ibrahimi, Javanmard, and B. V. Roy, 2012; Faradonbeh, Tewari, and Michailidis, 2020; Ouyang, Gagrani, and Jain, 2017; Abeille and Lazaric, 2018; Dean et al., 2019; Dean et al., 2018). The case of finite MDPs with uncertain parameters was studied by Iyengar (2005), Nilim and El Ghaoui (2005), and Wiesemann, Kuhn, and Rustem (2013), who showed that the main results of Dynamic Programming (DP) could be recovered under a *rectangularity* assumption for the structure of uncertainty. Their work was extended to the RL setting in (Tamar, Mannor, and Huan Xu, 2014).

2.6.3 Constrained Criterion

Another approach is to formulate safety as constrained optimisation: the task is to maximise a target function $f(x)$ while satisfying an inequality constraint $g(x) \leq \beta$. For instance, Berkenkamp, Schoellig, and Krause (2016) apply this principle to safe blackbox optimisation using Gaussian Process regression. This idea is extended to a sequential decision-making in the

Constrained Markov Decision Process (CMDP) framework (Altman, 1999; Achiam et al., 2017), a variant of MDPs augmented with a cost function C, and such that the return maximisation objective is subjected to a constraint on the maximum expected cumulative discounted cost:

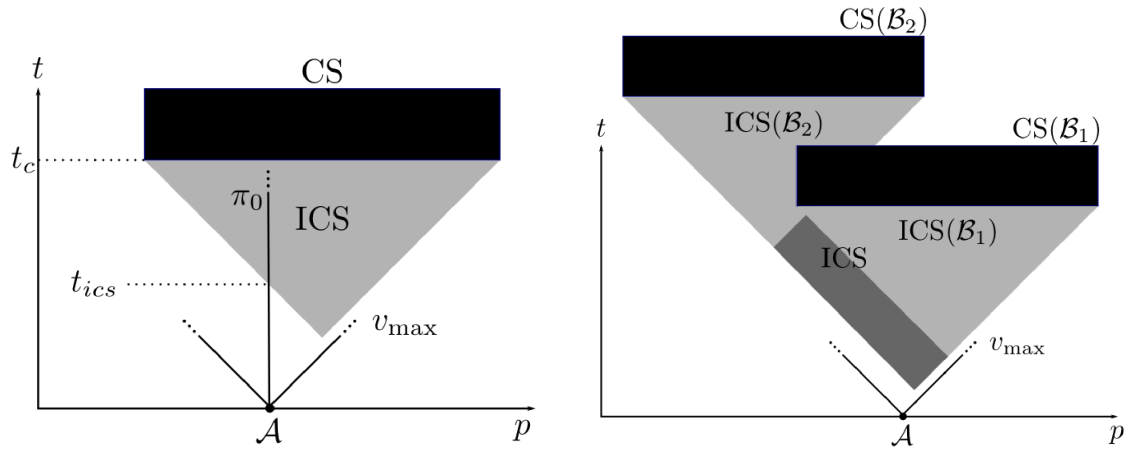
$$\max_{\pi} \mathop{\mathbb{E}}_{\pi,P}\left[\sum_t \gamma^t R_t\right] \text{ subject to } \mathop{\mathbb{E}}_{\pi,P}\left[\sum_t \gamma^t C_t\right] \leq \beta.$$

Extensions have also been proposed to this framework: in (Tessler, Mankowitz, and Mannor, 2019), more general constraints than discounted sums are considered, while in (Geibel and Wysotzki, 2005; Chow, Ghavamzadeh, et al., 2017) the constraint does concern the expectation of the discounted total cost, but rather its percentile or CVaR. CMDPs have been applied in (Maxime Bouton, Karlsson, et al., 2019; Maxime Bouton, Alireza Nakhaei, et al., 2019) to control the probability for the ego-vehicle to reach its goal safely, and in (Le, Voloshin, and Yue, 2019) to enforce two behavioural constraints in car racing simulation: smooth driving (expected number of braking actions) and lane tracking (expected distance to the lane centre).

2.6.4 Safe Exploration

The three paradigms mentioned so far aim to revisit the definition of optimality. However, these adjustments only alter the optimal policy once it is learnt, but they leave the exploration process subject to failures. Consequently, other approaches have been developed to enforce safety constraints at all time. The most common formalisation, robust constraint satisfaction, consists in preventing the agent from visiting any *error state* during exploration –or equivalently to remain within a safe set $\mathbb{X}$– by relying on prior knowledge such as expert demonstrations or an initial feasible policy. According to Thierry Fraichard (2014), this objective is challenging as it requires the ability to *"reason about the future"* as illustrated in Figure 2.11a, and to *"consider obstacles globally and not individually"* as shown in Figure 2.11b.

When the system is fully known, a general solution to this problem is provided by the Hamilton-Jacobi-Isaacs (HJI) reachability equation, in the form of a differential game. This approach is followed in (Leung et al., 2020; Fisac et al., 2019) by decomposing the system into a known deterministic part subject to an adversarial bounded perturbation. In the context of finite MDPs with unknown dynamics, assuming the safe states are defined as the level-set of a safety function $C(s,a) \leq \beta$ whose smoothness is known, Turchetta, Berkenkamp, and Krause (2016) propose an algorithm that explores a maximum *reachable ergodic safe set* without visiting unsafe states. Since solving the HJI equation is intractable in general, researchers have considered more structured special cases, namely linear systems and convex (often polytopic) constraints (Fukushima, Kim, and Sugie, 2007; Adetola and Guay, 2008; Aswani et al., 2013; Lorenzen, Allgöwer, and Cannon, 2017; Köhler et al., 2019; X. Lu and Cannon, 2019).



(**a**) A minimum horizon $T > t_c - t_{ics}$ is required

(**b**) Obstacles must be considered *globally*.

Figure 2.11 – Inevitable Collision States (ICS), images from (Thierry Fraichard, 2014).

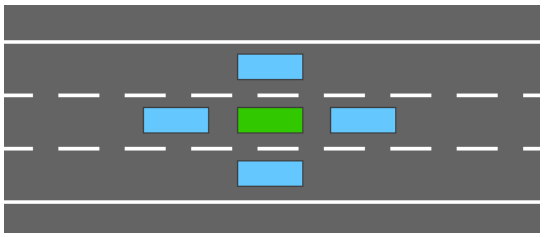


Figure 2.12 – An ordinary highway-driving situation, but prone to accidents under adversarial behaviours.

Yet, it is questionable whether the property of robust constraint satisfaction under adversarial disturbances is relevant for an Autonomous Driving application. Indeed, this requirement might be too restrictive since many nominal states are prone to errors under adversarial behaviours, as shown in the example of Figure 2.12. A strategy of avoiding this region of the state-space altogether would be overly cautious and unacceptable. To cope with that issue, weaker models of safety have been proposed. Thus, rather than avoiding all collisions, *Passive* motion safety (Maĉek et al., 2008; Sara Bouraine, Thierry Fraichard, and Salhi, 2012; S. Bouraine et al., 2014) only requires the robot to be motionless whenever a collision could possibly occur. To account for limited dynamic capabilities of other agents, the stronger *Passive-friendly* motion safety model was introduced (Mitsch et al., 2017), ensuring not only that the ego-vehicle safely stops itself, but also allows sufficient space for other vehicles to stop before a collision occurs. Note that this is equivalent to the original motion safety model with a restricted set of adversarial behaviours for other agents, requiring them to brake and avoid collisions when possible. Finally, Shalev-Shwartz, Shammah, and Shashua (2017) defines a notion of *Responsibility Sensitive Safety* specific to Autonomous Driving, which formalizes *"an interpretation of* Duty of Care *from Tort law"*. This interpretation is summarised by five main rules tailored for a

specific set of use-cases (including several road geometries, right of way rules, pedestrians and occlusions) and implemented as a set of dynamic geometrical constraints.

Chapter 3

Problem Statement

Notre héritage n'est précédé d'aucun testament.
On ne se bat bien que pour les causes qu'on modèle soi-même
et avec lesquelles on se brûle en s'identifiant.

René Char, *Feuillets d'Hypnos* (62–63).

Having discussed at length the range of Autonomous Driving modelling perspectives in Chapter 2, we now formalise the specific problem that we are going to consider in this thesis. This chapter attempts to cast Behavioural Planning as a Markov Decision Process, by specifying each element of a $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ tuple suitable for a set of tactical decision-making tasks.

Contents

3.1 Perceived states

As discussed in Sections 1.1.2 and 2.2, information about the world is typically obtained from noisy sensory measurements. The Perception module is responsible for recognising and tracking the signal from the noise, so as to provide a high-level probabilistic description of the scene. In particular,

(i) a *Mapping* layer reconstructs the geometry of the road network and its associated signage, including stop signs and traffic lights;

(ii) a *Localisation* layer recovers the position, velocity and heading of the ego-vehicle;

(iii) a *Scene understanding* layer returns the position, velocity and geometry of any vehicle or obstacle nearby.

Since we focus on the Decision module, we will take a simplifying assumption and ignore all aspects related to Perception. Namely, we take the liberty of assuming noise-free access to every feature of the driving scene that we will deem relevant.

Vehicles As mentioned in Chapter 1, the main challenge of Behavioural Planning is to interact with other vehicles. Therefore, the state should include a description of every vehicle nearby. In addition to the ego-vehicle, indexed by 0, the scene contains a number N_v of other vehicles indexed by the range $[1, N_v]$. Any vehicle of index $i \in [0, N_v]$ is represented by

(i) its position $(p_i^x, p_i^y) \in \mathbb{R}^2$,

(ii) its forward speed $v_i \in \mathbb{R}$,

(iii) its heading $\psi_i \in \mathbb{R}$.

The resulting joint state is the traffic description:

$$s = \begin{bmatrix} p_0^x & p_0^y & v_0 & \psi_0 \\ \vdots & \vdots & \vdots & \vdots \\ p_{N_v}^x & p_{N_v}^y & v_{N_v} & \psi_{N_v} \end{bmatrix} \in \mathcal{S} \triangleq \mathbb{R}^{(N_v+1)\times 4}. \tag{3.1}$$

We can make a few observations: first, the state space is continuous, which means we will have to resort to function approximation to represent either the policy π, the value function Q or the dynamics P. Second, it has a variable size, since it depends on the number of vehicles nearby, which the function approximation scheme will have to accommodate. Its dimensionality should be in the order of fifty at most, for a dozen observed vehicles.

Roads We also assume knowledge of the road network, comprising:

(i) a graph description of the network topology, where the nodes represent intersections and the edges represent road segments;

(ii) the geometry of every lane L in the network (every edge), described by its centre-line parametric curve $s \to (p_L^x(s), p_L^y(s)) \in \mathbb{R}^2$, and heading $\psi_L : s \to \tan^{-1}\left(\frac{\mathrm{d}p_L^y}{\mathrm{d}s}(s)/\frac{\mathrm{d}p_L^x}{\mathrm{d}s}(s)\right) \in \mathbb{R}$ (tangent to the curve) where $s \in [0, l_L]$ is the curvilinear abscissa and l_L is the length of the lane L.

However, we do not include this information as part of the state but rather of the system dynamics, described later. Consequently, model-free algorithms will learn policies tailored for the particular scene seen during training, and will not be able to adapt to different scenes, unless the state space is augmented to include road features. Conversely, model-based algorithms can leverage road information in their dynamics models and thus generalise to unseen scenes.

3.2 Behavioural decisions

We follow the hierarchical architecture of the Decision module discussed in Sections 1.1.2 and 2.3. Since we focus on Behavioural Planning specifically, we assume the availability[1] of

(i) a Route Planning layer, that automatically selects the next road segment to follow at each intersection, *e.g.* the proper exit on a highway, or the right direction at an intersection;

(ii) a Motion Planning and Control layer, that controls the vehicle by way of low-level throttle and steering actuators to reach any desired position and speed in the selected road segment.

Thus, the purpose of the Behavioural Planning layer is to specify short-term instructions for the Motion Planning layer, in the form of a lane to follow and a speed to adapt. The produced trajectory will always conform to the planned route, but the Behavioural Planner is in charge of *e.g.* deciding when to merge on a highway, negotiating right of way at an intersection, overtaking vehicles, *etc.*. To that end, we specify the following space of *meta-actions*:

$$\mathcal{A} \triangleq \left\{ \begin{array}{c} \text{change to the left lane, change to the right lane,} \\ \text{drive faster, drive slower, maintain speed and lane} \end{array} \right\} \tag{3.2}$$

Meta-actions are rather slow to affect the state of the vehicle and are thus executed at a low frequency of 1 Hz. We will consider a behavioural planning horizon of a dozen seconds (enough to *e.g.* take/give way to a vehicle and merge in traffic), which corresponds to a dozen of decision points. We describe next how these meta-actions influence the evolution of the state $s \in \mathcal{S}$.

[1] These two modules are described as part of the dynamics.

3.3 Traffic dynamics

This section describes how the behavioural decisions influence the evolution of the perceived states, under their above definitions, through the dynamics distribution $P(s' \mid s, a)$. As explained in Chapter 1, it is crucial that the simulated vehicles in the scene are able to *react* to the actions of the ego-vehicle, so that interaction patterns can be learnt.

3.3.1 Kinematics

We represent the non-holonomic motion capabilities of every vehicle $i \in [0, N_v]$ in the scene by the Kinematic Bicycle Model (see *e.g.* Polack, Altché, and D'Andréa-Novel, 2017):

$$\begin{aligned} \dot{p}_i^x &= v_i \cos(\psi_i + \beta_i), \\ \dot{p}_i^y &= v_i \sin(\psi_i + \beta_i), \\ \dot{\psi}_i &= \frac{v_i}{l} \sin(\beta_i), \end{aligned} \tag{3.3}$$

where l is the vehicle half-length, $\dot{v}_i$ is the throttle command and β_i is the slip angle at the centre of gravity, used as a steering command.

3.3.2 Motion planning and control

We equip the ego-vehicle –and also other vehicles $i \in [0, N_v]$ in the scene– with a capability to execute the meta actions $\mathcal{A}$. This requires the ability to follow a lane L_i, described by the road information mentioned above through its lateral position $p_{L_i}^y$ and heading ψ_{L_i} . To that end, vehicles follow a cascade controller of lateral position and heading in the form

$$\begin{aligned} \dot{\psi}_i &= K_i^\psi \left(\psi_{L_i} + \sin^{-1}\left(\frac{\tilde{v}_{i,y}}{v_i} \right) - \psi_i \right), \\ \tilde{v}_{i,y} &= K_i^y (p_{L_i}^y - p_i^y), \end{aligned} \tag{3.4}$$

where $K_i^y \in \mathbb{R}$ and $K_i^\psi \in \mathbb{R}$ are control gains. Note that the corresponding steering command β_i can be obtained from (3.4) as:

$$\beta_i = \sin^{-1}\left(\frac{l}{v_i} \dot{\psi}_i \right).$$

Furthermore, the ego-vehicle needs to be able to control its speed as per the meta actions $\mathcal{A}$. To that end, we use a linear longitudinal controller

$$\dot{v}_0 = K_0^v (v_r - v_0),$$

where $v_r \in \mathbb{R}$ is the reference speed, incremented by $\pm 5\,\mathrm{m/s}$ by the *drive faster* and *drive slower* meta-actions, and $K_0^v \in \mathbb{R}$ is a control gain.

3.3.3 Behavioural models

Other simulated vehicles follow simple behavioural models from the traffic simulation literature, that dictate how they accelerate and steer on the road.

Longitudinal behaviour The acceleration command a_i of a vehicle $i \in [1, N_v]$ is controlled directly by the Intelligent Driver Model (IDM) from (Treiber, Hennecke, and Helbing, 2000):

$$\begin{aligned} \dot{v}_i &= a_i \left[1 - \left(\frac{v}{v_i^0} \right)^\delta - \left(\frac{d_i^\star}{d_i} \right)^2 \right], \\ \text{where} \quad d_i^\star &= d_i^0 + T_i v_i + \frac{v_i \Delta v_i}{2\sqrt{a_i^+ b_i}}, \end{aligned} \tag{3.5}$$

v_i is the vehicle velocity, d_i is the distance to its front vehicle. The dynamics of vehicle $i \in [1, N_v]$ are thus parametrised by the desired velocity v_i^0 , the time gap T_i, the jam distance d_i^0, the maximum acceleration a_i and deceleration b_i, and the velocity exponent δ.

Lateral behaviour The discrete lane change decisions are given by the Minimizing Overall Braking Induced by Lane change (MOBIL) model from (Kesting, Treiber, and Helbing, 2007). According to this model, a vehicle $i \in [1, N_v]$ decides to change lane when

(i) it is *safe* to cut-in:

$$\tilde{a}_n \geq -b_{\text{safe}};$$

(ii) there is an *incentive*, for the ego-vehicle and possibly its followers:

$$\underbrace{\tilde{a}_c - a_c}_{\text{the vehicle}} + p \left(\underbrace{\tilde{a}_n - a_n}_{\text{new follower}} + \underbrace{\tilde{a}_o - a_o}_{\text{old follower}} \right) \geq \Delta a_{\text{th}};$$

where c is the centre vehicle, o is its old follower *before* the lane change, and n is its new follower *after* the lane change; a and $\tilde{a}$ are the predicted accelerations of the vehicles *before* and *after* the lane change respectively; p is a politeness coefficient, Δa_{th} is the acceleration gain required to trigger a lane change; and b_{safe} is the maximum braking imposed to a vehicle during a cut-in.

A lane change decision modifies the target lane L_i followed by vehicle i on its current road segment. The actual trajectory planning and steering control to track this lane is then performed by the lateral controller of (3.4).

3.3.4 Route planning

So far, we explained how both the ego-vehicle ($i = 0$) and the other simulated vehicles ($i \in [1, N_v]$) behave on a multi-lanes road segment, through their Behavioural and Control layers. The Route Planning layer is finally responsible for selecting the sequence of road segments leading to a destination, sampled randomly at initialisation. To that end, the Route Planning performs a Breadth-First Search in the graph description of the road network mentioned above, and returns a shortest path of road segments from the initial position to the destination.

In the end, we frame the state space $\mathcal{S}$ as fully observable but subjected to uncertain transition dynamics $P(s' \mid s, a)$, which are parametrised by several unobserved variables including the destinations of agents in the scenes and the parameters of their Behavioural and Control layers.

3.4 Rewards

As discussed in Section 2.4, choosing an appropriate reward function that yields realistic optimal driving behaviour is a challenging problem, that we do not address in this thesis. In particular, we do not wish to specify every single aspect of the expected driving behaviour inside the reward function, such as keeping a safe distance to the front vehicle. Instead, we would rather only specify a reward function as simple and straightforward as possible, and focus solely on the difficulties related to safe decision-making under uncertainty, in the hope to see adequate behaviour emerge from learning. In this perspective, keeping a safe distance would be optimal not for being directly rewarded but for robustness against the uncertain behaviour of the leading vehicle, which could brake at any time.

Thus, we focus on only two features: a vehicle should (i) progress quickly on the road; (ii) avoid collisions.

Since the MDP formalism requires rewards to be bounded, by convention we normalise them in the $[0, 1]$ range. Note that we forbid negative rewards, since they may incentivise the agent to prefer terminating an episode early (by causing a collision) rather than risking suffering a negative return if no satisfying trajectory can be found.

Thus, unless otherwise stated, the reward function R is chosen as follows:

$$R(s, a) = \begin{cases} 1 & \text{if the ego-vehicle is at full speed;} \\ 0 & \text{if the ego-vehicle has collided with another vehicle;} \\ 0.5 & \text{else.} \end{cases} \tag{3.6}$$

A more realistic reward function may include comfort terms, such as penalising high acceleration or jerk, and lane changes manoeuvres, but we do not consider them for simplicity.

This reward function is *dense*[2], since the maximum reward can easily be obtained from any state by accelerating, which should guide exploration to efficient driving styles. However, it is also *non-convex*, since *e.g.* the collision penalty is incurred at the locations of any two obstacles but not in-between. It is even *non-smooth*, given that it is discontinuous at collision states.

3.5 Implementation

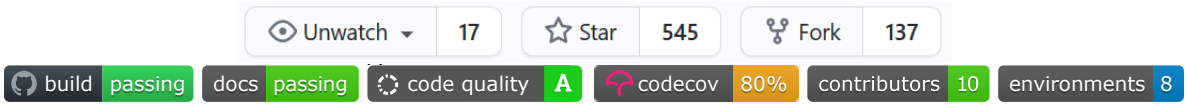


Figure 3.1 – HIGHWAY-ENV repository status (on 27/11/2020).

I created the HIGHWAY-ENV environment, a minimalist driving simulator tailored for behavioural planning tasks following the MDP formalisation presented in this chapter. It is written in Python and published online under an open-source license (Leurent, 2018). An extensive documentation is also available. We discuss at length the features and architecture of this software in Chapter A. We mention that in addition to our own works, several students and researchers already make use of this environment, as shown in Figure 3.1 and discussed in Section A.2. The source code for the agents, allowing to reproduce every numerical experiment presented throughout this manuscript, is also available in the RL-AGENTS repository.

[2]Rewards are said to be *dense* when they are a rich signal obtained at (nearly) every step of decisions, which helps quickly shaping the behaviour and guiding exploration. In contrast, *sparse* rewards are obtained for only a few goal states which are seldom reached (*e.g.* only at the exit of a maze, or the end of a board game), which makes exploration much harder and requires assigning *credit* to the action(s) responsible for a win/loss

Part II

Model-free

Agir en primitif…

René Char, *Feuillets d'Hypnos* (72).

Chapter 4

Considering Social Interactions

O what a strange parcel of creatures are we,
Scarce ever to quarrel, or even agree;
.....
Like social companions we never fall out,
Nor ever care what one another's about;

Elizabeth Hands, *On An Unsociable Family*.

Having detailed the MDP model in Part I, we now study in Part II how model-free Reinforcement Learning algorithms can learn an optimal behavioural planning policy. In this chapter, we focus on the design of *sample-efficient* learning architectures, tailored for dense traffic situations. Such architectures should deal with a varying number of nearby vehicles, be invariant to the ordering chosen to describe them, while staying accurate and compact. We observe that the two most popular representations in the literature do not fit these criteria, and perform badly on a complex negotiation task. We propose an attention-based architecture that satisfies all these properties and explicitly accounts for interactions between the traffic participants. We empirically show that this architecture enjoys significant performance gains, and is able to capture interactions patterns that can be visualised and qualitatively interpreted.[1]

Contents

[1]This chapter is based on a preprint (Leurent and Mercat, 2019) presented at the *Machine Learning for Autonomous Driving workshop* at the NeurIPS 2019 conference. It is a collaboration with my friend and colleague Jean Mercat, who pursued this idea in further work on trajectory forecasting, which was published at the *2020 International Conference on Robotics and Automation* (Mercat et al., 2020) and won two international competitions.

4.1 Motivation

Value-based Reinforcement Learning algorithms such as Q-Learning (Watkins and Dayan, 1992) and its variants rely on estimating the optimal state-action value function $Q^\star$. Since the state space $\mathcal{S}$ chosen in Chapter 3 is continuous, we must resort to function approximation. Thus, independently of how $\mathcal{S}$ was defined, we now have to specify how a state $s \in \mathcal{S}$ will be *represented* as an input to parametrised model Q_θ. The choice of both the model class and state representation will strongly influence the system performances. In particular, we claim that the two most widely used representations both suffer from different drawbacks: on the one hand, the *list of features* representation is compact and accurate but has a varying-size and depends on the choice of ordering. On the other hand, the *spatial grid* representation addresses these concerns but in return suffers from an accuracy-size trade-off.

Our contributions are the following: first, we propose an attention-based architecture for decision-making involving social interactions. This architecture allows to satisfy the variable-size and permutation invariance requirements even when using a *list of features* representation. It also naturally accounts for interactions between the ego-vehicle and any other traffic participant. Second, we evaluate our model on a challenging intersection-crossing task involving up to 15 vehicles perceived simultaneously. We show that our proposed method provides significant quantitative improvements and that it enables us to capture interaction patterns in a visually interpretable way.

4.1.1 Background

We start by giving some background on standard model-free learning algorithms –with a focus on value-based methods–, on usual state representations used for behavioural planning, and on attention mechanisms for Neural Networks (NNs).

Value-based deep Reinforcement Learning

Recall from Definition 1.3 that $Q^\star$ can be computed from an optimal policy $\pi^\star$ since $Q^\star = Q^{\pi^\star}$. However, the converse that $\pi^\star$ can be obtained from knowing $Q^\star$ is also true since

Proposition 4.1 (Optimality of the greedy policy, Bellman, 2010)**.** *A greedy policy defined as*

$$\forall s \in \mathcal{S}, \pi^\star(\cdot \mid s) = \delta_{a^\star}, \text{ where } a^\star \in \arg\max_a Q^\star(s, a),$$

and $\delta_x = \delta(x - \cdot)$ denotes the Dirac distribution in x, is optimal.

Finding an optimal policy thus reduces to computing the optimal value function $Q^\star$. Fortunately,

Theorem 4.2 (Bellman Optimality Equation, Bellman, 2010)**.** *The optimal action-value function $Q^\star$ satisfies the Bellman Optimality Equation:*

$$Q^\star(s,a) = (\mathcal{T}Q^\star)(s,a) \triangleq \mathop{\mathbb{E}}_{s'\sim P(s'|s,a)} \max_{a'\in\mathcal{A}} \left[R(s,a) + \gamma Q^\star(s',a')\right].$$

Moreover, $\mathcal{T}$ is a γ-contraction for the $\|\cdot\|_\infty$ norm:

$$\forall Q_1, Q_2 \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}},\ \|\mathcal{T}Q_1 - \mathcal{T}Q_2\|_\infty \leq \gamma\|Q_1 - Q_2\|_\infty.$$

Thus, since $Q^\star$ is a fixed-point of a contracting operator, it can be computed by iteratively applying $\mathcal{T}$ in a fixed-point iteration fashion. The *Q-learning* algorithm (Watkins and Dayan, 1992) follows this procedure by applying a sampling version $\mathcal{T}$ to a batch of collected experience. When dealing with a continuous state space S, we need to employ function approximation in order to generalise to nearby states. The Deep Q-Network (DQN) algorithm (Mnih et al., 2015) implements this idea by using a neural network model to represent the action-value function Q.

Common traffic state representations

In order to apply a reinforcement learning algorithm such as DQN to an autonomous driving problem, a state space S must first be chosen, that is, a representation of the scene. The state should at least contain a description of every nearby vehicle, when social interactions are relevant to the decision. We recall our definition (3.1) of $\mathcal{S}$ from Chapter 3, in which a vehicle driving on a road is described by it's continuous position, heading and velocity, and the joint state of a road traffic with one ego-vehicle denoted s_0 and N_v other vehicles can be described by a list of individual vehicle states:

$$s = \begin{bmatrix} p_0^x & p_0^y & v_0 & \psi_0 \\ \vdots & \vdots & \vdots & \vdots \\ p_{N_v}^x & p_{N_v}^y & v_{N_v} & \psi_{N_v} \end{bmatrix} \in \mathcal{S} \triangleq \mathbb{R}^{(N_v+1)\times 4}.$$

This description was appropriate to simply describe the system dynamics. However, it has several drawbacks when used for function approximation: because of its 2π-periodicity, the heading ψ_i is either clipped to $(-\pi, \pi]$ which causes a discontinuity at $\pm\pi$, or unclipped which causes several inputs to correspond to the same state. Likewise, the forward velocity v_i needs

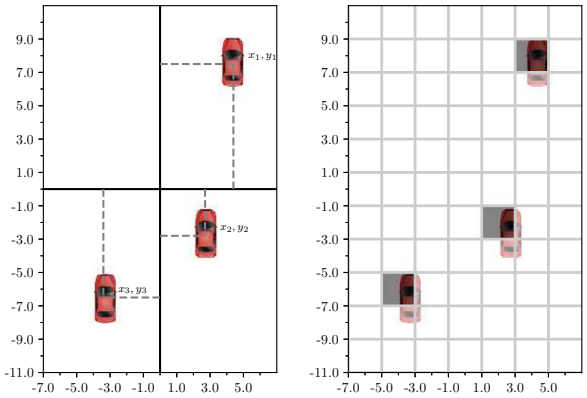


Figure 4.1 – The *list of features* (left) and *spatial grid* (right) representations

to be combined with the heading ψ_i and projected to inform the future positions p_i^x, p_i^y of the vehicle i. Consequently, we slightly modify the features describing the vehicle states as

$$s = (s_i)_{i\in[0,N_v]} \qquad \text{where} \qquad s_i = \begin{bmatrix} p_i^x & p_i^y & \dot{p}_i^x & \dot{p}_i^y & \cos\psi_i & \sin\psi_i \end{bmatrix} \tag{4.1}$$

This representation, that we call *list of features*, is illustrated in Figure 4.1 (left) and was used for instance in (Bai et al., 2015; Gindele, Brechtel, and Rudiger Dillmann, 2015; Song, Xiong, and H. Chen, 2016; Sunberg, C. J. Ho, and Mykel J. Kochenderfer, 2017; Paxton et al., 2017; Galceran et al., 2017; Y. F. Chen et al., 2017).

This encoding is efficient in the sense that it uses the smallest quantity of information necessary to represent the scene. However, it lacks two important properties. First, its size varies with the number of vehicles which can be problematic for the sake of function approximation which often expects constant-sized inputs. Second, we expect a driving policy π to be *permutation invariant*, *i.e.* not to be dependent on the order in which other traffic participants are listed. Ideally, this property should be enforced and not approximated by relying on the coverage of the $N_v!$ possible permutations τ of any given traffic state in the dataset. Formally, we require that

$$\pi(\cdot|(s_0, s_1, \dots, s_{N_v})) = \pi(\cdot|(s_0, s_{\tau(1)}, \dots, s_{\tau(N_v)})), \qquad \forall \tau \in \mathfrak{S}_{N_v}, \tag{4.2}$$

where $\mathfrak{S}_{N_v}$ is the *symmetric* group of permutations of the integer range $[1, N_v]$. A popular way to address these limitations is to use a *spatial grid* representation. Instead of explicitly representing spatial information as variables x, y along with other features f directly inside a state $\{s_i = (p_i^x, p_i^y, f_i)\}_{i\in[0,N]}$ indexed on the vehicles, they are instead represented implicitly through the layout of several feature variables f_{ij} organised in a tensor structure, where the (i, j) indexes refer to a quantisation of the 2D-space. This representation is illustrated in Figure 4.1 (right). Note that the size of this tensor is related to the area covered divided by the quantisation step, which reflects a trade-off between accuracy and dimensionality. In an occupancy grid,

the f features contains presence information (0-1) and additional channels such as velocity and heading, as in (*e.g.* Isele et al., 2018; Lex Fridman, Terwilliger, and Jenik, 2018; Bansal, Krizhevsky, and Ogale, 2018; Rehder, Wirth, et al., 2018). Another example is the use of top-view RGB images (*e.g.* J. Bagnell et al., 2010; Rehder, Quehl, and Stiller, 2017; Rehder, Wirth, et al., 2018; J. Liu et al., 2018).

This permutation invariance property (4.2) can also be implemented within the architecture of the policy π. A general technique to achieve this is to treat each entity similarly in the early stages – *e.g.* through weight sharing – before reducing them with a projection operator that is itself invariant to permutations, for instance, a max-pooling as in (Y. F. Chen et al., 2017; Hoel, Wolff, and Laine, 2018) or an average as in (Qi et al., 2017). A particular instance of this idea is attention mechanisms.

Attention mechanisms

The attention architecture was introduced to enable NNs to discover interdependencies within a variable number of inputs. It has been used for pedestrian trajectory forecasting in (Vemula, Muelling, and Oh, 2018) with spatiotemporal graphs and in (Sadeghian, Kosaraju, et al., 2019) with spatial and social attention using a generative Neural Network. Sadeghian, Legros, et al. (2018) use attention over top-view road scene images for car trajectory forecasting. Multi-head attention mechanism has been developed by Vaswani et al. (2017) for sentence translation. In (Messaoud et al., 2019) a mechanism called non-local multi-head attention is developed. However, this is a spatial attention that does not allow vehicle-to-vehicle attention. In the present chapter, we use a multi-head social attention mechanism to capture vehicle-to-ego dependencies and build varying input size and permutation invariance into the policy model.

4.2 A social attention architecture

Out of a complex scene description, the model should be able to filter information and consider only what is relevant for the decision. In other words, the agent should *pay attention* to vehicles that are close or conflict with the planned route.

The proposed architecture is presented in Figure 4.2. We use it to represent the Q-function that will be optimised by the DQN algorithm. It is composed of a first linear encoding layer whose weights are shared between all vehicles. At that point, the embeddings only contain individual features of size d_x. They are then fed to an ego-attention layer, composed of several heads stacked together. The *ego* prefix highlights that it similar to a multi-head self-attention layer (Vaswani et al., 2017) but with only a single output corresponding to the ego-vehicle. Such an ego-attention head is illustrated in Figure 4.3 and works in the following way: in order

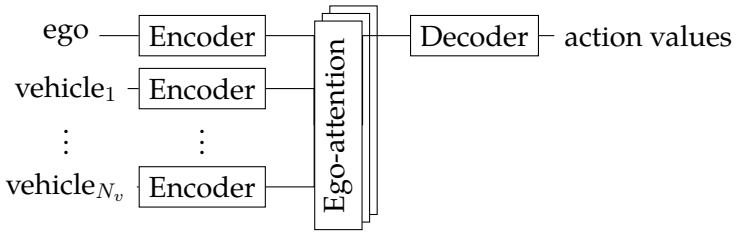


Figure 4.2 – Block diagram of our model architecture. It is composed of several identical linear encoders, a stack of ego-attention heads, and a linear decoder.

to select a subset of vehicles depending on the context, the ego-vehicle first emits a single query $Q = [q_0] \in \mathbb{R}^{1\times d_k}$, computed with a linear projection $L_q \in \mathbb{R}^{d_x\times d_k}$ of its embedding. This query is then compared to a set of keys $K = [k_0, \dots, k_{N_v}] \in \mathbb{R}^{(N_v+1)\times d_k}$ containing descriptive features k_i for each vehicle, again computed with a shared linear projection $L_k \in \mathbb{R}^{d_x\times d_k}$. The similarity between the query q_0 and any key k_i is assessed by their dot product $q_0 k_i^T$. These similarities are then scaled by the inverse-square-root-dimension $1/\sqrt{d_k}$[2] and normalised with a softmax function σ across vehicles. We obtain a stochastic matrix called the *attention matrix*, which is finally used to gather a set of output value $V = [v_0, \dots, v_{N_v}]$, where each value v_i is a feature computed with a shared linear projection $L_v \in \mathbb{R}^{d_x\times d_v}$. Overall, the attention computation for each head can be written as

$$\text{output} = \underbrace{\sigma\left(\frac{QK^T}{\sqrt{d_k}}\right)}_{\text{attention matrix}} V. \tag{4.3}$$

The outputs from all heads are finally combined with a linear layer, and the resulting tensor is then added to the ego encoding as in residual networks. We can easily see that this process is permutation invariant: indeed, a permutation τ will change the order of the rows in keys K and values V in (4.3) but will keep their correspondence. The final result is a dot product of values and key-similarities, which is independent of the ordering.

4.3 Experiments

Videos and source code of the experiments below are available[3].

Environment In this experiment, we use the HIGHWAY-ENV environment (Leurent, 2018) presented in Chapter A. We consider a task where vehicle-to-vehicle interaction plays a significant part: crossing a four-way intersection. The scene – composed of two roads crossing perpendicularly – is populated with several traffic participants initialised with random positions, velocities,

[2] This scaling is due to the fact that the dot-product of two independent random vectors with mean 0, variance 1, and dimension d_k, is a random variable with mean 0 and variance d_k

[3] https://eleurent.github.io/social-attention/

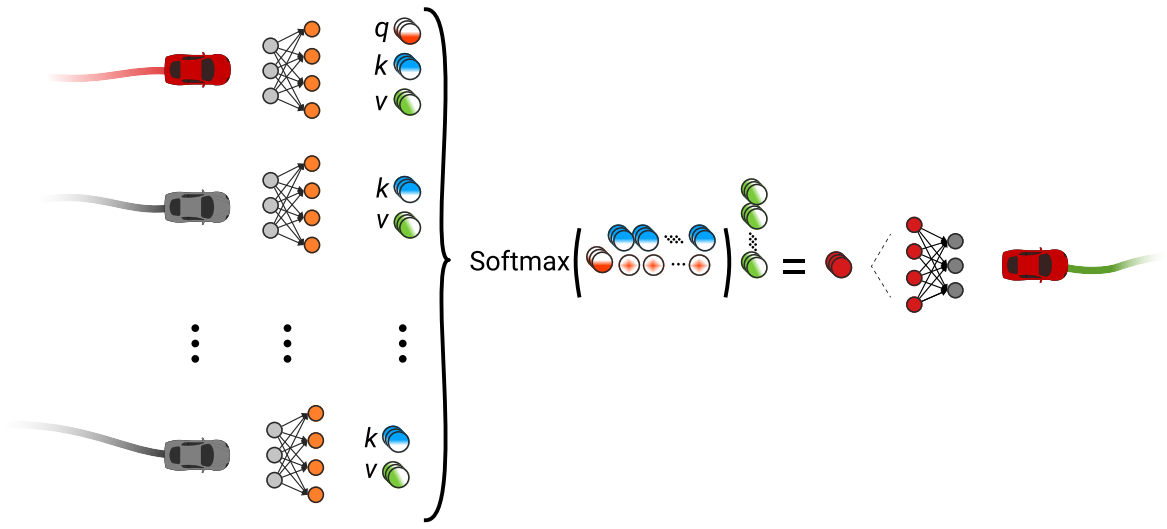


Figure 4.3 – Architecture of an ego-attention head. After received the encoded vehicle states, the ego-query q, all keys k and all values v are produced by three linear projections. Then, the attention matrix is computed by matching the keys K to the ego query q_0, and the corresponding values V are retrieved. The resulting embedding is finally forwarded to a decoder to obtain the predicted Q-values as an output.

and destinations. As described in Chapter 3, these vehicles are simulated with the Kinematic Bicycle Model, their lateral control is achieved by a low-level steering controller tracking a target route, and their longitudinal behaviour follows the IDM model (Treiber, Hennecke, and Helbing, 2000). However, this model only considers same-lane interactions and special care was required to prevent lateral collisions at the intersection. To that end, I implemented the following simplistic behaviour: each vehicle predicts the future positions of its neighbours over a three-seconds horizon by using a constant velocity model. When a collision with a neighbour is predicted, the yielding vehicle is determined based on road priorities and brakes until the collision prediction ceases.

In this context, the agent must drive a vehicle by controlling its speed chosen from a finite set of actions $\mathcal{A} \triangleq \{\text{drive faster}, \text{drive slower}, \text{maintain speed}\}$. Note that we removed the lane change actions from the general definition (3.2) of Chapter 3 since the roads are all single-lane in this example. The lateral control is performed automatically by a low-level controller, such that the problem complexity is focused on the high-level interactions with other vehicles, namely the decision to either give or take way. The reward function R is defined as in (3.6).

Agents We evaluate three different agents, whose characteristics are summarised in Table 4.1.

- `FCN/List`: a *list of features* state representation is used, as described in Section 4.1.1. The model is a simple Fully-Connected Network (FCN). Because this architecture requires a fixed-size input, we use zero-padding to fill the input tensor up to a maximum number

Table 4.1 – Characteristics of the agents

Architecture	`FCN/List`	`CNN/Grid`	`Ego-Attention`
Input sizes	$[15, 7]$	$[32, 32, 7]$	$[\cdot, 7]$
Layers sizes	$[128, 128]$	Convolutional layers: 3 Kernel Size: 2 Stride: 2 Head: $[20]$	Encoder: $[64, 64]$ Attention: 2 heads $d_k = 32$ Decoder: $[64, 64]$
Number of parameters	3.0×10^4	3.2×10^4	3.4×10^4
Variable input size	No	No	Yes
Permutation invariant	No	Yes	Yes

$N = 14$ of observed vehicles, and add an additional *presence* feature to the coordinates described in (4.1) so as to identify active rows.

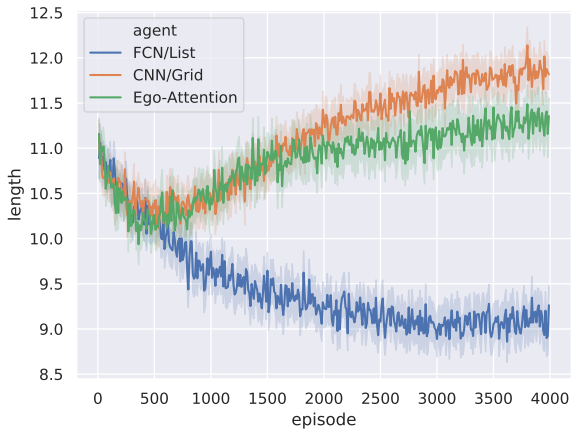
- `CNN/Grid`: a *spatial grid* representation is used, as described in Section 4.1.1, with a 32×32 grid where each cell represents a 2m $\times$ 2m square. The model is a Convolutional Neural Network (CNN).
- `Ego-Attention`: a *list of features* state representation is used along with the Ego-Attention architecture described in Section 4.2. As this model supports varying-size inputs, zero-padding is not required.

These agents are all trained with the DQN algorithm using the same hyperparameters, and their architectures are scaled to admit about the same number of trainable parameters for fair comparison.

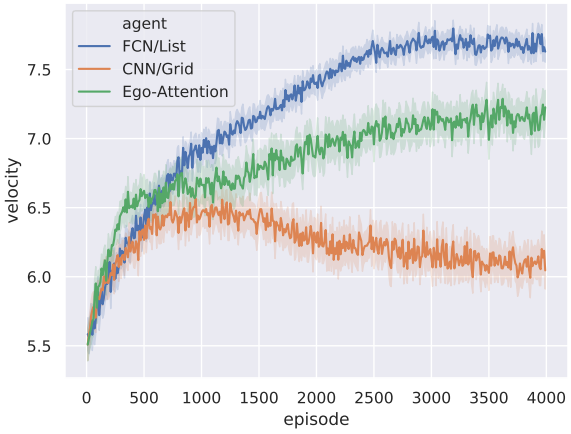
Performances We plot in Figure 4.4 the evolution of the total reward, episode length and average speed during training, over 4000 episodes and repeated across 120 random seeds. The `FCN/List` agent learns to accelerate to earn short-term rewards, as shown by its high average speed, but fails to exploit the information of other vehicles and crashes often, leading to short episodes. We obtain a risky and blind policy that is the worst-performing. Conversely, the `CNN/Grid` architecture benefits from its invariance to permutations and manages to learn to brake upon arrival at the intersection to avoid collisions, as we can see from its higher episode length. However, it only proceeds when the intersection has been fully cleared, as reflected by its low average speed. This results in an overly cautious policy – a common trait colloquially known as the *"freezing robot problem"* (Trautman and Krause, 2010) – with a slight increase in performance. In stark contrast, the `Ego-Attention` policy quickly learns both when it must slow down at the intersection (see the high episode length), but also when it can exploit the gaps in the traffic and take way to vehicles that are far or slow enough (see the higher average

speed than `CNN/Grid`). This translates as a significant performance improvement, and the resulting overall behaviour is qualitatively more nuanced and human-like.

(a) Average return $G = \sum_t \gamma^t R(s_t, a_t)$ (higher is better)



(b) Average episode length. Higher is better, since episodes are terminated at collisions (or after 13 steps).



(c) Average speed v_0 of the ego-vehicle (higher is better)

Figure 4.4 – Performances of the tree agents according to various measures. We display the mean values – along with their 95% confidence interval – averaged over 120 random seeds.

Attention interpretation In any given state, the attention matrix can be visualised in the following way: we connect the ego-vehicle to every vehicle by a line of width proportional to the corresponding attention weight. Since the architecture can contain several ego-attention heads, we use different colours to distinguish them. In our experiments, two attention heads were used and are represented in green and blue. We observe in Figure 4.5 that they specialised to focus on different areas: the green head is only watching the vehicles coming from the left,

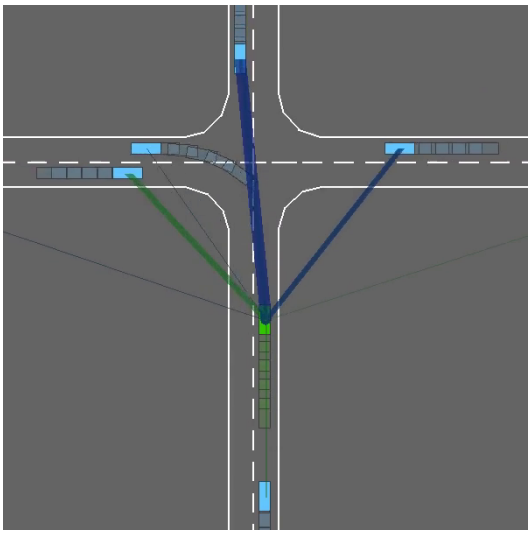


Figure 4.5 – The attention heads specialised in different areas: left and front/right.

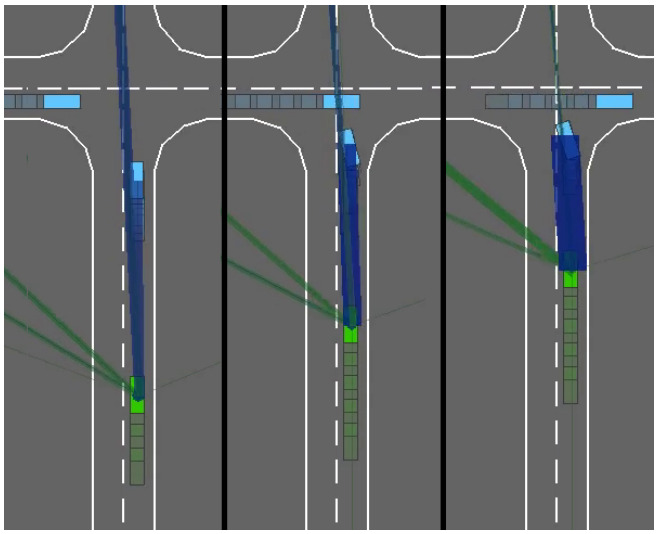
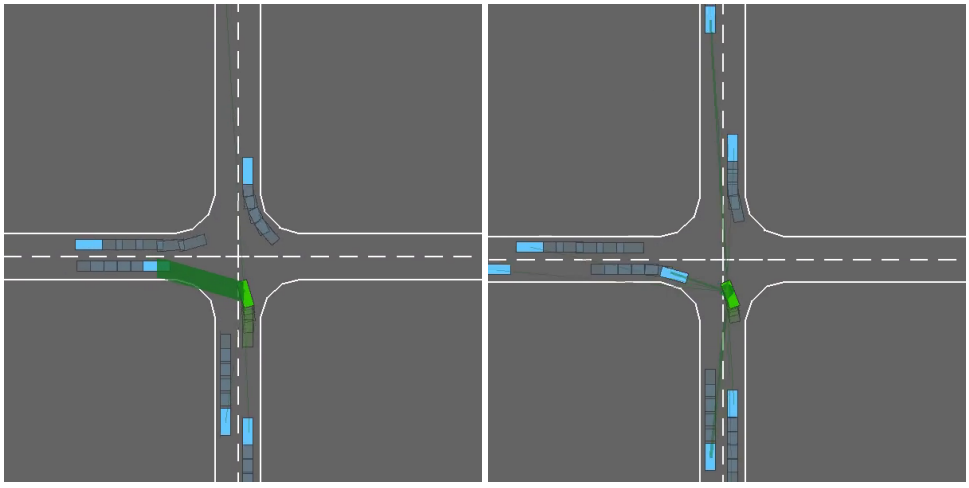


Figure 4.6 – The attention paid to a vehicle tends to increase as it gets closer.

while the blue head restricts itself to vehicles in the front and right directions. However, we notice that both heads exhibit a common behaviour: they direct their attention to incoming vehicles that are likely to collide with the ego-vehicle, depending on their current position, heading, velocity, and ignore those that are too far or in a conflict-less situation. In particular, the attention tends to increase when vehicles get closer, as shown in Figure 4.6. It can also be very sensitive to small variations in the traffic state, as reflected in Figure 4.7. A full episode showcasing interactions with several vehicles is shown in Figure 4.8.

Exploiting interaction patterns The agent decisions regarding the right of way are not enforced through rewards but experienced interactions: based on the defined road priorities, some vehicles will take way to the ego-vehicle while others will not. By changing which is a priority road, we can influence the rules of interactions which affects the learnt behaviour. In Figure 4.9, we compare two policies placed in the exact same initial state and observe how their decisions are affected by their internal model of how incoming vehicles interact with them. This difference showcases the ability of our proposed architecture to discover and exploit such interaction patterns.

Goal conditioning In the previous examples, we trained a policy tailored for left-turns only because it is the hardest direction with the most conflict points and the lowest priority level. Two individual policies tailored for right turns and driving straight can be trained as well, with similar results. Training a generic intersection policy would be less efficient without any prior information on where the ego-vehicle is headed. To remedy this problem, the destination could be added as additional features in (4.1), for instance encoded as a desired direction



(**a**) The agent has stopped at the intersection, its attention is focused on an incoming vehicle whose destination is still uncertain.

(**b**) As soon as the vehicle orientation changes, revealing its intention of turning right, the attention drops and the agent starts accelerating right away.

Figure 4.7 – Sensitivity to uncertainty.

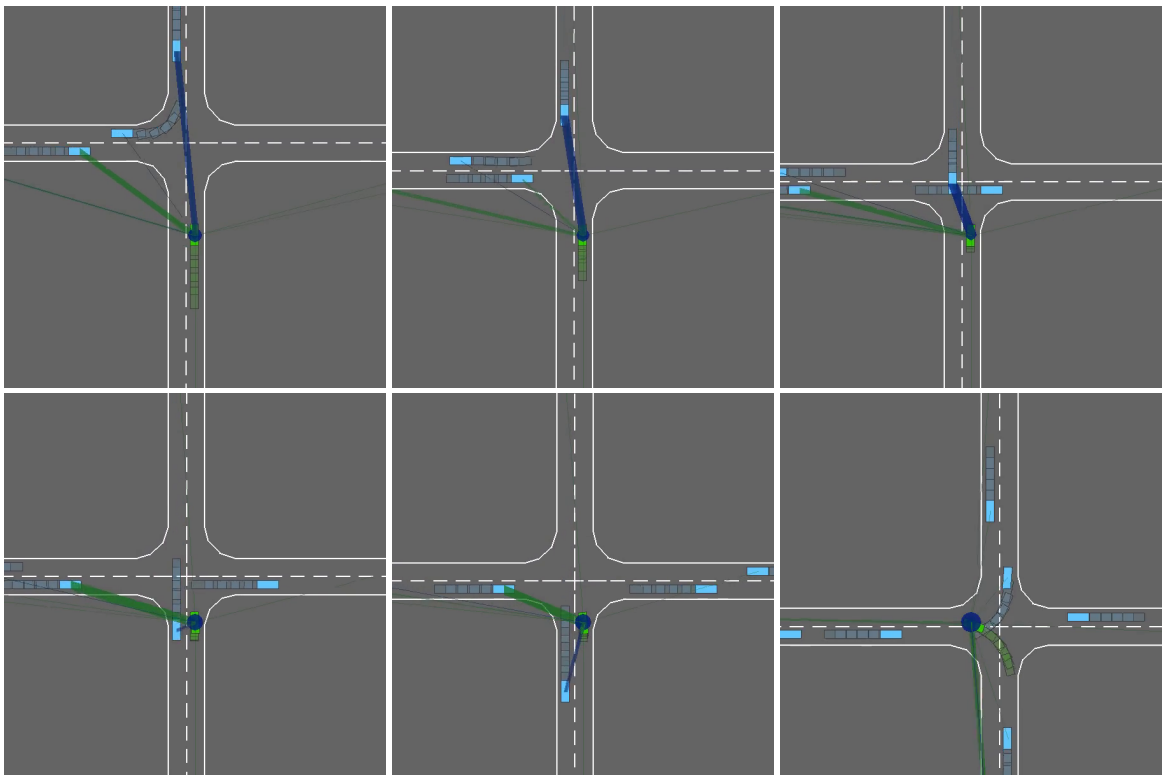
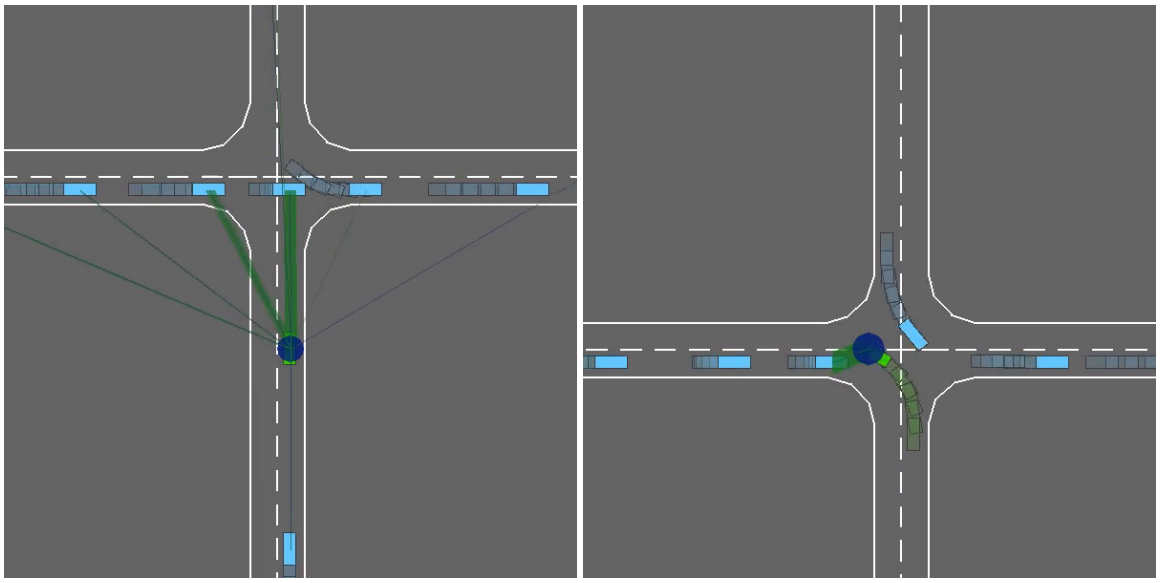


Figure 4.8 – A complete episode. *From left to right, top to bottom*: 1. The green and blue heads direct their attentions to the left and front vehicles, respectively; 2. The left-vehicle is passing and is no longer a threat; 3. Immediately, the green attention head switches to the next vehicle coming from the left; 4. The front vehicle has now passed, and the blue attention head is now focused on the ego-vehicle; 5. The ego-vehicle waits for one last vehicle coming from the left 6. The ego-vehicle can finally proceed, and its attention is focused on itself.



(**a**) When trained on a non-priority road, the agent learns to yield to incoming vehicles.

(**b**) When trained on a priority road, the agent expects other vehicles to give way and is consequently more aggressive.

Figure 4.9 – Effect of the right of way.

(d_x, d_y). This destination feature could also be used for other traffic participants to encode blinker information when available. This should result in a more efficient and generic policy.

Chapter conclusion

In this chapter, we showed that the *list of features* representation, commonly used to describe vehicles in autonomous driving literature, is not tailored for use in a function approximation setting, in particular with neural networks. These concerns can be addressed by the *spatial grid* representation, but at the price of an increased input size and loss of accuracy. In contrast, we proposed an attention-based neural network architecture to tackle the aforementioned issues of the *list of features* representation without compromising either size or accuracy. This architecture enjoys a better performance on a simulated negotiation and intersection crossing task, and is also more interpretable thanks to the visualisation of the attention matrix. The resulting policy successfully learns to recognise and exploit the interaction patterns that govern the nearby traffic.

Let us back up a moment and reflect again on the behaviours exhibited by Figure 4.4. Though `Ego-Attention` performs better than `CNN/Grid`, it also has shorter episodes, which means it collides more often with other vehicles. Though this aggressive behaviour is deemed better by our choice of reward function, it may not be desirable in practice. A straightforward way to make the optimal policy more conservative is to manually tune the reward function

and increase the weight of collisions. However, setting too high a penalty may also result in overly cautious behaviours. Finding the sweet spot between these two extremes can be difficult and demanding since changing the reward requires retraining the policy entirely. In the next chapter, we study a way to learn not a single policy but rather a whole range of policies that exhibit different levels of risk.

Chapter 5

Acting under Adjustable Constraints

If you can meet with Triumph and Disaster
And treat those two impostors just the same;
.....
If you can make one heap of all your winnings
And risk it on one turn of pitch-and-toss

Rudyard Kipling, *If–*.

When we drive, we must comply with two contradictory objectives: *efficiency* and *safety*. In this chapter, we strive to reconcile them by formalising a first notion of *risk*. We consider BMDPs, in which risk is implemented as a cost signal constrained to lie below an *adjustable* threshold. The latter provides the system manufacturer with a slider allowing them to adjust in real-time the level of risk taken by the vehicle. So far, BMDPs could only be solved in the case of known dynamics and finite state spaces, which is not suitable for our application which features continuous kinematic states and unknown human behaviours. This chapter extends the state-of-the-art to continuous spaces and unknown dynamics. Our approach is motivated by the prospect of training a risk-sensitive driving policy for a two-way road, where overtaking a vehicle requires driving on the wrong lane.[1]

Contents

[1]This chapter is based on an article published in the proceedings of the *32nd conference on advances in Neural Information Processing Systems* (*NeurIPS*) (Carrara, Leurent, et al., 2019). It is joint work with Nicolas Carrara, who came up with the algorithm and carried-out the dialogue experiment. I did most of the theoretical analysis, the driving experiment; and we worked together on the exploration procedure and scaling-up the implementation.

5.1 Motivation

As stated in Chapter 1, Reinforcement Learning is a general framework for decision-making under uncertainty. Formally, we seek a policy $\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}$ that maximises in expectation the γ-discounted return of rewards $G^\pi = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$.

However, this modelling assumption comes at a price: no control is given over the spread of the performance distribution (Dann et al., 2019). In many critical real-world applications, including Autonomous Driving, failures may turn out very costly. This is an issue as most decision-makers would rather give away some amount of expected optimality to increase the performances in the lower-tail of the distribution. As discussed in Chapter 2, this has led to the development of several risk-averse variants where the optimisation criteria include other statistics of the performance, such as the worst-case realisation (Iyengar, 2005; Nilim and El Ghaoui, 2005; Wiesemann, Kuhn, and Rustem, 2013), the variance-penalised expectation (Tamar, Di Castro, and Mannor, 2012; García, Fern, and Fernández, 2015), the Value at Risk (Mausser and Rosen, 1999; Luenberger, 2013), or the Conditional Value at Risk (Chow, Tamar, et al., 2015; Chow, Ghavamzadeh, et al., 2017).

Reinforcement Learning also assumes that the performance can be described by a single reward function R. Conversely, real problems typically involve many aspects, some of which can be contradictory (C. Liu, X. Xu, and Hu, 2014). In our case, a self-driving car needs to balance between progressing quickly on the road and avoiding collisions. In Multi-Objective Reinforcement Learning (MORL), each of these aspects is independently modelled by a separate reward signal. Then, the set of policies is partitioned into (i) the class of *dominated* policies π, for which there exists an *improvement*, *i.e.* another policy π' with at least some objectives increased, and none decreased; (ii) the *Pareto front* $\Pi^\star$, of undominated policies, illustrated in Figure 5.1.

A standard way to cast a MOMDP into an MDP is to aggregate several objectives in a single reward function (Roijers et al., 2013). However, this does not allow to explicitly control the trade-off between the different objectives, since higher rewards can compensate for higher penalties. For instance, if a weighted sum is used to balance velocity v and crashes c, then for any given choice of weights ω the optimality equation $\omega_v \, \mathbb{E}[\sum \gamma^t v_t] + \omega_a \, \mathbb{E}[\sum \gamma^t c_t] = G^\star = \max_\pi G^\pi$ is the equation of a line in $(\mathbb{E}[\sum \gamma^t v_t], \mathbb{E}[\sum \gamma^t c_t])$, and the automotive company cannot control where its optimal policy $\pi^\star$ lies on that line.

Both of these concerns are addressed in the Constrained Markov Decision Process (CMDP) setting (Beutler and K. W. Ross, 1985; Altman, 1999), illustrated in Figure 5.2a. In this multi-objective formulation, task completion and safety are considered separately. We equip the MDP with a cost signal $C \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ and a cost budget $\beta \in \mathbb{R}$. Similarly to G^π, we define the

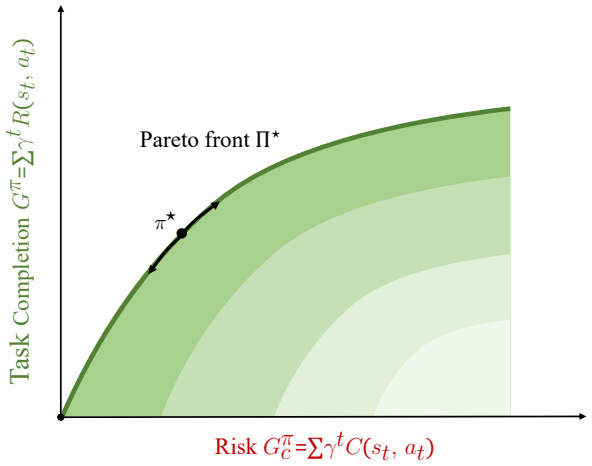


Figure 5.1 – A Multi-Objective Markov Decision Process (MOMDP) with two objectives: the rewards R must be maximised, while the costs C must be minimised. The policies are partitioned into dominated policies, shown in light shades of green, and the Pareto front $\Pi^\star$, shown in dark green. Cautious policies with low efficiency and risk are located on the bottom-left, while aggressive policies with high efficiency and risk are on the top-right.

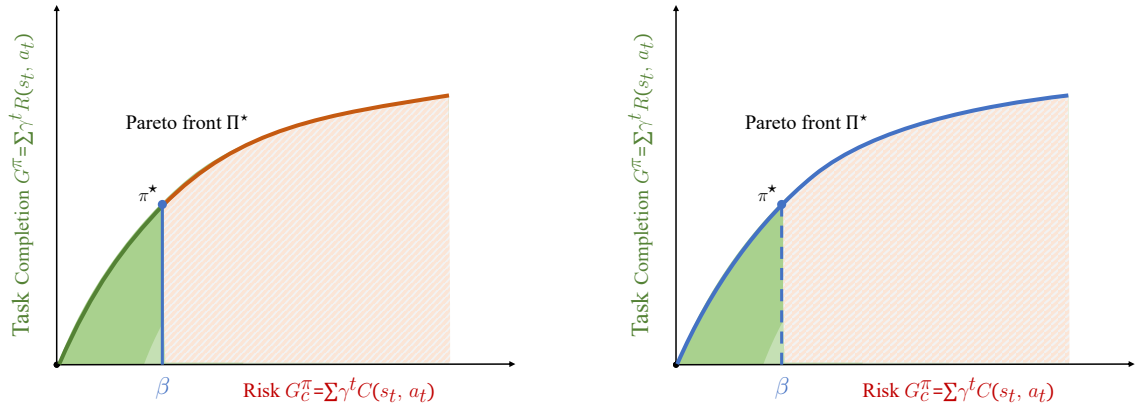
return of costs $\boldsymbol{G_c^\pi} = \sum_{t=0}^{\infty} \gamma^t C(s_t, a_t)$ and the new cost-constrained objective:

$$\max_{\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}} \mathbb{E}[\boldsymbol{G^\pi} | s_0 = s] \quad \text{s.t.} \quad \mathbb{E}[\boldsymbol{G_c^\pi} | s_0 = s] \leq \boldsymbol{\beta} \tag{5.1}$$

This constrained framework allows for better control of the performance-safety trade-off. However, it suffers from a major limitation: the budget has to be chosen before training, and cannot be changed afterwards.

To tackle this issue, the Budgeted Markov Decision Process (BMDP) was introduced in (Boutilier and T. Lu, 2016) as an extension of CMDPs to enable the online control over the budget β within a closed interval $\mathscr{B} \subset \mathbb{R}$ of admissible budgets. Instead of fixing the budget prior to training, the objective is now to find a generic optimal policy $\pi^\star$ that takes β as input so as to solve the corresponding CMDP (5.1) for all budgets $\beta \in \mathscr{B}$. This gives the system designer the ability to move in real-time the optimal policy $\pi^\star$ along the Pareto front of the different reward-cost trade-offs, as shown in Figure 5.2b.

In the seminal work of Boutilier and T. Lu (2016), BMDPs were originally studied in the context of finite states $\mathcal{S}$, finite horizon, and known BMDP parameters. Our first contribution is to re-frame the BMDP formulation in the context of continuous states and infinite discounted horizon. We then propose a novel Budgeted Bellman Optimality Operator and prove the optimal value function to be a fixed point of this operator. Second, we use this operator in `Budgeted Fitted Q-Learning (BFTQ)`, a batch RL algorithm, for solving BMDPs online, without prior knowledge of the (P, R, C) parameters, by interacting with an environment. Third, we scale



(**a**) In a CMDP, we learn a single policy $\pi^\star$ (blue dot •) with a fixed expected risk $\beta \in \mathscr{B}$

(**b**) In a BMDP, we learn a set $\Pi^\star$ of policies called the Pareto front (blue frontier –), for the whole range $\mathscr{B}$ of allowed risks

Figure 5.2 – Comparison between the CMDP and BMDP frameworks.

this algorithm to large problems by (i) providing an efficient implementation of the Budgeted Bellman Optimality operator based on convex programming, (ii) a tailored risk-sensitive exploration procedure, and (iii) leveraging tools from Deep Reinforcement Learning such as Neural Networks for function approximation and synchronous parallel computing. Finally, we validate our approach in two environments that display a clear trade-off between rewards and costs: a dialogue system example, and a behavioural planning problem for overtaking on a two-way road.

5.2 Budgeted dynamic programming

We work in the space of budgeted policies, where π both depends on β and also outputs the next budget β_a. Hence, the budget β is neither fixed nor constant as in the CMDP setting but instead evolves as part of the dynamics.

We cast the BMDP problem as a MOMDP problem (Roijers et al., 2013) by considering *augmented* state and action spaces $\overline{\mathcal{S}} = \mathcal{S} \times \mathscr{B}$ and $\overline{\mathcal{A}} = \mathcal{A} \times \mathscr{B}$, and equip them with the augmented dynamics $\overline{P} \in \mathcal{M}(\overline{\mathcal{S}})^{\overline{\mathcal{S}} \times \overline{\mathcal{A}}}$ defined as:

$$\overline{P}\left(\overline{s}' \mid \overline{s}, \overline{a}\right) = \overline{P}\left((s', \beta') \mid (s, \beta), (a, \beta_a)\right) \triangleq P(s'|s,a)\delta(\beta' - \beta_a), \tag{5.2}$$

where δ is the Dirac indicator distribution.

In other words, in these augmented dynamics, the output budget β_a returned at time t by a budgeted policy $\overline{\pi} \in \overline{\Pi} = \mathcal{M}(\overline{\mathcal{A}})^{\overline{\mathcal{S}}}$ will be used to condition the policy at the next timestep $t+1$.

We stack the rewards and cost functions in a single *vectorial* signal $\overline{R} \in (\mathbb{R}^2)^{\overline{\mathcal{S}} \times \overline{\mathcal{A}}}$:

Definition 5.1. *Given an augmented transition $(\overline{s}, \overline{a}) = ((s, \beta), (a, \beta_a))$, we define*

$$\overline{R}(\overline{s}, \overline{a}) \triangleq \begin{bmatrix} R(s,a) \\ C(s,a) \end{bmatrix} \in \mathbb{R}^2. \tag{5.3}$$

Likewise, we augment the return:

Definition 5.2. *The return $\overline{G}^{\overline{\pi}} = (G^{\overline{\pi}}, G_c^{\overline{\pi}})$ of a budgeted policy $\overline{\pi} \in \overline{\Pi}$ refers to*

$$\overline{G}^{\overline{\pi}} \triangleq \sum_{t=0}^{\infty} \gamma^t \overline{R}(\overline{s}_t, \overline{a}_t). \tag{5.4}$$

as well as the value functions:

Definition 5.3. *The value functions $\overline{V}^{\overline{\pi}}$, $\overline{Q}^{\overline{\pi}}$ of a budgeted policy $\overline{\pi} \in \overline{\Pi}$ are defined as*

$$\begin{aligned} \overline{V}^{\overline{\pi}}(\overline{s}) &= (V_r^{\overline{\pi}}, V_c^{\overline{\pi}}) \triangleq \mathbb{E}\left[\overline{G}^{\overline{\pi}} \,\middle|\, \overline{s_0} = \overline{s}\right] \\ \overline{Q}^{\overline{\pi}}(\overline{s}, \overline{a}) &= (Q_r^{\overline{\pi}}, Q_c^{\overline{\pi}}) \triangleq \mathbb{E}\left[\overline{G}^{\overline{\pi}} \,\middle|\, \overline{s_0} = \overline{s}, \overline{a_0} = \overline{a}\right]. \end{aligned} \tag{5.5}$$

We restrict $\overline{\mathcal{S}}$ to feasible budgets only: $\overline{\mathcal{S}}_f \triangleq \{(s, \beta) \in \overline{\mathcal{S}} : \exists \overline{\pi} \in \overline{\Pi}, V_c^{\overline{\pi}}(s) \leq \beta\}$ that we assume to be non-empty for the BMDP to admit a solution. We still write $\overline{\mathcal{S}}$ in place of $\overline{\mathcal{S}}_f$ for brevity of notations.

Proposition 5.4 (Budgeted Bellman Expectation)**.** *The value functions $\overline{V}^{\overline{\pi}}$ and $\overline{Q}^{\overline{\pi}}$ verify:*

$$\overline{V}^{\overline{\pi}}(\overline{s}) = \sum_{\overline{a} \in \overline{\mathcal{A}}} \overline{\pi}(\overline{a}|\overline{s}) \overline{Q}^{\overline{\pi}}(\overline{s}, \overline{a}) \qquad \overline{Q}^{\overline{\pi}}(\overline{s}, \overline{a}) = \overline{R}(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \overline{P}\left(\overline{s}' \mid \overline{s}, \overline{a}\right) \overline{V}^{\overline{\pi}}(\overline{s}'). \tag{5.6}$$

Moreover, consider the Budgeted Bellman Evaluation operator $\overline{\mathcal{T}}^{\overline{\pi}}$: $\forall \overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{S}}\overline{\mathcal{A}}}, \overline{s} \in \overline{\mathcal{S}}, \overline{a} \in \overline{\mathcal{A}}$,

$$\overline{\mathcal{T}}^{\overline{\pi}} \overline{Q}(\overline{s}, \overline{a}) \triangleq \overline{R}(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \sum_{\overline{a}' \in \overline{\mathcal{A}}} \overline{P}(\overline{s}'|\overline{s}, \overline{a}) \overline{\pi}(\overline{a}'|\overline{s}') \overline{Q}(\overline{s}', \overline{a}'). \tag{5.7}$$

Then $\overline{\mathcal{T}}^{\overline{\pi}}$ is a γ-contraction and $\overline{Q}^{\overline{\pi}}$ is its unique fixed point.

Proof. We provide the proof in Section B.1.1. □

We now come to the definition of budgeted optimality. We want an optimal budgeted policy to: (i) respect the cost budget β; (ii) maximise the γ-discounted return of rewards G; (iii) in case of tie, minimise the γ-discounted return of costs G_c.

Definition 5.5 (Budgeted Optimality)**.** *To that end, we define for all $\overline{s} \in \overline{\mathcal{S}}$,*

(i) admissible policies $\overline{\Pi}_a$ as

$$\overline{\Pi}_a(\overline{s}) \triangleq \{\overline{\pi} \in \overline{\Pi} : \boldsymbol{V_c^{\overline{\pi}}(\overline{s})} \leq \boldsymbol{\beta}\} \text{ where } \overline{s} = (s, \beta); \tag{5.8}$$

(ii) the optimal value function for rewards $V_r^\star$ and candidate policies $\overline{\Pi}_r$ as

$$V_r^\star(\overline{s}) \triangleq \max_{\overline{\pi} \in \overline{\Pi}_a(\overline{s})} V_r^{\overline{\pi}}(\overline{s}) \qquad \overline{\Pi}_r(\overline{s}) \triangleq \arg\max_{\overline{\pi} \in \overline{\Pi}_a(\overline{s})} V_r^{\overline{\pi}}(\overline{s}); \tag{5.9}$$

(iii) the optimal value function for costs $V_c^\star$ and optimal policies $\overline{\Pi}^\star$ as

$$V_c^\star(\overline{s}) \triangleq \min_{\overline{\pi} \in \overline{\Pi}_r(\overline{s})} V_c^{\overline{\pi}}(\overline{s}), \qquad \overline{\Pi}^\star(\overline{s}) \triangleq \arg\min_{\overline{\pi} \in \overline{\Pi}_r(\overline{s})} V_c^{\overline{\pi}}(\overline{s}). \tag{5.10}$$

We define the budgeted action-value function $\overline{Q}^\star$ similarly as

$$Q_r^\star(\overline{s}, \overline{a}) \triangleq \max_{\overline{\pi} \in \overline{\Pi}_a(\overline{s})} Q_r^{\overline{\pi}}(\overline{s}, \overline{a}) \qquad Q_c^\star(\overline{s}, \overline{a}) \triangleq \min_{\overline{\pi} \in \overline{\Pi}_r(\overline{s})} Q_c^{\overline{\pi}}(\overline{s}, \overline{a}), \tag{5.11}$$

and denote $\overline{V}^\star = (V_r^\star, V_c^\star)$, $\overline{Q}^\star = (Q_r^\star, Q_c^\star)$.

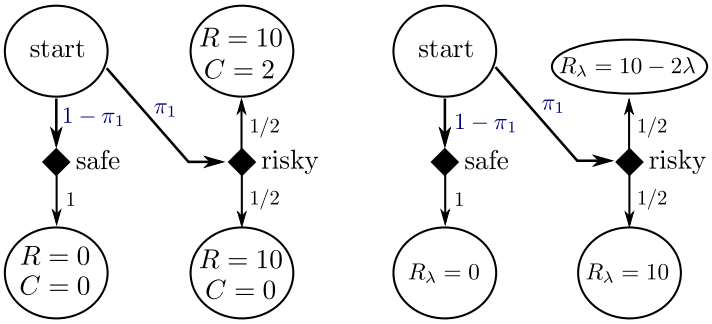


Figure 5.3 – On the left hand side, a simple *risky-vs-safe* BMDP. The probability of picking the risky action is π_1. On the right hand side an attempt to relax the problem with negative rewards.

Contrary to MDPs which always admit a deterministic optimal policy, this is generally not the case in a CMDP, and *a fortiori* in a BMDP. To illustrate this fact, let us consider the trivial BMDP on the left of Figure 5.3. In this example we have $G^\pi = 10\pi_1$ and $G_c^\pi = \pi_1$. The

deterministic policy consisting in always picking the safe action is feasible for any $\beta \geq 0$. But if $\beta = 1/2$, the most rewarding feasible policy is to randomly pick the safe and risky actions with equal probabilities. If we attempt to cast this BMDP into an MDP by replacing the costs by negative rewards, the corresponding greedy policy will be deterministic, hence sub-optimal.

However, the optimal value function $\overline{Q}^\star$ in a BMDP can still be characterised by a fixed-point equation, similarly to Theorem 4.2 for MDPs.

Theorem 5.6 (Budgeted Bellman Optimality)**.** *The optimal budgeted action-value function $\overline{Q}^\star$ verifies*

$$\overline{Q}^\star(\overline{s},\overline{a}) = \overline{\mathcal{T}}^\star \overline{Q}^\star(\overline{s},\overline{a}) \triangleq \overline{R}(\overline{s},\overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \overline{P}(\overline{s}'|\overline{s},\overline{a}) \sum_{\overline{a}' \in \overline{\mathcal{A}}} \overline{\pi}_{greedy}(\overline{a}'|\overline{s}';\overline{Q}^\star)\overline{Q}^\star(\overline{s}',\overline{a}'), \quad (5.12)$$

where the greedy policy $\overline{\pi}_{greedy}$ is defined by: $\forall \overline{s} = (s,\beta) \in \overline{\mathcal{S}}, \overline{a} \in \overline{\mathcal{A}}, \forall \overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{A}}\times\overline{\mathcal{S}}}$,

$$\overline{\pi}_{greedy}(\overline{a}|\overline{s};\overline{Q}) \in \arg\min_{\rho \in \overline{\Pi}_r^{\overline{Q}}} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_c(\overline{s},\overline{a}), \quad (5.13a)$$

$$\text{where} \quad \overline{\Pi}_r^{\overline{Q}} \triangleq \arg\max_{\rho \in \mathcal{M}(\overline{\mathcal{A}})} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_r(\overline{s},\overline{a}) \quad (5.13b)$$

$$\text{s.t.} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_c(\overline{s},\overline{a}) \leq \beta. \quad (5.13c)$$

Proof. We provide the proof in Section B.1.2. □

Remark 5.7 (Appearance of the greedy policy)**.** *In classical Reinforcement Learning, the greedy policy takes a simple form $\pi_{greedy}(s;Q^\star) = \arg\max_{a\in\mathcal{A}} Q^\star(s,a)$, and the term $\pi_{greedy}(a'|s';Q^\star)Q^\star(s',a')$ in (5.12) conveniently simplifies to $\max_{a'\in\mathcal{A}} Q^\star(s',a')$. Unfortunately, in a budgeted setting the greedy policy requires solving the nested constrained optimisation program (5.13) at each state and budget in order to apply this Budgeted Bellman Optimality operator.*

Proposition 5.8 (Optimality of the greedy policy)**.** *The greedy policy $\overline{\pi}_{greedy}(\cdot\,;\overline{Q}^\star)$ is uniformly optimal:*

$$\text{for all } \overline{s} \in \overline{\mathcal{S}},\ \overline{\pi}_{greedy}(\cdot\,;\overline{Q}^\star) \in \overline{\Pi}^\star(\overline{s}).$$

In particular,

$$\overline{V}^{\overline{\pi}_{greedy}(\cdot;\overline{Q}^\star)} = \overline{V}^\star \text{ and } \overline{Q}^{\overline{\pi}_{greedy}(\cdot;\overline{Q}^\star)} = \overline{Q}^\star.$$

Proof. We provide the proof in Section B.1.3. □

Algorithm 5.1: Budgeted Value Iteration

Data: P, R_r, R_c
Result: $Q^\star$
$Q_0 \leftarrow 0$
repeat
  $Q_{k+1} \leftarrow \mathcal{T} Q_k$
until *convergence* $(\|Q_{k+1} - Q_k\|_\infty \leq \varepsilon)$

Budgeted Value Iteration The Budgeted Bellman Optimality equation is a fixed-point equation, which motivates the introduction of a fixed-point iteration procedure. We introduce Algorithm 5.1, a Dynamic Programming algorithm for solving known BMDPs. If it were to converge to a unique fixed point, this algorithm would provide a way to compute $\overline{Q}^\star$ and recover the associated optimal budgeted policy $\overline{\pi}_{\text{greedy}}(\cdot\,; \overline{Q}^\star)$.

Theorem 5.9 (Non-contractivity of $\overline{\mathcal{T}}^\star$)**.** *For any BMDP $(\mathcal{S}, \mathcal{A}, P, R, C, \gamma)$ with $|\mathcal{A}| \geq 2$, $\overline{\mathcal{T}}^\star$ is* not *a contraction. Precisely,*

$$\forall \varepsilon > 0, \exists \overline{Q}^1, \overline{Q}^2 \in (\mathbb{R}^2)^{\overline{\mathcal{S}\mathcal{A}}} : \|\overline{\mathcal{T}}\overline{Q}^1 - \overline{\mathcal{T}}\overline{Q}^2\|_\infty \geq \frac{1}{\varepsilon}\|\overline{Q}^1 - \overline{Q}^2\|_\infty.$$

Proof. We provide the proof in Section B.1.4. □

Unfortunately, as $\overline{\mathcal{T}}^\star$ is not a contraction, we can guarantee neither the convergence of Algorithm 5.1 nor the unicity of its fixed points. Despite those theoretical limitations, we empirically observed the convergence to a fixed point in our experiments (Section 5.4). We provide a possible explanation:

Theorem 5.10 (Contractivity of $\mathcal{T}$ on smooth Q-functions)**.** *The operator $\overline{\mathcal{T}}^\star$ is a contraction when restricted to the subset $\mathcal{L}_\gamma$ of $\overline{Q}$-functions such that "Q_r is Lipschitz with respect to Q_c":*

$$\mathcal{L}_\gamma = \left\{ \begin{array}{l} \overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{S}\mathcal{A}}} \text{ s.t. } \exists L < \frac{1}{\gamma} - 1 : \forall \overline{s} \in \overline{\mathcal{S}}, \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \\ |Q_r(\overline{s}, \overline{a}_1) - Q_r(\overline{s}, \overline{a}_2)| \leq L |Q_c(\overline{s}, \overline{a}_1) - Q_c(\overline{s}, \overline{a}_2)| \end{array} \right\}, \tag{5.14}$$

$$\forall \overline{Q}^1, \overline{Q}^2 \in \mathcal{L}_\gamma,\ \|\overline{\mathcal{T}}\overline{Q}^1 - \overline{\mathcal{T}}\overline{Q}^2\|_\infty < \|\overline{Q}^1 - \overline{Q}^2\|_\infty.$$

Proof. We provide the proof in Section B.1.5. □

Thus, we expect that Algorithm 5.1 is likely to converge when $\overline{Q}^\star$ is smooth, but could diverge if the slope of $\overline{Q}^\star$ is too high. L^2-regularisation can be used to encourage smoothness and mitigate the risk of divergence.

5.3 Budgeted reinforcement learning

In this section, we consider BMDPs with unknown parameters that must be solved by interaction with an environment.

5.3.1 Budgeted Fitted-Q

When the BMDP is unknown, we need to adapt Algorithm 5.1 to work with a batch of samples $\mathcal{D} = \{(\overline{s}_t, \overline{a}_t, \overline{r}_t, \overline{s}'_t)\}_{t\in[1,N]}$ collected by interaction with the environment. Applying $\overline{\mathcal{T}}^\star$ in (5.12) would require computing an expectation $\mathbb{E}_{\overline{s}'\sim\overline{P}}$ over next states $\overline{s}'$ and hence an access to the model $\overline{P}$. We instead use $\hat{\overline{\mathcal{T}}}^\star$, a sampling operator, in which this expectation is replaced by

$$\hat{\overline{\mathcal{T}}}^\star \overline{Q}(\overline{s}, \overline{a}, \overline{r}, \overline{s}') \triangleq \overline{r} + \gamma \sum_{\overline{a}'\in\overline{\mathcal{A}}} \pi_{\text{greedy}}(\overline{a}'|\overline{s}'; \overline{Q})\overline{Q}(\overline{s}', \overline{a}').$$

We introduce in Algorithm 5.2 the `BFTQ` algorithm, an extension of the `Fitted Q-Learning (FTQ)` algorithm adapted to solve unknown BMDPs. Because we work with continuous state space $\mathcal{S}$ and budget space $\mathscr{B}$, we need to employ function-approximation in order to generalise to nearby states and budgets. Precisely, given a parametrized model $\overline{Q}_\theta$, we seek to minimise a regression loss $\mathcal{L}(\overline{Q}_\theta, \overline{Q}_{\text{target}}; \mathcal{D}) = \sum_{\mathcal{D}} \|\overline{Q}_\theta(\overline{s}, \overline{a}) - \overline{Q}_{\text{target}}(\overline{s}, \overline{a}, r, \overline{s}')\|_2^2$. Any model can be used, such as linear models, regression trees, or Neural Networks.

Algorithm 5.2: Budgeted Fitted-Q

Data: $\mathcal{D}$
Result: $Q^\star$
$\overline{Q}_{\theta_0} \leftarrow 0$
repeat
  $\theta_{k+1} \leftarrow \arg\min_\theta \mathcal{L}(\overline{Q}_\theta, \hat{\mathcal{T}}\overline{Q}_{\theta_k}; \mathcal{D})$
until *convergence* $(\|Q_{\theta_{k+1}} - Q_{\theta_k}\|_\infty \leq \varepsilon)$

5.3.2 Risk-sensitive exploration

In order to run Algorithm 5.2, we must first gather a batch of samples $\mathcal{D}$. The following strategy is motivated by the intuition that a wide variety of risk levels needs to be experienced during training, which can be achieved by enforcing the risk constraints during data collection. Ideally, we would need samples from the asymptotic state-budget distribution $\lim_{t\to\infty}\mathbb{P}\left(\overline{s}_t\right)$ induced by an optimal policy $\overline{\pi}^\star$ given an initial distribution $\mathbb{P}\left(\overline{s}_0\right)$, but as we are actually building this policy, it is not possible. Following the same idea of ε-greedy exploration for `FTQ`, we introduce an algorithm for risk-sensitive exploration. We follow an exploration policy: a mixture between a random budgeted policy $\overline{\pi}_{\text{rand}}$ and the current greedy policy $\overline{\pi}_{\text{greedy}}$. The batch $\mathcal{D}$ is split into several minibatches generated sequentially, and $\overline{\pi}_{\text{greedy}}$ is updated by running Algorithm 5.2 on $\mathcal{D}$ upon mini-batch completion. $\overline{\pi}_{\text{rand}}$ is designed to obtain trajectories that only explore feasible budgets: we impose that the joint distribution $\mathbb{P}\left(a, \beta_a | s, \beta\right)$ verifies $\mathbb{E}[\beta_a] \leq \beta$. This condition defines a probability simplex $\Delta_{\overline{\mathcal{A}}}$ from which we sample uniformly. Finally, when interacting with an environment, the initial state s_0 is usually sampled from a starting distribution $\mathbb{P}\left(s_0\right)$. In the budgeted setting, we also need to sample the initial budget β_0. Importantly, we pick a uniform distribution $\mathbb{P}\left(\beta_0\right) = \mathcal{U}(\mathscr{B})$ so that the entire range of risk-level is explored, and not only reward-seeking behaviours as would be the case with a traditional risk-neutral ε-greedy strategy. The pseudo-code of our exploration procedure is shown in Algorithm 5.3.

Algorithm 5.3: Risk-sensitive exploration

Data: An environment, a BFTQ solver, W CPU workers
Result: A batch of transitions $\mathcal{D}$
$\mathcal{D} \leftarrow \emptyset$
for *each intermediate batch* **do**
  split episodes between W workers
  for *each episode in batch* **do** `// run this loop on each worker in parallel`
    sample initial budget $\beta \sim \mathcal{U}(\mathcal{B})$
    while *episode not done* **do**
      update ε from schedule
      sample $z \sim \mathcal{U}([0, 1])$
      if $z < \varepsilon$ **then** sample $(a, \beta_a) \sim \mathcal{U}(\Delta_{\mathcal{AB}})$ `// Explore`
      else sample $(a, \beta_a) \sim \pi_{\text{greedy}}(a, \beta_a | s, \beta; Q^\star)$ `// Exploit`
      append transition $(s, \beta, a, \beta_a, R, C, s')$ to batch $\mathcal{D}$
      step episode budget $\beta \leftarrow \beta_a$
  $\pi_{\text{greedy}}(\cdot;\, Q^\star) \leftarrow \texttt{BFTQ}(\mathcal{D})$
return *the batch of transitions* $\mathcal{D}$

In the following, we introduce an implementation of the `BFTQ` algorithm designed to operate efficiently and handle large batches of experiences $\mathcal{D}$.

5.3.3 How to compute the greedy policy?

As stated in Remark 5.7, computing the greedy policy $\overline{\pi}_{\text{greedy}}$ in (5.12) is not trivial since it requires solving the nested constrained optimisation program (5.13). However, it can be solved efficiently by exploiting the *structure* of the set of solutions with respect to β, that is, concave and increasing.

Proposition 5.11 (Equality of $\overline{\pi}_{\text{greedy}}$ and $\overline{\pi}_{\text{hull}}$)**.** *The greedy policy $\overline{\pi}_{greedy}$ is equal to a convex-hull policy $\overline{\pi}_{hull}$ defined in Algorithm 5.4:*

$$\overline{\pi}_{greedy}(\overline{a}|\overline{s};\overline{Q}) = \overline{\pi}_{hull}(\overline{a}|\overline{s};\overline{Q}).$$

Thus, Algorithm 5.1 and Algorithm 5.2 can be run by replacing $\overline{\pi}_{greedy}$ in the equation (5.12) of $\overline{\mathcal{T}}$ with $\overline{\pi}_{hull}$.

Proof. We provide the proof in Section B.1.6. □

Algorithm 5.4: Convex hull policy $\overline{\pi}_{\text{hull}}(\overline{a}|\overline{s};\overline{Q})$

Data: $\overline{s} = (s, \beta)$, $\overline{Q}$
$\overline{Q}^+ \leftarrow \{Q_c > \min\{Q_c(\overline{s},\overline{a}) \text{ s.t. } \overline{a} \in \arg\max_{\overline{a}} Q_r(\overline{s},\overline{a})\}\}$ `// dominated points`
$\mathcal{F} \leftarrow$ top frontier of `convex_hull`$(\overline{Q}(\overline{s},\overline{\mathcal{A}}) \setminus \overline{Q}^+)$ `// candidate mixtures`
$\mathcal{F}_{\overline{Q}} \leftarrow \mathcal{F} \cap \overline{Q}(\overline{s},\overline{\mathcal{A}})$
for *points* $q = \overline{Q}(\overline{s},\overline{a}) \in \mathcal{F}_{\overline{Q}}$ *in clockwise order* **do**
  if *find two successive points* $((q_c^1, q_r^1), (q_c^2, q_r^2))$ *of* $\mathcal{F}_{\overline{Q}}$ *such that* $q_c^1 \leq \beta < q_c^2$ **then**
    $p \leftarrow (\beta - q_c^1)/(q_c^2 - q_c^1)$
    return the mixture $(1-p)\delta(\overline{a} - \overline{a}^1) + p\delta(\overline{a} - \overline{a}^2)$
else return $\delta(\overline{a} - \arg\max_{\overline{a}} Q_r(\overline{s},\overline{a}))$ `// budget` β `always respected`

The computation of π_{hull} in Algorithm 5.4 is illustrated in Figure 5.4: first we get rid of dominated points. Then we compute the top frontier of the convex hull of the Q-function. Next, we find the two closest augmented actions $\overline{a}_1$ and $\overline{a}_2$ with cost-value Q_c surrounding β: $Q_c(\overline{s},\overline{a}_1) \leq \beta < Q_c(\overline{s},\overline{a}_2)$. Finally, we mix the two actions such that the expected spent budget is equal to β. Because of the concavity of the convex hull top frontier, any other combination of augmented actions would lead to a lower expected reward Q_r.

5.3.4 Function approximation

Neural Networks are well suited to model $\overline{Q}$-functions, as is done in the DQN algorithm (Mnih et al., 2015). We approximate $\overline{Q} = (Q_r, Q_c)$ using one single Neural Network, as illustrated in

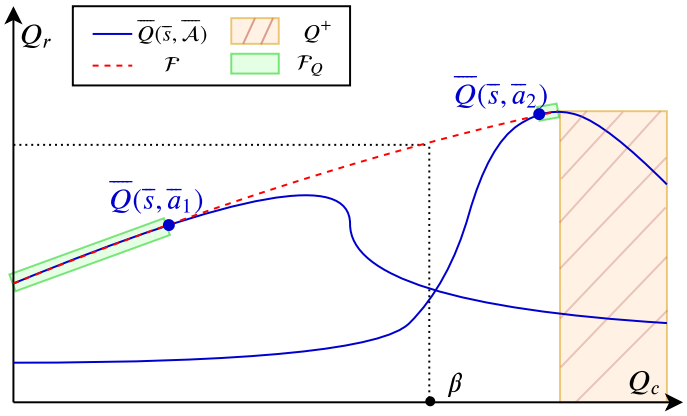


Figure 5.4 – Representation of π_{hull}. When the budget lies between $Q(\overline{s},\overline{a}_1)$ and $Q(\overline{s},\overline{a}_2)$, two points of the top frontier of the convex hull, then the policy is a mixture of these two points.

Figure 5.5. Thus, the two components are optimised jointly, which accelerates convergence and fosters the learning of useful shared representations. Moreover, as in (Mnih et al., 2015), we are dealing with a finite (categorical) action space $\mathcal{A}$. Instead of including the action in the input, we add an output to the $\overline{Q}$-function for each action. Again, it provides a faster convergence toward useful shared representations and it only requires one forward pass to evaluate all action values. Finally, beside the state s there is one more input to a budgeted $\overline{Q}$-function: the budget β_a. This budget is a scalar value whereas the state s is a vector of potentially large size. To avoid a weak influence of the budget β_a compared to the state s in the prediction, we include an additional encoder for the budget, whose width and depth may depend on the application. A straightforward choice is a single layer with the same width as the state.

5.3.5 Parallel computing

In a simulated environment, a first process that can be distributed is the collection of transitions in the exploration procedure of Algorithm 5.3, as $\overline{\pi}_{\text{greedy}}$ stays constant within each minibatch which avoids the need of synchronisation between workers. Second, the main bottleneck of `BFTQ` is the computation of the target $\overline{\mathcal{T}Q}$. Indeed, when computing $\overline{\pi}_{\text{hull}}$ we must perform at each epoch a Graham-scan of complexity $\mathcal{O}(|\mathcal{A}||\tilde{\mathscr{B}}|\log|\mathcal{A}||\tilde{\mathscr{B}}|)$ per transition in $\mathcal{D}$ to compute the convex hulls of $\overline{Q}$ (where $\tilde{\mathscr{B}}$ is a finite discretisation of $\mathscr{B}$). The resulting total time-complexity[2] is $\mathcal{O}(\frac{1}{\mathcal{D}}|\mathcal{A}||\tilde{\mathscr{B}}|\log_\gamma(\varepsilon(1-\gamma))\log|\mathcal{A}||\tilde{\mathscr{B}}|)$. This operation can easily be distributed over several CPUs provided that we first evaluate the model $\overline{Q}(\overline{s}',\mathcal{A}\times\tilde{\mathscr{B}})$ for each state $\overline{s}$ extracted from the dataset $\mathcal{D}$, which can be done in a single forward pass. By using multiprocessing in

[2] $\log_\gamma(\varepsilon(1-\gamma))$ is the sample complexity of Value Iteration with accuracy ε, and each of these iterations requires a Graham-scan for each state in the dataset $\mathcal{D}$, action $a\in\mathcal{A}$ and budget $\beta\in\tilde{\mathscr{B}}$.

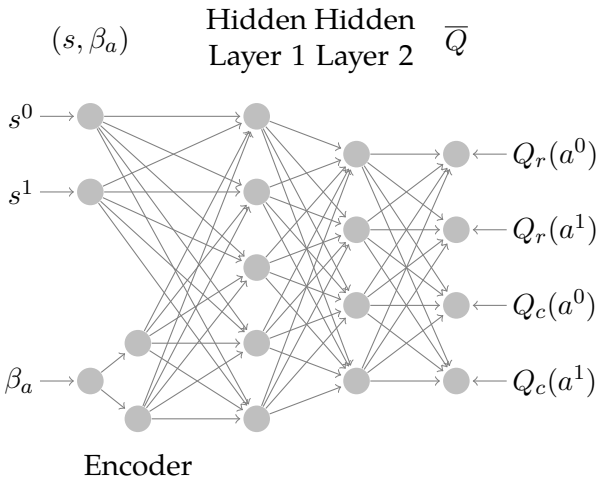


Figure 5.5 – Neural Network for Q-functions approximation when $\mathcal{S} = \mathbb{R}^2$ and $|\mathcal{A}| = 2$.

the computations of $\overline{\pi}_{\text{hull}}$, we enjoy a linear speedup. The full description of our scalable implementation of BFTQ is recalled in Algorithm 5.5.

5.4 Experiments

There are two hypotheses we want to validate.

Exploration strategies We claimed in Section 5.3.2 that a risk-sensitive exploration was required in the setting of BMDPs. We test this hypothesis by confronting our strategy to a classical risk-neutral strategy. The latter is chosen to be a ε-greedy policy slowly transitioning from a random to a greedy policy[3] that aims to maximise $\mathbb{E}_\pi G^\pi$ regardless of $\mathbb{E}_\pi G_c^\pi$. The quality of the resulting batch $\mathcal{D}$ is assessed by training a BFTQ policy and comparing the resulting performance.

Budgeted algorithms We compare our BFTQ algorithm to an FTQ(λ) baseline. This baseline consists in approximating the BMDP by a finite set of CMDPs problems. We solve each of these CMDP using the standard technique of Lagrangian Relaxation: the cost constraint is converted into a soft penalty weighted by a Lagrangian multiplier λ in a surrogate reward function: $\max_\pi \mathbb{E}_\pi[G^\pi - \lambda G_c^\pi]$. As shown in Figure 5.6, the optimal deterministic policy can be obtained by a line-search on the Lagrange multiplier values λ. Then, according to Beutler and K. W. Ross (1985, Theorem 4.4), the optimal policy is a randomised mixture of two deterministic policies: the safest deterministic policy that violates the constraint $\pi_{\lambda-}$ and the most risky feasible policy $\pi_{\lambda+}$. The resulting MDPs can be solved by any RL algorithm, and we chose

[3] We train this greedy policy using FTQ.

Algorithm 5.5: A scalable implementation of `BFTQ`

Data: $\mathcal{D}$, $\tilde{\mathcal{B}}$ a finite subset of $\mathcal{B}$, γ, a model $Q \in (\mathbb{R}^2)^{S\overline{\mathcal{A}}}$, a regression algorithm `fit`, a set of CPU workers W
Result: $Q^\star$
$Q \leftarrow 0$
$X \leftarrow \{s_i, a_i, \beta_{a_i}\}_{i\in[0,|\mathcal{D}|]}$
$S' \leftarrow \{s'_i\}_{i\in[0,|\mathcal{D}|]}$
repeat
  Evaluate $Q(S', \mathcal{A}, \tilde{\mathcal{B}})$ in a single forward pass
  Split $\mathcal{D}$ among workers: $\mathcal{D} = \cup_{w\in W}\mathcal{D}_w$
  for $w \in W$ **do** `// Run in parallel`
    for $(\cdot, \cdot, \beta_{a_i}, R_{ri}, R_{ci}, s'_i) \in \mathcal{D}$ **do**
      $\mathcal{P} \leftarrow \{(Q_c(s'_i, \mathcal{A}, \tilde{\mathcal{B}}), Q_r(s'_i, \mathcal{A}, \tilde{\mathcal{B}}))\}$
      $\mathcal{P}$.`prune()` `// Remove all dominated points`
      $\mathcal{H} \leftarrow$ `convex_hull`$(\mathcal{P})$.vertices() `// in cw order`
      $k \leftarrow \min\{k : \beta_i \geq q_c \text{ with } (q_c, q_r) = \mathcal{H}[k]\}$
      $q_c^2, q_r^2, q_c^1, q_r^1 \leftarrow \mathcal{H}[k], \mathcal{H}[k-1]$
      $p \leftarrow (\beta_{a_i} - q_a^1)/(q_c^2 - q_c^1)$
      $Y_c^{w,i} \leftarrow R_{ci} + \gamma((1-p)q_c^1 + pq_c^2)$
      $Y_r^{w,i} \leftarrow R_{ri} + \gamma((1-p)q_r^1 + pq_r^2)$
  Join the results: $Y \leftarrow \cup_{w\in W}(Y_c^w, Y_r^w)$
  $Q \leftarrow$ `fit`(X, Y)
until *convergence*

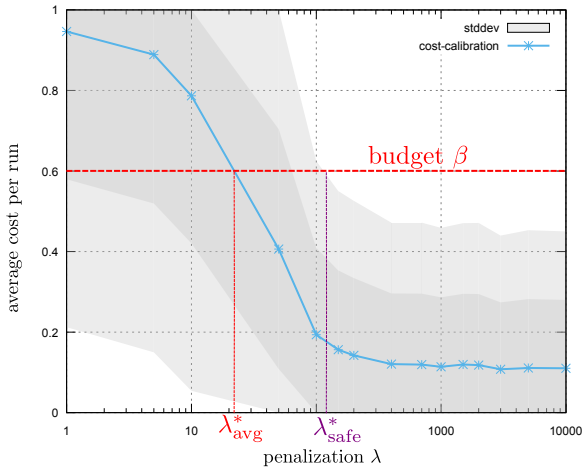


Figure 5.6 – Calibration of a penalty multiplier according to the budget β. The optimal multiplier $\lambda^\star_{\text{avg}}$ is the smallest one to satisfy the budget constraint on average. Safer policies can also be selected according to the largest deviation from this mean cost.

`FTQ` for being closest to `BFTQ`. In our experiments, a single training of `BFTQ` corresponds to 10 training runs for `FTQ`(λ) policies. Each run was repeated N_{seeds} times. Given the high variance, it requires a lot of simulations to get a proper estimate of the calibration curve. Our purpose is to avoid this calibration phase.

5.4.1 Environments

We evaluate our method on three different environments involving reward-cost trade-offs.

Corridors This simple environment is only meant to highlight clearly the specificity of exploration in a budgeted setting. It is a continuous gridworld with Gaussian perturbations, consisting in a maze composed of two corridors: a risky one with high rewards and costs, and a safe one with low rewards and no cost. In both corridors, the outermost cell is the one yielding the most reward, which motivates an in-depth exploration.

Parameter	Description	Value
-	Size of the environment	7 x 6
-	Standard deviation of the Gaussian noise applied to actions	(0.25,0.25)
H	Trajectory duration	9

Table 5.1 – Parameters of `Corridors`

Spoken dialogue system Our second application is a dialogue-based slot-filling simulation that has already benefited from batch RL optimisation in the past (L. Li, Williams, and S. Balakrishnan, 2009; Chandramohan, Geist, and Pietquin, 2010; Pietquin et al., 2011). The system fills in a form of slot-values by interacting a user through speech, before sending them a response. For example, in a restaurant reservation domain, it may ask for three slots: the area of the restaurant, the price range and the food type. The user could respectively provide those three slot-values : `Cambridge`, `Cheap` and `Indian-food`. In this application, we do not focus on how to extract such information from the user utterances; but rather on the decision-making for filling in the form. To that end, the system can choose among a set of generic actions. As in (Carrara, Laroche, et al., 2018), there are two ways of asking for a slot value: a slot value can be either be provided with an utterance, which may cause speech recognition errors with some probability, or by requiring the user to fill in the slots by using a numeric pad. In this case, there are no recognition errors but a counterpart risk of hang-up: we assume that manually filling a key-value form is time-consuming and annoying. The environment yields a reward if all slots are filled without errors, and a constraint if the user hang-ups. Thus, there is a clear trade-off between using utterances and potentially committing a mistake, or using the numeric pad and risking a premature hang-up.

Parameter	Description	Value
ξ	Sentence Error Rate	0.6
$\mu_\perp$	Gaussian mean for misunderstanding	-0.25
$\mu_\top$	Gaussian mean for understanding	0.25
σ	Gaussian standard deviation	0.6
p	Probability of hang up	0.25
H	Trajectory duration	10
-	Number of slots	3

Table 5.2 – Parameters of `Slot-Filling`

Two-way road In our third application, we use the HIGHWAY-ENV environment presented in Chapters A and 3. We define a task that displays a clear trade-off between safety and efficiency, illustrated in Figure 5.7. As we mentioned, the agent controls a vehicle with a finite set $\mathcal{A}$ of manoeuvres (3.2) implemented by low-level controllers. It is driving on a two-way road populated with other traffic participants: the vehicles in front of the agent drive slowly, and there are incoming vehicles on the opposite lane. The parameters controlling their behaviours are randomised, which introduces some uncertainty concerning their possible future trajectories. The task consists in driving as fast as possible, which is modelled by a reward proportional to the velocity: $R(s_t, a_t) \propto v_t$. This motivates the agent to try and overtake its preceding vehicles by driving fast on the opposite lane. This optimal but overly aggressive behaviour can be tempered through a cost function that embodies a safety objective: $C(s_t, a_t)$ is set to $1/H$

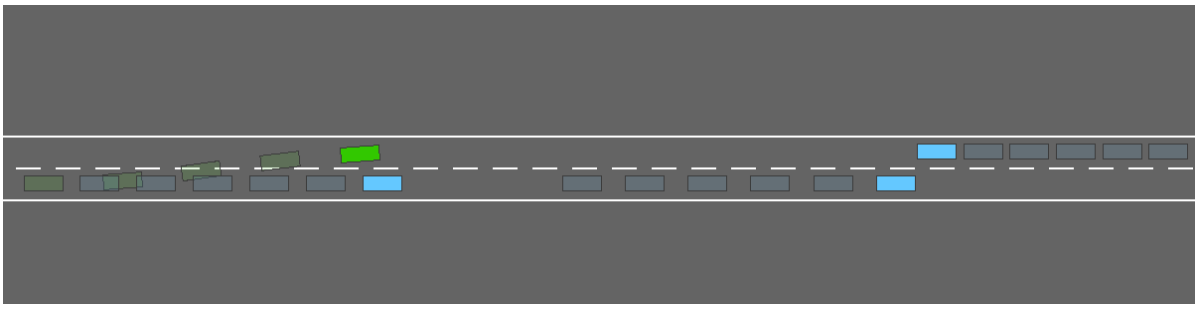


Figure 5.7 – The two-way road environment requires the vehicle to drive in the wrong lane and risk front collisions in order to overtake slow vehicles.

whenever the ego-vehicle is driving on the opposite lane, where H is the trajectory horizon. Thus, the constrained signal G_c is the maximum proportion of time that the agent is allowed to drive on the wrong side of the road.

Parameter	Description	Value
N_v	Number of other vehicles	2 - 6
σ_p	Standard deviation of vehicles initial positions	100 m
σ_v	Standard deviation of vehicles initial velocities	3 m s^{-1}
H	Trajectory duration	15 s

Table 5.3 – Parameters of `highway-env`

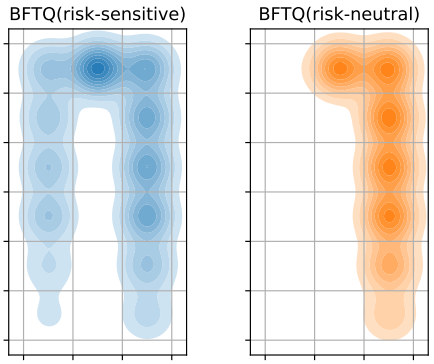
5.4.2 Results

In the following figures, each patch represents the mean and 95% confidence interval over N_{seeds} seeds of the means of (G^π, G_c^π) ($(G^{\overline{\pi}}, G_c^{\overline{\pi}})$ for `BFTQ`) over N_{trajs} trajectories. That way, we display the variation related to learning (and batches) rather than the variation in the execution of the policies.

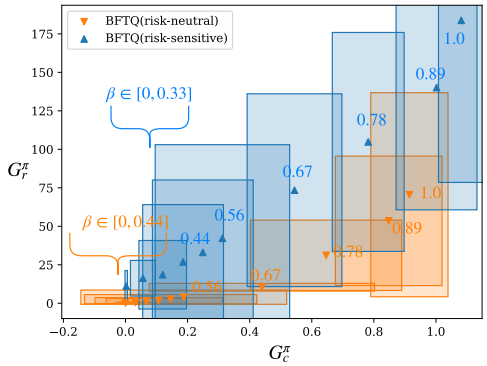
We first bring to light the role of risk-sensitive exploration in the corridors environment. Figure 5.8a shows how the two strategies behave in the corridor environment: the risk-neutral procedure focuses on high-reward corridor only, while the risk-sensitive procedure also explores low-risk trajectories. Videos showing the data collection process are available[4]. In Figure 5.8b, we observe that this better distributed exploration translates as a uniformly better performance across the range $\mathscr{B}$ of risk budgets. When the budget is low, the corresponding optimal budgeted policy $\overline{\pi}^\star$ takes the safest path on the left. When the budget increases, it gradually switches to the other lane, earning higher rewards but also costs. This gradual process could

[4] https://budgeted-rl.github.io/#risk-sensitive-exploration

not be achieved with a deterministic policy as it would choose either one path or the other. Videos illustrating these optimal policies for different level of risks are available [5].



(**a**) State occupations for the two strategies. *Left*: in the risk-sensitive batch, trajectories are well-distributed among both corridors. *Right*: conversely, in the risk-neutral batch, trajectories focus on the risky corridor (to the right) only and ignore the safe corridor (to the left).



(**b**) Performances of the optimal budgeted policy $\overline{\pi}^\star$ trained on batches of transitions obtained by following a risk-neutral and a risk-sensitive exploration. The risk-sensitive procedure attains a better performance across the whole spectrum of risk budgets.

Figure 5.8 – Comparison of two exploration strategies in the corridors environment.

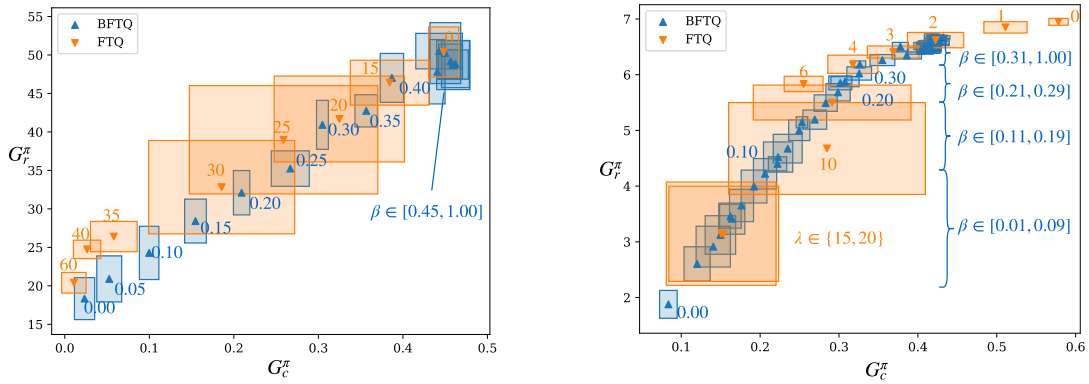


Figure 5.9 – Performance comparison of FTQ(λ) and BFTQ on slot-filling (left) and highway-env(right)

In a second experiment displayed in Figure 5.9, we compare the performance of FTQ(λ) to that of BFTQ in the dialogue and autonomous driving tasks. For each algorithm, we plot the reward-cost trade-off curve. In both cases, BFTQ performs almost as well as FTQ(λ) despite only requiring a single model. All budgets are well-respected on slot-filling, but on highway-env we can observe an underestimation of Q_c, since *e.g.* $\mathbb{E}[G_c|\beta = 0] \simeq 0.1$. This underestimation can be a consequence of two approximations: the use of the sampling operator $\hat{\overline{\mathcal{T}}}$ instead of

[5] https://budgeted-rl.github.io/#optimal-budgeted-policies-learnt-with-a-risk-sensitive-exploration

the true population operator $\mathcal{T}$, and the use of the neural network function approximation $\overline{Q}_\theta$ instead of $\overline{Q}$. Still, BFTQ provides better control over the expected cost of the policy, than FTQ(λ). Besides, BFTQ behaves more consistently than FTQ(λ) overall, as shown by its lower extra-seed variance. Qualitatively, the budgeted agents display a variety of behaviours, shown in several videos[6]. When $\beta = 1$, the ego-vehicle drives in a very aggressive style: it immediately switches to the opposite lane and drives as fast as possible to pass slower vehicles, swiftly changing lanes to avoid incoming traffic. On the contrary, when $\beta = 0$, the ego-vehicle is conservative: it stays on its lane and drives at a low velocity. With intermediate budgets such as $\beta = 0.2$, the agent sometimes decides to overtake its front vehicle but promptly steers back to its original lane afterwards.

Discussion

Algorithm 5.2 is an algorithm for solving large unknown BMDPs with continuous states. To the best of our knowledge, no algorithm in the current literature combines all those features.

Algorithms have been proposed for CMDPs, which are less flexible sub-problems of the more general BMDP. When the environment parameters (P, R, C) are known but not tractable, solutions relying on function approximation (Undurti, Geramifard, and How, 2011) or approximate linear programming (Poupart et al., 2015) have been proposed. For unknown environments, Online algorithms (Geibel and Wysotzki, 2005; Abe et al., 2010; Achiam et al., 2017; Chow, Ghavamzadeh, et al., 2017) and a batch algorithm (Thomas, Theocharous, and Ghavamzadeh, 2015; Ghavamzadeh, Petrik, and Chow, 2016; Laroche, Trichelair, and Combes, 2019; Le, Voloshin, and Yue, 2019) can solve large unknown CMDPs. Nevertheless, these approaches are limited in that the constraints thresholds are fixed before training and cannot be updated in real-time at policy execution to select the desired level of risk.

Budgeted Markov Decision Processes algorithms To our knowledge, there were only two ways of solving a BMDP. The first one is to approximate it with a finite set of CMDPs (*e.g.* see our FTQ(λ) baseline). As explained on Figure 5.6, the optimal deterministic policy can be obtained by a line-search on the Lagrange multiplier values λ. Then, according to Beutler and K. W. Ross (1985, Theorem 4.4), the optimal policy is a randomised mixture of two deterministic policies: the safest deterministic policy that violates the constraint $\pi_{\lambda-}$ and the riskier of the feasible ones $\pi_{\lambda+}$. So FTQ can be easily adapted for continuous states CMDP and BMDP through this methodology, but given the high variance, it requires many simulations to get a proper estimate of the calibration curve. Our solution not only requires one single model but also avoids any supplementary interaction.

[6] https://budgeted-rl.github.io/#driving-styles

The only other existing BMDP algorithm, and closest work to ours, is the DP algorithm proposed by Boutilier and T. Lu (2016). However, their work was established for finite state spaces only, and their solution relies heavily on this property. For instance, they enumerate and sort the next states $s' \in \mathcal{S}$ by their expected value-by-cost, which could not be performed in a continuous state space $\mathcal{S}$. Moreover, they rely on the knowledge of the model (P, R, C), and do not address the question of learning from interaction data.

Chapter Conclusion

Budgeted Markov Decision Processes are a principled framework for safe decision making under uncertainty, which could be beneficial to the diffusion of Reinforcement Learning in industrial applications. They formulate risk as an expected cumulative cost, which can be estimated and controlled in a model-free fashion. However, BMDPs could so far only be solved in finite state spaces which limits their interest for Autonomous Driving applications that require dealing with continuous variables such as vehicle positions. We extend their scope to continuous states by introducing a novel Dynamic Programming operator, that we build upon to propose a Reinforcement Learning algorithm. In order to scale to large problems, we provide an efficient implementation that exploits the structure of the value function and leverages tools from Deep Distributed Reinforcement Learning. We show that on two simulated tasks our solution performs similarly to a baseline Lagrangian relaxation method while only requiring a single model to train, and relying on an interpretable risk budget β instead of the tedious tuning of the penalty λ.

Part Conclusion

Review of our Requirements

Let us come back to the specifications of desirable properties for a behavioural planning algorithm, that we advocated in Chapter 1. In Table 5.4, we examine whether these criteria are met by the methods developed in Part II.

Criterion		Description
Social Awareness	✓	In Chapter 4, we introduced an attention-based Neural Network architecture that explicitly attends to other drivers in the scene, sorting out irrelevant vehicles from those that represent a source of danger.
Sample Efficiency	✓	We showed in Figure 4.4a that this architecture also comes with an inductive bias – permutation invariance – that allows to fasten the training process.
Safety	✓	A first notion of risk was introduced in Chapter 5, in the shape of a cost signal $C(s, a)$ constrained to remain below a threshold β, in expectation. This formulation makes it possible to state safety specifications orthogonal to the traditional reward maximisation objective.
Balance between safety and efficiency	✓	The cost budget β can be adjusted in real time as an input of the budgeted policy $\overline{\pi}$ to trade-off safety with efficiency

Table 5.4 – Do the methods of Part II comply with the specifications of Chapter 1?

Part III

Model-based

. . . et prévoir en stratège.

René Char, *Feuillets d'Hypnos* (72).

Chapter 6

Planning Fast

by Hoping for the Best

Nous voulons, tant ce feu nous brûle le cerveau,
Plonger au fond du gouffre, Enfer ou Ciel, qu'importe ?
Au fond de l'Inconnu pour trouver du nouveau !

Charles Baudelaire, *Le Voyage*.

This third part studies model-based RL algorithms, that estimate the MDP dynamics P so as to *plan* for the corresponding optimal policy. In this chapter, we focus on this planning step under real-time requirements, which calls for provably and empirically sample-efficient algorithms. Since our continuous state space $\mathcal{S}$ precludes the use of Dynamic Programming, we consider tree-based planning algorithms. First, to handle uncertain human behaviours modelled as *stochastic* dynamics, we consider the `OLOP` algorithm, highlight its faulty behaviour and propose a modified version that alleviates this issue. Second, we tackle a paradox: despite the MDP transitions having a graph structure, MCTS algorithms use a tree structure that prevents them from merging similar states. We show that doing so with a graph-based planner better exploits the structure of motion planning problems where trajectories tend to overlap.[1]

Contents

[1]This chapter is based on two articles published in the *2019 European* and *2020 Asian Conferences on Machine Learning* (Leurent and Maillard, 2020b; Leurent and Maillard, 2020a).

6.1 Motivation

In this chapter, we assume that an *estimation oracle* provides us with a good estimate of the MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, and we ponder over the *planning* problem: how to compute the corresponding optimal policy $\pi^\star$? When state-action space $\mathcal{S}\times\mathcal{A}$ is discrete, Dynamic Programming algorithms such as Value Iteration (Bellman, 2010) and Policy Iteration (Howard, 1960) enable to compute an ε-optimal policy with a computational complexity of $\mathcal{O}\left(|\mathcal{S}||\mathcal{A}|\log_\gamma(\varepsilon(1-\gamma))\right)$. However, when $\mathcal{S}$ is continuous or large, exact DP is not feasible. Then, a popular solution to this issue is to perform Approximate Dynamic Programming (ADP), where the value function or policy is approximated within a given hypothesis class, at the cost of the loss of optimality.

Another option is to resort to sampling-based optimisation. This family of methods does not require the full knowledge of the MDP parameters, but rather only assume access to a *generative model* (*e.g.* a simulator) which yields samples of the next state $s' \sim P(s'|s,a)$ and reward $R(s,a)$ when queried. Thus, black-box optimisation algorithms such as `Cross Entropy Method (CEM)` or `Covariance Matrix Adaptation Evolution Strategy (CMA-ES)` can be directly applied in the space of sequences of actions. When the action space $\mathcal{A}$ is discrete[2], the sequential nature of the optimisation problem is better exploited by MCTS algorithms, which leverage the discrete action branching to build a *look-ahead* tree rooted at the current state. This online planning strategy is illustrated in Figure 6.1. At each decision step, a look-ahead tree rooted at the current state is progressively expanded by sampling trajectories through n calls to the generative model, before returning a recommendation for the estimated best action $\hat{a}_n$. The quality of recommended actions is evaluated by their *simple regret*

$$r_n(\hat{a}_n) = V^\star(s) - Q^\star(s,\hat{a}_n).$$

There are two main frames of analysis for planning algorithms.

- In the *fixed-confidence* setting, the generative model is called for a random number of samples n until we can confidently identify a near optimal-action:

  $$\mathbb{P}\left(r_n(\hat{a}_n) \leq \varepsilon\right) \geq 1-\delta.$$

  So-called Probably Approximately Correct (PAC) algorithms verifying this property are then evaluated by their expected sample complexity $\mathbb{E}[n]$.

- In the *fixed-budget* setting, the generative model can only be called a fixed number of times n. Fixed-budget algorithms aim at minimising the expected simple regret $\mathbb{E}[r_n]$.

[2]This requirement has been circumvented by the work of Coulom (2007a), Chaslot et al. (2008), Wang, Audibert, and Rémi Munos (2009), Buşoniu, Daniels, et al. (2013), and Buşoniu, Páll, and Rémi Munos (2018)

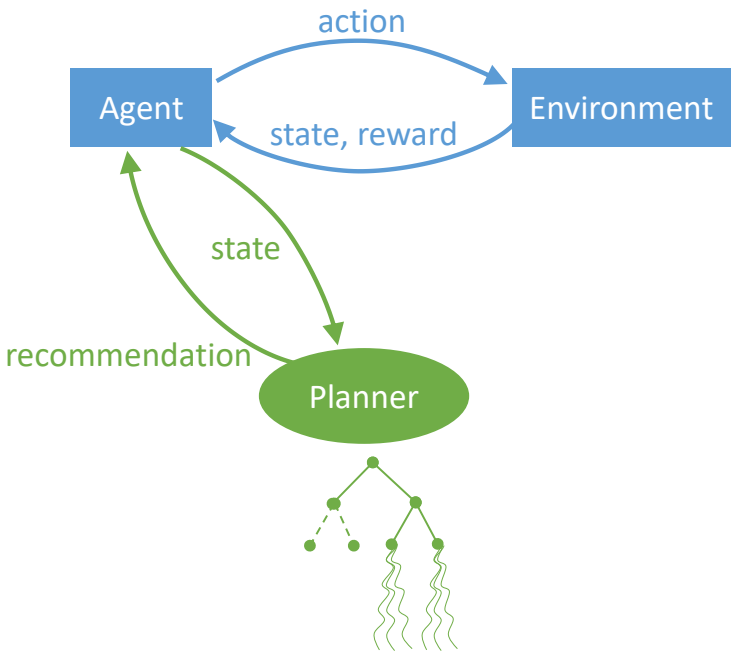


Figure 6.1 – Online planning with a generative model. The true interaction cycle between the agent and the environment is depicted in blue. At each step of real interaction, a full planning cycle of simulated trajectories is run, depicted in green.

In a real-time scenario, decisions need to be taken at a given frequency, even if this means settling for a suboptimal action. Therefore, we will focus on the fixed-budget setting, whose bounded computational complexity makes it more appropriate for our application.

MCTS algorithms were a breakthrough for online decision-making in MDPs, that lead to key successes in the domain, including Computer Go (Coulom, 2007b; Silver, Hubert, et al., 2018). They enjoy two main benefits: first, they do not require the knowledge of the MDP parameters contrary to *e.g.* DP algorithms, but only the access to a *generative model* that allows sampling trajectories from the current state. Second, the theoretical performance bounds of MCTS algorithms are typically independent of the size of the state space $\mathcal{S}$. Instead, they depend on the maximum depth at which an optimal node in the search tree can be reached within the allowed budget n of trajectory samples. This translates as an *effective branching factor* in the regret bounds, related to the notion of near-optimality dimension in multi-armed bandits.

Related work Algorithms for planning with a generative model date back at least to the seminal work of Kearns, Mansour, and Ng (2002) who proposed the `Sparse Sampling` algorithm using a tree structure to represent the value estimate and uniform sampling of trajectories. This strategy was further analysed more recently in (Feldman and Domshlak, 2014), where the `Best Recommendation with Uniform Estimation (BRUE)` algorithm provides an

enhanced value estimation. Another family of algorithms rely on the principle of Optimism in the Face of Uncertainty (OFU) (surveyed by Rémi Munos, 2014), inspired by the Multi-Armed Bandits (MAB) problem. This principle was first used in the context of planning in the `CrazyStone` software (Coulom, 2007b) for computer Go. It was later formalised with the `Upper Confidence bounds applied to Trees (UCT)` algorithm (Kocsis and Szepesvári, 2006), but was shown by Coquelin and Rémi Munos (2007) to have a doubly-exponential complexity in the worst case. The `Optimistic Planning of Deterministic Systems (OPD)` algorithm introduced by Hren and Rémi Munos (2008) was the first to provide a polynomial regret bound, but was limited to systems with deterministic rewards and dynamics. It was then extended to stochastic rewards and dynamics with the `Open-Loop Optimistic Planning (OLOP)` algorithm (Sébastien Bubeck and Rémi Munos, 2010), but only in the *open-loop* setting of state-independent policies (*i.e.* sequences of actions), a restriction of the policy class that causes a loss of optimality. Known stochastic transitions were handled by Buşoniu and Remi Munos (2012). For MDPs with stochastic and unknown transitions, polynomial sample complexities have been obtained for `Stochastic Optimistic Planning (StOP)` (Szörényi, Kedenburg, and Remi Munos, 2014), `TrailBlazer` (Grill, Valko, and Remi Munos, 2016) and `SmoothCruiser` (Grill, Darwiche Domingues, et al., 2019), but despite their theoretical merits these algorithms are intractable in practice: `StOP` requires the expensive storage of policies, while `TrailBlazer` and `SmoothCruiser` only terminate after a prohibitive amount of samples, even for very small MDPs.

Contributions This chapter focuses on two questions. First, in Section 6.2, we exhibit a faulty behaviour of the `OLOP` algorithm when applied to numerical problems, and propose a modified version that addresses it, leading to improved performance with a retained guarantee. Second, in Section 6.3 we look into how MCTS can benefit from merging overlapping trajectories, in order to better exploit the underlying graphical structure of the dynamics.

6.2 Open-loop optimistic planning

The goal of this section is to study the empirical performance of the `OLOP` algorithm. We focus on that algorithm for its ability to tractably handle stochastic dynamics (though in open-loop only). Indeed, our MDP formulation of Chapter 3 involves unknown parameters in the dynamics, such as the driving styles and destinations of other drivers. Thus, in a probabilistic modelling of this uncertainty, a prior distribution over these unknown parameters induces a distribution over the next state of each observed vehicle, *i.e.* stochastic dynamics. However, `OLOP` was introduced with a theoretical sample complexity analysis, but no experiment was carried out. We show in our experiments that this algorithm is overly pessimistic, especially in the low-budget regime, and we provide an intuitive explanation by casting light on an unintended effect that alters its

behaviour. We circumvent this issue by leveraging modern tools from the MAB literature to design and analyse a modified version with tighter upper-confidence bounds called KL-OLOP. We show that we retain the asymptotic regret bounds of OLOP while improving its performances by an order of magnitude in numerical experiments.

This section is structured as follows: in Section 6.2.1, we present OLOP, give some intuition on its limitations, and introduce KL-OLOP, whose sample complexity is further analysed in Sections 6.2.2 and 6.2.3, and whose empirical performance is evaluated in Section 6.2.4 in several numerical experiments.

Notations We follow the notations from (Sébastien Bubeck and Rémi Munos, 2010) and use the standard notations over alphabets, as described in the List of Symbols.

During the planning process, the agent iteratively selects sequences of actions until it reaches the allowed budget of n actions. More precisely, at time t during the m^{th} sequence, the agent played $a_{1:t}^m = a_1^m \cdots a_t^m \in \mathcal{A}^t$ and receives a reward $R(s_t^m)$. We denote the probability distribution of this reward as $\nu(a_{1:t}^m) = \mathbb{P}\left(R(s_t^m, a_t^m)|s_t^m, a_t^m\right) \prod_{k=1}^{t-1} \mathbb{P}\left(s_{k+1}^m|s_k^m, a_k^m\right)$, and its mean as $\mu(a_{1:t}^m)$, where $s_1^m = s_1$ is the current state.

Definition 6.1 (Sequence values)**.** *We define the value $V(a)$ of a sequence of actions $a \in \mathcal{A}^h$ as the maximum expected discounted cumulative reward one may obtain after executing a:*

$$V(a) = \sup_{b \in a\mathcal{A}^\infty} \sum_{t=1}^{\infty} \gamma^t \mu(b_{1:t}). \tag{6.1}$$

6.2.1 Kullback-Leibler Open-Loop Optimistic Planning

We present KL-OLOP, a combination of the OLOP algorithm of (Sébastien Bubeck and Rémi Munos, 2010) with the tighter Kullback-Leibler upper confidence bounds from (Olivier Cappé et al., 2013). We first frame both algorithms in a common structure before specifying their implementations.

General structure First, following OLOP, the total sample budget n is split in M trajectories of length L in the following way:

$$\begin{aligned} &M \text{ is the largest integer such that } M\lceil \log M/(2\log 1/\gamma)\rceil \leq n;\\ &L = \lceil \log M/(2\log 1/\gamma)\rceil. \end{aligned}$$

The look-ahead tree of depth L is denoted $\mathcal{T} = \sum_{h=0}^{L} \mathcal{A}^h$.

Then, we introduce some useful definitions. Consider episode $1 \leq m \leq M$. For any $1 \leq h \leq L$ and $a \in \mathcal{A}^h$, let

$$N_a(m) \triangleq \sum_{i=1}^{m} \mathbb{1}\{a^i_{1:h} = a\}$$

be the number of times we played an action sequence starting with a, and $S_a(m)$ the sum of rewards collected at the last transition of the sequence a

$$S_a(m) \triangleq \sum_{i=1}^{m} R(s^i_h, a^i_{1:h}) \mathbb{1}\{a^i_{1:h} = a\}.$$

The empirical mean reward of a is $\hat{\mu}_a(m) \triangleq \dfrac{S_a(m)}{N_a(m)}$ if $N_a(m) > 0$, and $+\infty$ otherwise. Here, we provide a more general form for upper and lower confidence bounds on these empirical means:

$$U^{\mu}_a(m) \triangleq \max \{q \in I : N_a(m) d(\hat{\mu}_a(m), q) \leq f(m)\} \tag{6.2}$$

$$L^{\mu}_a(m) \triangleq \min \{q \in I : N_a(m) d(\hat{\mu}_a(m), q) \leq f(m)\} \tag{6.3}$$

where I is an interval, d is a divergence on $I \times I \to \mathbb{R}^+$ and f is a non-decreasing function. They are left unspecified for now and their particular implementations and associated properties will be discussed in the following sections.

These upper-bounds U^{μ}_a for intermediate rewards finally enable us to define an upper bound U_a for the value $V(a)$ of the entire sequence of actions a:

$$U_a(m) \triangleq \sum_{t=1}^{h} \gamma^t U^{\mu}_{a_{1:t}}(m) + \frac{\gamma^{h+1}}{1-\gamma}. \tag{6.4}$$

where $\frac{\gamma^{h+1}}{1-\gamma}$ comes from upper-bounding by one every reward-to-go in the sum (6.1), for $t \geq h+1$. In (Sébastien Bubeck and Rémi Munos, 2010), there is an extra step to "sharpen the bounds" of sequences $a \in \mathcal{A}^L$ by taking

$$B_a(m) \triangleq \inf_{1 \leq t \leq L} U_{a_{1:t}}(m) \tag{6.5}$$

The general algorithm structure is shown in Algorithm 6.1. We now discuss two specific implementations that differ in their choice of divergence d and non-decreasing function f. They are compared in Table 6.1.

Algorithm 6.1: General structure for Open-Loop Optimistic Planning

for *each episode* $m = 1, \cdots, M$ **do**
  Compute $U_a(m-1)$ from (6.4) for all $a \in \mathcal{T}$
  Compute $B_a(m-1)$ from (6.5) for all $a \in \mathcal{A}^L$
  Sample a sequence with highest B-value: $a^m \in \arg\max_{a \in \mathcal{A}^L} B_a(m-1)$
return the most played sequence $a(n) \in \arg\max_{a \in \mathcal{A}^L} N_a(m)$

Table 6.1 – Different implementations of Algorithm 6.1 in `OLOP` and `KL-OLOP`

Algorithm	`OLOP`	`KL-OLOP`
Interval I	$\mathbb{R}$	$[0, 1]$
Divergence d	$d_{\texttt{QUAD}}$	$d_{\texttt{BER}}$
$f(m)$	$4 \log M$	$2 \log M + 2 \log \log M$

`OLOP`

To recover the original `OLOP` algorithm of Sébastien Bubeck and Rémi Munos (2010) from Algorithm 6.1, we can use a quadratic divergence $d_{\texttt{QUAD}}$ on $I = \mathbb{R}$ and a constant function f_4 defined as follows:

$$d_{\texttt{QUAD}}(p, q) \triangleq 2(p-q)^2, \qquad f_4(m) \triangleq 4 \log M$$

Indeed, in this case $U_a^\mu(m)$ can then be explicitly computed as

$$U_a^\mu(m) = \max \left\{ q \in \mathbb{R} : 2(\hat{\mu}_a(m) - q)^2 \leq \frac{4 \log M}{N_a(m)} \right\} = \hat{\mu}_a(m) + \sqrt{\frac{2 \log M}{N_a(m)}},$$

which is the Chernoff-Hoeffding bound used originally in section 3.1 of (Sébastien Bubeck and Rémi Munos, 2010).

An unintended behaviour

From the definition of $U_a(m)$ as an upper-bound of the value of the sequence a, we expect increasing sequences $(a_{1:t})_t$ to have non-increasing upper-bounds. Indeed, every new action a_t encountered along the sequence is a potential loss of optimality. However, this property is only true if the upper-bound defined in (6.2) belongs to the reward interval $[0, 1]$.

Lemma 6.2 (Monotony of $U_a(m)$ along a sequence)**.**

- *If it holds that* $U_b^\mu(m) \in [0, 1]$ *for all* $b \in \mathcal{A}^*$*, then for any* $a \in \mathcal{A}^L$ *the sequence* $(U_{a_{1:h}}(m))_{1 \leq h \leq L}$ *is non-increasing, and we simply have* $B_a(m) = U_a(m)$*.*

- *Conversely, if $U_b^\mu(m) > 1$ for all $b \in \mathcal{A}^*$, then for any $a \in \mathcal{A}^L$ the sequence $(U_{a_{1:h}}(m))_{1\leq h\leq L}$ is non-decreasing, and we have $B_a(m) = U_{a_{1:1}}(m)$.*

Proof. We prove the first proposition, and the same reasoning applies to the second. For $a \in \mathcal{A}^L$ and $1 \leq h \leq L-1$, we have by (6.4):

$$\begin{aligned} U_{a_{1:h+1}}(m) - U_{a_{1:h}}(m) &= \gamma^{h+1} U^\mu_{a_{1:h+1}}(m) + \frac{\gamma^{h+2}}{1-\gamma} - \frac{\gamma^{h+1}}{1-\gamma} \\ &= \gamma^{h+1}(\underbrace{U^\mu_{a_{1:h+1}}(m)}_{\in[0,1]} - 1) \leq 0 \end{aligned}$$

We can conclude that $(U_{a_{1:h}}(m))_{1\leq h\leq L}$ is non-increasing and that $B_a(m) = \inf_{1\leq h\leq L} U_{a_{1:h}}(m) = U_{a_{1:L}}(m) = U_a(m)$. □

Yet, the Chernoff-Hoeffding bounds used in `OLOP` start in the $U_a^\mu(m) > 1$ regime – initially $U_a^\mu(m) = \infty$ – and can remain in this regime for a long time especially in the near-optimal branches where $\hat{\mu}_a(m)$ is close to one.

Under these circumstances, the Lemma 6.2 has a drastic effect on the search behaviour. Indeed, as long as a subtree under the root verifies $U_a^\mu(m) > 1$ for every sequence a, then all these sequences share the same B-value $B_a(m) = U_{a_{1:1}(m)}$. This means that `OLOP` cannot differentiate them and exploit information from their shared history as intended, and behaves as uniform sampling instead. Once the early depths have been explored sufficiently, `OLOP` resumes its intended behaviour, but the problem is only shifted to deeper unexplored subtrees.

This consideration motivates us to leverage the recent developments in the Multi-Armed Bandits literature, and modify the upper-confidence bounds for the expected rewards $U_a^\mu(m)$ so that they respect the reward bounds.

`KL-OLOP`

We propose a novel implementation of Algorithm 6.1 where we leverage the analysis of the kl-UCB algorithm from (Olivier Cappé et al., 2013) for multi-armed bandits with general bounded rewards. Likewise, we use the Bernoulli Kullback-Leibler divergence defined on the interval $I = [0, 1]$ by

$$d_{\text{BER}}(p, q) \triangleq p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$$

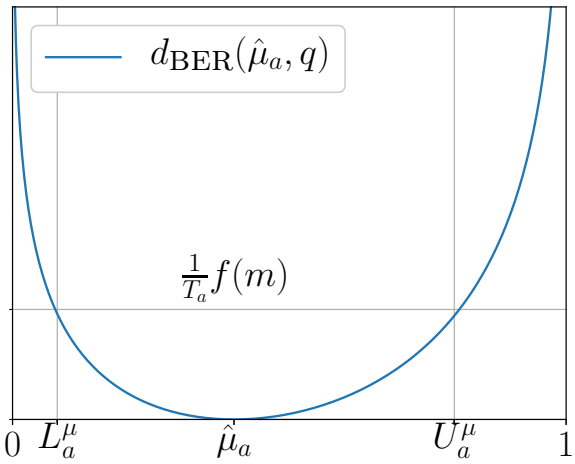


Figure 6.2 – The Bernoulli Kullback-Leibler divergence $d_{\mathtt{BER}}$, and the corresponding upper and lower confidence bounds U_a^μ and L_a^μ for the empirical average $\hat{\mu_a}$. Lower values of $f(m)$ give tighter confidence bounds that hold with lower probabilities.

with, by convention, $0 \log 0 = 0 \log 0/0 = 0$ and $x \log x/0 = +\infty$ for $x > 0$. This divergence and the corresponding bounds are illustrated in Figure 6.2.

$U_a^\mu(m)$ and $L_a^\mu(m)$ can be efficiently computed using Newton iterations, as for any $p \in [0, 1]$ the function $q \to d_{\mathtt{BER}}(p, q)$ is strictly convex and increasing (resp. decreasing) on the interval [p, 1] (resp. [0, p]).

Moreover, we use the constant function $f_2 : m \to 2 \log M + 2 \log \log M$. This choice is justified in the end of Section 6.2.3. Because f_2 is lower than f_4, the Figure 6.2 shows that the bounds are tighter and hence less conservative than that of `OLOP`, which should increase the performance, provided that their associated probability of violation does not invalidate the regret bound of `OLOP`.

> **Remark 6.3** (Upper bounds sharpening). *The introduction of the B-values $B_a(m)$ was made necessary in `OLOP` by the use of Chernoff-Hoeffding confidence bounds which are not guaranteed to belong to $[0, 1]$. On the contrary, we have in `KL-OLOP` that $U_a^\mu(m) \in I = [0, 1]$ by construction. By Lemma 6.2, the upper bounds sharpening step in line 3 of Algorithm 6.1 is now superfluous as we trivially have $B_a(m) = U_a(m)$ for all $a \in \mathcal{A}^L$.*

6.2.2 Sample complexity

We say that $u_n = \tilde{\mathcal{O}}(v_n)$ if there exist $\alpha, \beta > 0$ such that $u_n \leq \alpha \log(v_n)^\beta v_n$. Let us denote the proportion of near-optimal nodes κ_2 as

$$\kappa_2 \triangleq \limsup_{h\to\infty} \left| \left\{ a \in a^H : V(a) \geq V - 2\frac{\gamma^{h+1}}{1-\gamma} \right\} \right|^{1/h}$$

Theorem 6.4 (Sample complexity). *We show that* KL-OLOP *enjoys the same asymptotic regret bounds as* OLOP*. More precisely, for any $\kappa' > \kappa_2$,* KL-OLOP *satisfies:*

$$\mathbb{E}[r_n] = \begin{cases} \tilde{\mathcal{O}}\left(n^{-\frac{\log 1/\gamma}{\log \kappa'}}\right) & \text{if } \gamma\sqrt{\kappa'} > 1; \\ \tilde{\mathcal{O}}\left(n^{-\frac{1}{2}}\right) & \text{if } \gamma\sqrt{\kappa'} \leq 1. \end{cases}$$

Proof. We provide the proof in Section 6.2.3. □

We provide a time and memory efficient implementation of OLOP and KL-OLOP in Section C.2.1, bringing an exponential speed-up that allows scaling these algorithms to high sample budgets.

6.2.3 Proof of Theorem 6.4

We follow step-by-step the pyramidal proof of (Sébastien Bubeck and Rémi Munos, 2010), and adapt it to the Kullback-Leibler upper confidence bound. The adjustments resulting from the change of confidence bounds are highlighted. The proofs of lemmas which are not significantly altered are listed in Section C.1.

We start by recalling their notations. Let $1 \leq H \leq L$ and $a^\star \in \mathcal{A}^L$ such that $V(a^\star) = V$. Considering sequences of actions of length $1 \leq h \leq H$, we define the subset $\mathcal{I}_h$ of near-optimal sequences and the subset $\mathcal{J}$ of sub-optimal sequences that were near-optimal at depth $h-1$

$$\mathcal{I}_h = \left\{ a \in \mathcal{A}^h : V - V(a) \leq 2\frac{\gamma^{h+1}}{1-\gamma} \right\}, \mathcal{J}_h = \left\{ a \in \mathcal{A}^h : a_{1:h-1} \in \mathcal{I}_{h-1} \text{ and } a \notin \mathcal{I}_h \right\}$$

By convention, $\mathcal{I}_0 = \{\emptyset\}$. From the definition of κ_2, we have that for any $\kappa' > \kappa_2$, there exists a constant C such that for any $h \geq 1$, $|\mathcal{I}_h| \leq C{\kappa'}^h$ Hence, we also have $|\mathcal{J}_h| \leq K|\mathcal{I}_{h-1}| = O({\kappa'}^h)$.

Now, for $1 \leq m \leq M$, $a \in \mathcal{A}^t$ with $t \leq h$, $h' < h$, we define the set $\mathcal{P}^a_{h,h'}(m)$ of suffixes of a in $\mathcal{J}_h$ that have been played at least a certain number of times

$$\mathcal{P}^a_{h,h'}(m) = \left\{ b \in a\mathcal{A}^{h-t} \cap \mathcal{J}_h : N_b(m) \geq 2f(m)(h+1)^2\gamma^{2(h'-h+1)} + 1 \right\}$$

and the random variable

$$\tau^a_{h,h'}(m) = \mathbb{1}\{N_a(m-1) < 2f(m)(h+1)^2\gamma^{2(h'-h+1)} + 1 \leq N_a(m)\}$$

Lemma 6.5 (Regret and sub-optimal pulls)**.** *The following holds true:*

$$\mathbb{E}[r_n] \leq \frac{2K\gamma^{H+1}}{1-\gamma} + \frac{3K}{M}\sum_{h=1}^{H}\sum_{a\in\mathcal{J}_h}\frac{\gamma^h}{1-\gamma}N_a(m)$$

Proof. We provide the proof in Section C.1.1. □

The rest of the proof is devoted to the analysis of the term $\mathbb{E}\sum_{a\in\mathcal{J}_h} N_a(m)$. The next lemma describes under which circumstances a suboptimal sequence of actions in $\mathcal{J}_h$ can be selected.

Lemma 6.6 (Conditions for sub-optimal pull)**.** *Assume that at step $m+1$ we select a sub-optimal sequence a^{m+1}: there exist $0 \leq h \leq L, a \in \mathcal{J}_h$ such that $a^{m+1} \in a\mathcal{A}^*$. Then, it implies that one of the following propositions is true:*

$$U_{a^\star}(m) < V, \qquad \text{(UCB violation)}$$

or

$$\sum_{t=1}^{h}\gamma^t L^\mu_{a_{1:t}}(m) \geq V(a), \qquad \text{(LCB violation)}$$

or

$$\sum_{t=1}^{h}\gamma^t(U^\mu_{a_{1:t}}(m) - L^\mu_{a_{1:t}}(m)) > \frac{\gamma^{h+1}}{1-\gamma} \qquad \text{(Large CI)}$$

Proof. As $a^{m+1}_{1:h} = a$ and because the U-values are monotonically increasing along sequences of actions (see Remark 6.3 and Lemma 6.2), we have $U_a(m) \geq U_{a^{m+1}}(m)$. Moreover, by Algorithm 6.1, we have $a^{m+1} = \arg\max_{a\in\mathcal{A}^L} U_a(m)$ and $a^\star \in \mathcal{A}^L$, so $U_{a^{m+1}}(m) \geq U_{a^\star}(m)$ and finally $U_a(m) \geq U_{a^\star}(m)$.

Assume that (UCB violation) is false, then

$$\sum_{t=1}^{h}\gamma^t U^\mu_{a_{1:t}}(m) + \frac{\gamma^{h+1}}{1-\gamma} = U_a(m) \geq U_{a^\star}(m) \geq V \tag{6.6}$$

Assume that (LCB violation) is false, then

$$\sum_{t=1}^{h} \gamma^t L^{\mu}_{a_{1:t}}(m) < V(a), \tag{6.7}$$

By taking the difference (6.6) - (6.7),

$$\sum_{t=1}^{h} \gamma^t \left(U^{\mu}_{a_{1:t}}(m) - L^{\mu}_{a_{1:t}}(m)\right) + \frac{\gamma^{h+1}}{1-\gamma} > V - V(a)$$

But $a \in \mathcal{J}_h$, so $V - V(a) \geq \frac{2\gamma^{h+1}}{1-\gamma}$, which yields (Large CI) and concludes the proof. □

In the following lemma, for each episode m we bound the probability of (UCB violation) or (LCB violation) by a desired confidence level δ_m, whose choice we postpone until the end of this proof. For now, we simply assume that we picked a function f that satisfies $f(m)\log(m)e^{-f(m)} = O(\delta_m)$. We also denote $\Delta_M = \sum_{m=1}^{M} \delta_m$.

Lemma 6.7 (Boundary crossing probability)**.** *The following holds true, for any $1 \leq h \leq L$ and $m \leq M$,*

$$\mathbb{P}\left((\text{UCB violation}) \text{ or } (\text{LCB violation}) \text{ is true}\right) = O((L+h)\delta_m)$$

Proof. Since $V \leq \sum_{t=1}^{h} \gamma^t \mu(a^\star_{1:t}) + \frac{\gamma^{h+1}}{1-\gamma}$, we have,

$$\begin{aligned}
\mathbb{P}\left((\text{UCB violation})\right) &= \mathbb{P}\left(U_{a^\star}(m) \leq V\right) \\
&= \mathbb{P}\left(\sum_{t=1}^{L} \gamma^t U^{\mu}_{a^\star_{1:t}}(m) \leq \sum_{t=1}^{L} \gamma^t \mu(a^\star_{1:t})\right) \\
&\leq \mathbb{P}\left(\exists 1 \leq t \leq L : U^{\mu}_{a^\star_{1:t}}(m) \leq \mu(a^\star_{1:t})\right) \\
&\leq \sum_{t=1}^{L} \mathbb{P}\left(U^{\mu}_{a^\star_{1:t}}(m) \leq \mu(a^\star_{1:t})\right)
\end{aligned}$$

In order to bound this quantity, we reduce the question to the application of a deviation inequality. For all $1 \leq t \leq L$, we have on the event $\{U^{\mu}_{a^\star_{1:t}}(m) \leq \mu(a^\star_{1:t})\}$ that $\hat{\mu}_{a^\star_{1:t}}(m) \leq U^{\mu}_{a^\star_{1:t}}(m) \leq \mu(a^\star_{1:t}) < 1$. Therefore, for all $0 < \delta < 1 - \mu(a^\star_{1:t})$, by definition of $U^{\mu}_{a^\star_{1:t}}(m)$:

$$d(\hat{\mu}_{a^\star_{1:t}}(m), U^{\mu}_{a^\star_{1:t}}(m) + \delta) > \frac{f(m)}{N_{a^\star_{1:t}}(m)}$$

As d is continuous on $(0,1) \times [0,1]$, we have by letting $\delta \to 0$ that:

$$d(\hat{\mu}_{a^\star_{1:t}}(m), U^\mu_{a^\star_{1:t}}(m)) \geq \frac{f(m)}{N_{a^\star_{1:t}}(m)}$$

Since d is non-decreasing on $[\hat{\mu}_{a^\star_{1:t}}(m), \mu(a^\star_{1:t})]$,

$$d(\hat{\mu}_{a^\star_{1:t}}(m), \mu(a^\star_{1:t})) \geq d(\hat{\mu}_{a^\star_{1:t}}(m), U^\mu_{a^\star_{1:t}}(m)) \geq \frac{f(m)}{N_{a^\star_{1:t}}(m)}$$

We have thus shown the following inclusion:

$$\{U^\mu_{a^\star_{1:t}}(m) \leq \mu(a^\star_{1:t})\} \subseteq \left\{\mu(a^\star_{1:t}) > \hat{\mu}_{a^\star_{1:t}}(m) \text{ and } d(\hat{\mu}_{a^\star_{1:t}}(m), \mu(a^\star_{1:t})) \geq \frac{f(m)}{N_{a^\star_{1:t}}(m)}\right\}$$

Decomposing according to the values of $N_{a^\star_{1:t}}(m)$ yields:

$$\{U^\mu_{a^\star_{1:t}}(m) \leq \mu(a^\star_{1:t})\} \subseteq \bigcup_{n=1}^{m} \left\{\mu(a^\star_{1:t}) > \hat{\mu}_{a^\star_{1:t},n} \text{ and } d(\hat{\mu}_{a^\star_{1:t},n}, \mu(a^\star_{1:t})) \geq \frac{f(m)}{n}\right\}$$

We now apply the deviation inequality provided in Lemma 2 of Appendix A in (Olivier Cappé et al., 2013): $\forall \varepsilon > 1$, provided that $0 < \mu(a^\star_{1:t}) < 1$,

$$\mathbb{P}\left(\bigcup_{n=1}^{m} \left\{\mu(a^\star_{1:t}) > \hat{\mu}_{a^\star_{1:t},n} \text{ and } n d_{\texttt{BER}}(\hat{\mu}_{a^\star_{1:t},n}, \mu(a^\star_{1:t})) \geq \varepsilon\right\}\right) \leq e\lceil \varepsilon \log m \rceil e^{-\varepsilon}\,.$$

By choosing $\varepsilon = f(m)$, it comes

$$\mathbb{P}\left((\text{UCB violation})\right) \leq \sum_{t=1}^{L} e\lceil f(m) \log m \rceil e^{-f(m)} = O(L\delta_m)$$

The same reasoning gives: $\mathbb{P}\left((\text{LCB violation})\right) = O(h\delta_m)$. □

Lemma 6.8 (Confidence interval length and number of plays). *Let $1 \leq h \leq L$, $a \in \mathcal{J}_h$ and $0 \leq h' < h$. Then (Large CI) is not satisfied if the following propositions are true:*

$$\forall 0 \leq t \leq h', N_{a_{1:t}}(m) \geq 2f(m)(h+1)^2\gamma^{2(t-h-1)} \tag{6.8}$$

and

$$N_a(m) \geq 2f(m)(h+1)^2\gamma^{2(h'-h-1)} \tag{6.9}$$

Proof. We start by providing an explicit upper-bound for the length of the confidence interval $U^{\mu}_{a_{1:t}} - L^{\mu}_{a_{1:t}}$. By Pinsker's inequality:

$$d_{\texttt{BER}}(p,q) > d_{\texttt{QUAD}}(p,q)$$

Hence for all $C > 0$,

$$d_{\texttt{BER}}(p,q) \leq C \implies 2(q-p)^2 < C \implies p - \sqrt{C/2} < q < p + \sqrt{C/2}$$

And thus, for all $b \in \mathcal{A}^*$, by definition of U^{μ} and L^{μ}:

$$U^{\mu}_b(m) - L^{\mu}_b(m) \leq \frac{S_b(m)}{N_b(m)} + \sqrt{\frac{f(m)}{2N_b(m)}} - \left(\frac{S_b(m)}{N_b(m)} - \sqrt{\frac{f(m)}{2N_b(m)}}\right) = \sqrt{\frac{2f(m)}{N_b(m)}}$$

Now, assume that (6.8) and (6.9) are true. Then, we clearly have

$$\begin{aligned}
\sum_{t=1}^{h} \gamma^t \left(U^{\mu}_{a_{1:t}}(m) - L^{\mu}_{a_{1:t}}(m)\right) &\leq \sum_{t=1}^{h'} \gamma^t \sqrt{\frac{2f(m)}{N_{a_{1:t}}(m)}} + \sum_{t=h'+1}^{h} \gamma^t \sqrt{\frac{2f(m)}{N_{a_{1:t}}(m)}} \\
&\leq \frac{1}{(h+1)\gamma^{-h-1}} \sum_{t=1}^{h'} 1 + \frac{1}{(h+1)\gamma^{-h-1}} \sum_{t=h'+1}^{h} \gamma^{t-h'} \\
&\leq \frac{\gamma^{h+1}}{h+1}\left(h' + \frac{\gamma}{1-\gamma}\right) \leq \frac{\gamma^{h+1}}{1-\gamma}\,.
\end{aligned}$$

□

Lemma 6.9. *Let $1 \leq h \leq L$, $a \in \mathcal{J}_h$ and $0 \leq h' < h$. Then $\tau^{a}_{h,h'} = 1$ implies that either equation (UCB violation) or (LCB violation) is satisfied or the following proposition is true:*

$$\exists 1 \leq t \leq h' : |\mathcal{P}^{a_{1:t}}_{h,h'}(m)| < \gamma^{2(t-h')} \tag{6.10}$$

Proof. We provide the proof in Section C.1.2. □

Lemma 6.10. *Let $1 \leq h \leq L$ and $0 \leq h' < h$. Then the following holds true,*

$$\mathbb{E}\,|\mathcal{P}^{\emptyset}_{h,h'}(M)| = \tilde{O}\left(\gamma^{-2h'}\mathbb{1}_{h'>0}\sum_{t=0}^{h'}(\gamma^2\kappa')^t + (\kappa')^h\Delta_M\right).$$

Proof. We provide the proof in Section C.1.3. □

Lemma 6.11. *Let $1 \leq h \leq L$. The following holds true,*

$$\mathbb{E} \sum_{a \in \mathcal{J}_h} N_a(m) = \tilde{O}\left(\gamma^{-2h} + (\kappa')^h(1 + M\Delta_M + \Delta_M) + (\kappa'\gamma^{-2})^h \Delta_M\right)$$

Proof. We provide the proof in Section C.1.4. □

Thus by combining Lemmas 6.5 and 6.11 we obtain

$$\mathbb{E}[r_n] = \tilde{O}\left(\gamma^H + \gamma^{-H}M^{-1} + (\kappa'\gamma)^H M^{-1}(1 + M\Delta_M + \Delta_M) + (\kappa')^H\gamma^{-H}M^{-1}\Delta_M\right)$$

Finally,

- if $\kappa'\gamma^2 \leq 1$, we take $H = \lfloor \log M/(2\log 1/\gamma) \rfloor$ to obtain

  $$\mathbb{E}[r_n] = \tilde{O}\left(M^{-\frac{1}{2}} + M^{-\frac{1}{2}} + M^{-\frac{1}{2}} M^{\frac{\log \kappa'}{2\log 1/\gamma}} \Delta_M\right)$$

  For the last term to be of the same order of the others, we need to have $\Delta_M = O(M^{-\frac{\log \kappa'}{2\log 1/\gamma}})$. Since $\kappa'\gamma^2 \leq 1$, we achieve this by taking $\Delta_M = O(M^{-1})$.

- if $\kappa'\gamma^2 > 1$, we take $H = \lfloor \log M/\log \kappa' \rfloor$ to obtain

  $$\mathbb{E}[r_n] = \tilde{O}\left(M^{\frac{\log \gamma}{\log \kappa'}} + M^{\frac{\log \gamma}{\log \kappa'}}(1 + M\Delta_M + \Delta_M) + M^{\frac{\log 1/\gamma}{\log \kappa'}} \Delta_M\right)$$

  Since $\kappa'\gamma^2 > 1$, the dominant term in this sum is $M^{\frac{\log \gamma}{\log \kappa'}} M\Delta_M$. Again, taking $\Delta_M = O(M^{-1})$ yields the claimed bounds.

Thus, the claimed bounds are obtained in both cases as long as we can impose $\Delta_M = O(M^{-1})$, that is, find a sequence $(\delta_m)_{1 \leq m \leq M}$ and a function f verifying:

$$\sum_{m=1}^{M} \delta_m = O(M^{-1}) \quad \text{and} \quad f(m)\log(m)e^{-f(m)} = O(\delta_m) \tag{6.11}$$

By choosing $\delta_m = M^{-2}$ and $f(m) = 2\log M + 2\log\log M$, the corresponding `KL-OLOP` algorithm does achieve the regret bound claimed in Theorem 6.4.

6.2.4 Experiments

We have performed some numerical experiments to evaluate and compare the following planning algorithms[1]:

- `Random`: returns a random action, we use it as a minimal performance baseline.
- `OPD`: the *Optimistic Planning for Deterministic systems* from (Hren and Rémi Munos, 2008), used as a baseline of optimal performance. This planner is only suited for deterministic environments, and exploits this property to obtain faster rates. However, it is expected to fail in stochastic environments.
- `OLOP`: as described in Section 6.2.1.[2]
- `KL-OLOP`: as described in Section 6.2.1.[2]
- `KL-OLOP` (1): an aggressive version of `KL-OLOP` where we used $f_1(m) = \log M$ instead of $f_2(m)$. This threshold function makes the upper bounds even tighter, at the cost of an increased probability of violation. Hence, we expect this solution to be more efficient in close-to-deterministic environments. However, since we have no theoretical guarantee concerning its regret as we do with `KL-OLOP`, it might not be conservative enough and converge too early to a suboptimal sequence, especially in highly stochastic environments.

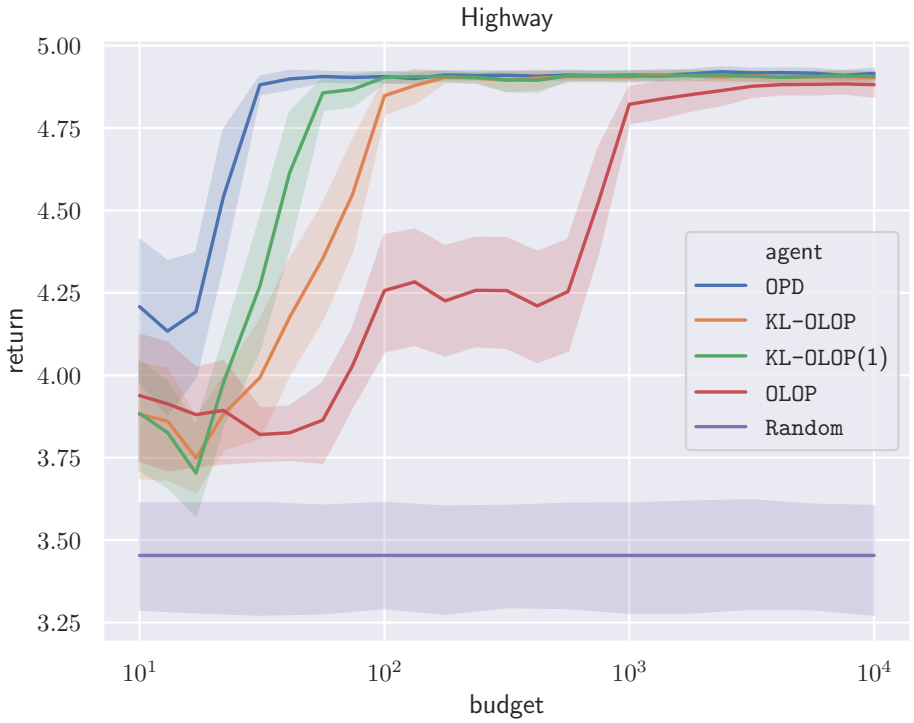
They are evaluated on the following tasks, using a discount factor of $\gamma = 0.8$:

- A highway driving environment (Leurent, 2018): a vehicle is driving on a road randomly populated with other slower drivers, and must make their way as fast as possible while avoiding collisions by choosing on the the following actions: `change-lane-left`, `change-lane-right`, `no-op`, `faster`, `slower`.
- A gridworld environment (Chevalier-Boisvert, Willems, and Pal, 2018): the agent navigates in a randomly-generated gridworld composed of either empty cells, terminal lava cells, and goal cells where a reward of 1 is collected at the first visit.
- A stochastic version of the gridworld environment with noisy rewards, where the noise is modelled as a Bernoulli distribution with a 15% probability of error, *i.e.* receiving a reward of 1 in an empty cell or 0 in a goal cell.

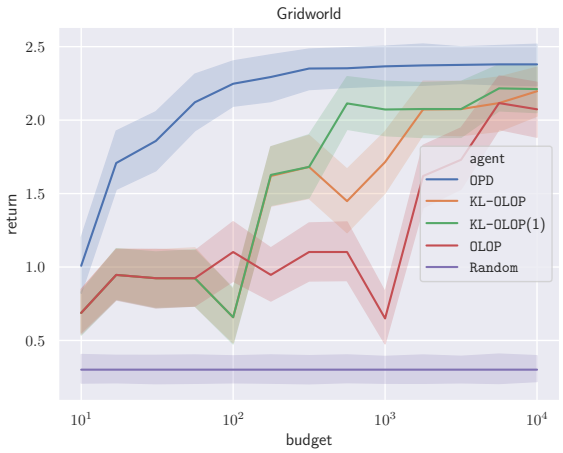
The results of our experiments are shown in Figure 6.3. The `OPD` algorithm converges very quickly to the optimal return in the two first environments, shown in Figure 6.3a and Figure 6.3b, because it exploits their deterministic nature: it needs neither to estimate the rewards through upper-confidence bounds nor to sample whole sequences all the way from

[1] The source code is available at https://eleurent.github.io/kl-olop/.

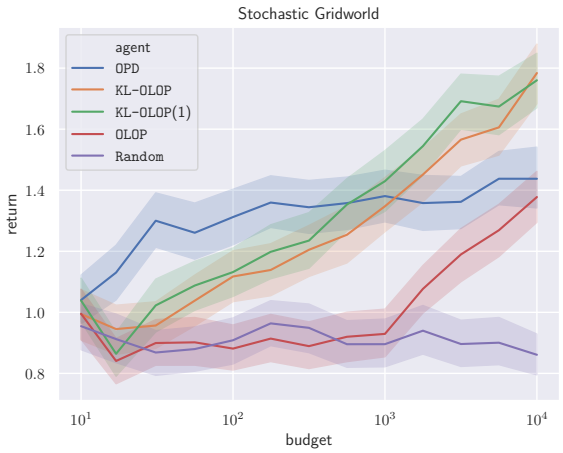
[2] Note that we use the lazy version of `OLOP` and `KL-OLOP` presented in Section C.2.1, otherwise the exponential running-time would have been prohibitive.



(a) Highway



(b) Gridworld



(c) Stochastic Gridworld

Figure 6.3 – Numerical experiments: for each environment-agent configuration, we compute the average return over 100 runs — along with its 95% confidence interval — with respect to the available budget n.

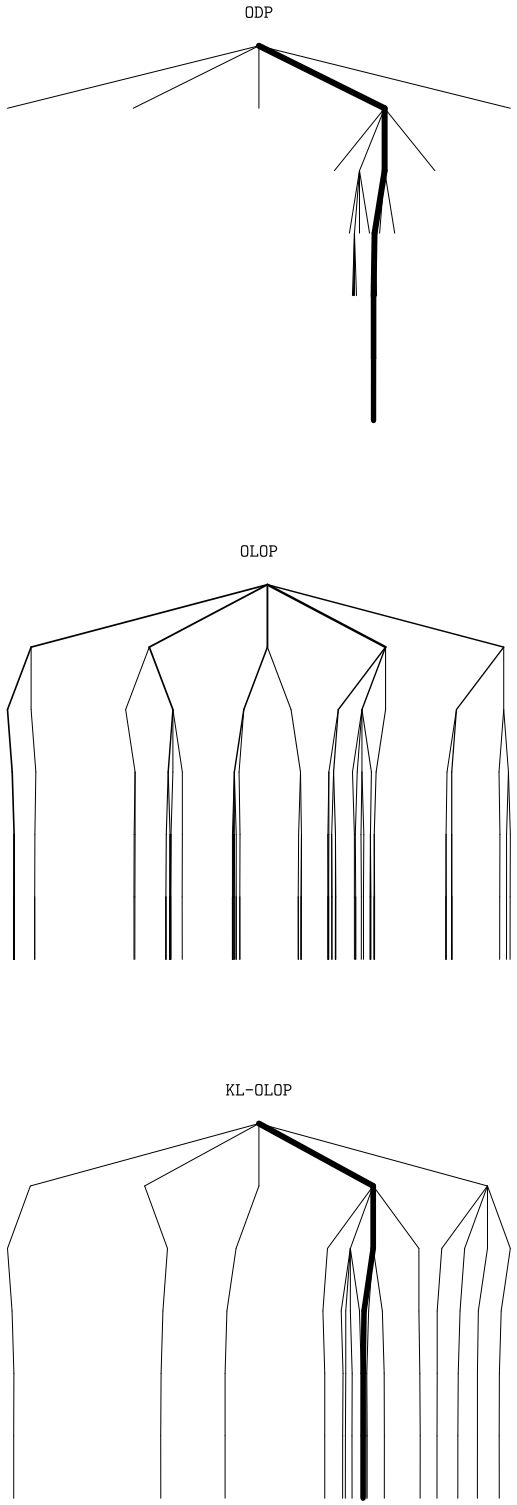


Figure 6.4 – The look-ahead trees (down to depth 6) expanded by the planning algorithms from the same initial state in the highway environment with the same budget $n = 10^3$. The width of edges represents the nodes visit count $N_a(m)$.

the root when expanding a leaf, which provides a significant speed-up. It can be seen as an oracle allowing to measure the conservativeness of stochastic planning algorithms. And indeed, even before introducing stochasticity, we can see that OLOP performs quite badly on the two environments, only managing to solve them with a budget in the order of $10^{3.5}$. In stark contrast, KL-OLOP makes much better use of its samples and reaches the same performance an order of magnitude faster. This is illustrated by the expanded trees shown in Figure 6.4: OPD exploits the deterministic setting and produces a sparse tree densely concentrated around the optimal trajectory. Conversely, the tree developed by OLOP is evenly balanced, which suggests that OLOP behaves as uniform planning as hypothesised in Line 5. KL-OLOP is more efficient and expands a highly unbalanced tree, exploring the same regions as OPD. Furthermore, in the stochastic gridworld environment shown in Figure 6.3c, we observe that the deterministic OPD planner's performance saturates as it settles to suboptimal trajectories, as expected. Conversely, the stochastic planners all find better-performing open-loop policies, which justifies the need for this framework. Again, KL-OLOP converges an order of magnitude faster than OLOP. Finally, KL-OLOP (1) enjoys good performance overall and displays the most satisfying trade-off between aggressiveness in deterministic environments and conservativeness in stochastic environments; hence we recommend this tuning for practical use.

6.2.5 On a closed-loop algorithm

We briefly mention an extension of KL-OLOP to closed-loop policies, in the setting where the transitions have a finite support of size $B < \infty$. The algorithm, named MDP Gap-based Estimation (MDPGapE), was developed as part of a collaboration (Jonsson et al., 2020) and analysed in the fixed-confidence setting.

It relies on estimating confidence sets $\mathcal{C}_t$ on the probability vector $p(\cdot|s,a)$

$$\mathcal{C}_t(s,a) \triangleq \left\{ p \in \Sigma_S : \mathrm{KL}(\widehat{p}_t(\cdot|s,a), p) \leq \frac{\beta^p(n_t(s,a), n)}{n_t(s,a)} \right\},$$

and forming recursive upper and lower confidence bounds in the form

$$U_h(s) = \max_{a \in \mathcal{A}} \left[u_t(s,a) + \gamma \max_{q \in \mathcal{C}_t(s,a)} \sum_{s'} q(s'|s,a) U_{h+1}(s') \right],$$
$$L_h(s) = \max_{a \in \mathcal{A}} \left[\ell_t(s,a) + \gamma \min_{q \in \mathcal{C}_t(s,a)} \sum_{s'} q(s'|s,a) L_{h+1}(s') \right].$$

6.3 Graph-based optimistic planning

The goal of this section is to address a limitation of MCTS algorithms: they rely on a tree structure that –despite its simplicity– *does not allow to merge information across states*. That is, if a state s can be reached via two trajectories, it will be represented twice in the look-ahead tree. For instance, in Figure 6.5 (left), two paths lead to the same state represented in orange. MCTS algorithms do not merge the information of the two trajectories to update a shared estimate of the state value.

Related work The idea of merging information between branches of a search tree appears in (Silver, Hubert, et al., 2018), where the state values are approximated with a shared Neural Network. However, this network is merely updated between two planning instances and not during the planning procedure itself. Another work of interest is that of Hostetler, A. Fern, and Dietterich (2014), who propose to partition the state space S into a smaller set $\mathcal{Y}$ of equivalence classes. By aggregating similar states within a class, they reduce the branching factor of the search tree from $|\mathcal{S}||\mathcal{A}|$ to $|\mathcal{Y}||\mathcal{A}|$, which substantially improves sample complexity as they illustrate empirically. However, this procedure requires providing a relevant state partition, only aggregates trajectories that traverse the same sequence of classes (*i.e.* local deformations), and comes with a (bounded) loss of optimality. The closest work to ours is that of Ballesteros et al. (2013), in the context of partially observable MDPs, who identify similar belief states and plan with a graph structure. They focus on empirically comparing various similarity measures on robotic tasks and do not provide any theoretical analysis of the effect of aggregation. This is precisely our goal and contribution here.

In this section, we introduce a planning algorithm named `GBOP-D`, a graph-based version of the tree-based `OPD` algorithm for deterministic systems. We analyse the benefits of this graph-based formulation in Section 6.3.2, and provide in Theorem 6.27 a regret guarantee. The corresponding regret bound features a novel problem-dependent difficulty measure that we introduce to capture the benefit of using a graph structure. We show that this measure can only improve over the performance of `OPD`, and provide an example where it does. We discuss in Section 6.3.3 an extension of our method to stochastic MDPs, called `GBOP`. Finally, Section 6.3.4 illustrates the benefits of `GBOP` in two numerical simulations.

6.3.1 Graph-Based Planning for Deterministic Systems

In this section, we introduce a simple yet highly effective variant of tree-based planning algorithms. We first consider the simple setting of MDPs with deterministic dynamics, and will denote $P(s,a)$ the unique next state s' sampled from $P\left(s'|s,a\right)$. We start by giving some background on the interplay of data structures and optimistic planning algorithms.

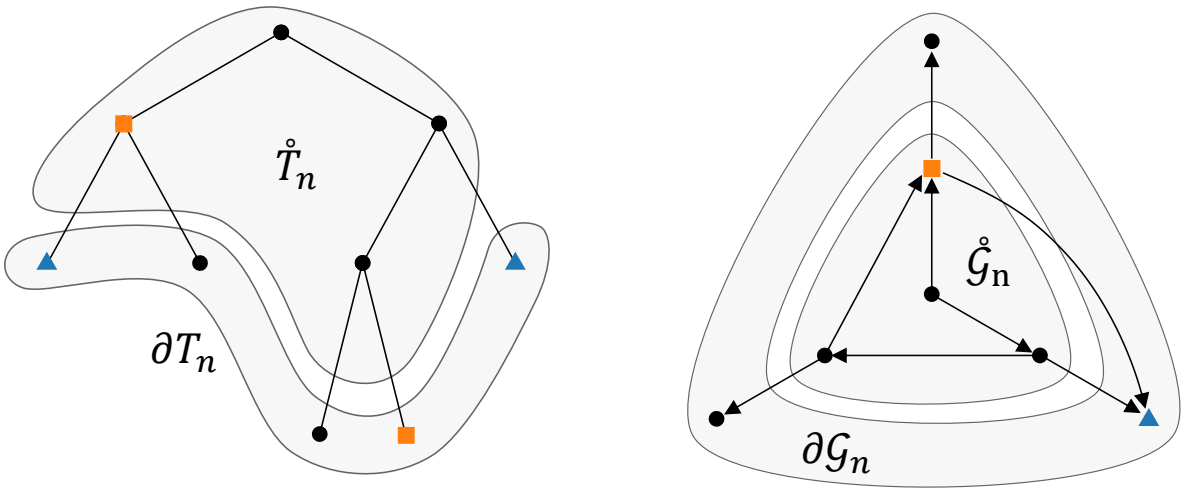


Figure 6.5 – Black arrows depict how the Bellman backup operators B_n (left) and $\mathcal{B}_n$ (right) propagate value estimates from successor nodes to their parents. Information travels freely in a graph, but only upwards in a tree.

Data structures

In this section, we compare two data structures for planning in an MDP: tree and (directed) graph, represented in Figure 6.5. In order to distinguish them, we refer to trees with Roman symbols, *e.g.* T, U, L, B; and to graphs with calligraphic symbols, *e.g.* $\mathcal{G}, \mathcal{U}, \mathcal{L}, \mathcal{B}$. In both structures, we say that a node is *internal* if it has outgoing edges, and *external* else.

In a tree, a node of depth h represents a sequence of actions $a \in \mathcal{A}^h$. The *root* of the tree corresponds to the empty action sequence, and hence to the initial state $s_1 \in S$. At iteration n, we denote the current tree as $\mathcal{T}_n$. Borrowing notations from topology, we denote its set of internal nodes as $\mathring{\mathcal{T}}_n$ and its set of external nodes (the leaves) as $\partial \mathcal{T}_n$. Note that since the MDP is deterministic, a sequence of action a is associated with its final state denoted $s(a)$, but this association is not one-to-one: several sequences of action can lead to the same state, which will be represented several times in the tree.

In a graph, the nodes represent states $s \in S$, and the edges represent transitions between states. The *source* of the graph corresponds to the initial state s_0. At iteration n, we denote the current graph as $\mathcal{G}_n$, its set of internal nodes as $\mathring{\mathcal{G}}_n$ and its set of *sinks* as $\partial \mathcal{G}_n$.

Both structures are built iteratively from a single starting node, by selecting an external node (leaf or sink) to expand. The *expansion* of a node a or s refers to calling the generative model to sample the reward r and next state s' for each action $a \in \mathcal{A}$, and adding child nodes to the data structure. In a tree, the expansion of a node $a \in \mathcal{A}^h$ always lead to the creation of new leaves that represent the suffix sequence of action $ab \in \mathcal{A}^{h+1}$, $b \in \mathcal{A}$. The maximum depth of an expanded node in T_n is denoted d_n. In contrast, in a graph the next state s' reached from

s, a might already be present in $\mathcal{G}_n$, in which case we add the edge between s and s' without creating a new node. These data structures can be used to store information about the MDP, such as the transitions and rewards $r(s, a)$, or other information useful for planning.

Optimistic planning

A planning algorithm is typically composed of two main rules:

(i) A *sampling rule*, that selects promising transitions to simulate at each iteration n;

(ii) A *recommendation rule*, that recommends a good first action $\hat{a}_n$ to take (in s_1).

These rules can be chosen with the goal of minimising the simple regret r_n. A popular approach is to follow the principle of Optimism in the Face of Uncertainty (OFU) (see Rémi Munos, 2014), which consists in exploring the option that maximises an upper-bound of the true objective. In the context of planning, it has been applied by forming bounds on the state value function $V^\star$, that we simply denote V for brevity.

Definition 6.12 (Value bounds)**.**

On trees. *We denote by $L : \mathcal{T}_n \to \mathbb{R}$ and $U : \mathcal{T}_n \to \mathbb{R}$ a lower-bound and upper-bound for the state value V defined on the tree $\mathcal{T}_n$, such that*

$$\forall a \in \mathcal{T}_n, \qquad L(a) \leq V(s(a)) \leq U(a).$$

On graphs. *Likewise, we denote by $\mathcal{L} : \mathcal{G}_n \to \mathbb{R}$ and $\mathcal{U} : \mathcal{G}_n \to \mathbb{R}$ a lower-bound and upper-bound for the state value V defined on the graph $\mathcal{G}_n$, such that*

$$\forall s \in \mathcal{G}_n, \qquad \mathcal{L}(s) \leq V(s) \leq \mathcal{U}(s).$$

Following the OFU principle, at iteration n we must leverage available information to design an upper-bound U_n (or $\mathcal{U}_n$) on V as tight as possible. Then, in order to select a promising external node to expand, the sampling rule starts from the root (or source) and follows the optimistic strategy of always selecting the action which maximises U_n (or $\mathcal{U}_n$), until reaching an optimistic leaf (or sink) to expand. This strategy was used in (*e.g.* Kocsis and Szepesvári, 2006; Hren and Rémi Munos, 2008; Sébastien Bubeck and Rémi Munos, 2010; Buşoniu and Remi Munos, 2012).

For instance, since we assume that the rewards are bounded in $[0, 1]$, trivial bounds on $V(s)$ are $0 \leq V(s) \leq V_{\max} \triangleq \sum_t \gamma^t 1 = 1/(1-\gamma)$. However, these trivial bounds are the same for every node, which makes them non-informative, and do not make use of the observed

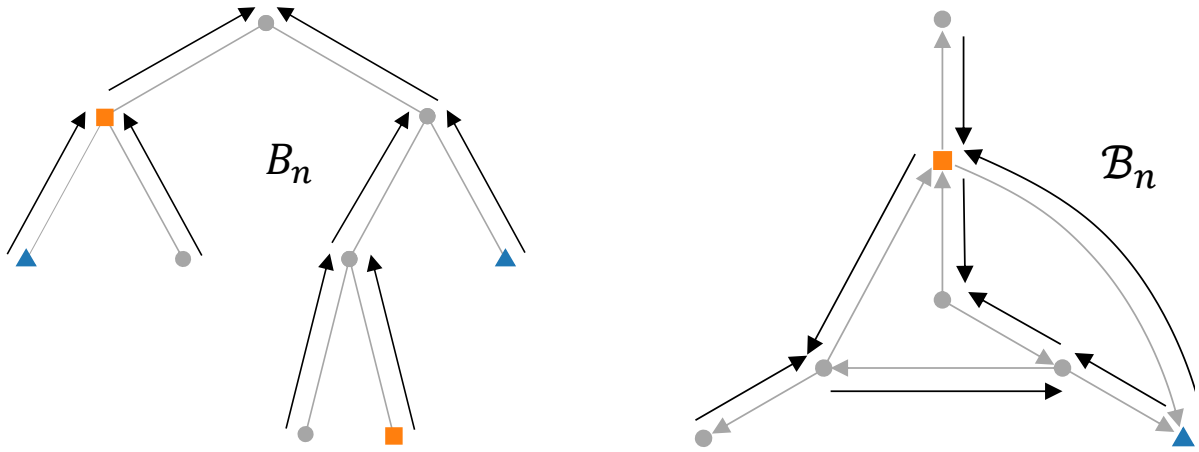


Figure 6.6 – Illustration of the Bellman backup operators B (left) and $\mathcal{B}$ (right). Notice that B_n only propagates information upward in the tree.

information. Still, they can be used as a valid starting point. Every observed transition stored can then be used to tighten these bounds, by resorting to the Bellman optimality operator.

Definition 6.13 (Bellman optimality operator)**.**

On trees. *We define the Bellman optimality operator B_n on the tree $\mathcal{T}_n$ as*

$$B_n(f)(a) \triangleq \begin{cases} \max_{b \in \mathcal{A}} R(s(a), b) + \gamma f(ab) & \text{if } a \in \mathring{\mathcal{T}}_n; \\ f(a) & \text{if } a \in \partial\mathcal{T}_n. \end{cases} \tag{6.12}$$

On graphs. *Likewise, we define the Bellman optimality operator $\mathcal{B}_n$ on the graph $\mathcal{G}_n$ as*

$$\mathcal{B}_n(f)(s) \triangleq \begin{cases} \max_{b \in \mathcal{A}} R(s, b) + \gamma f(P(s, b)) & \text{if } s \in \mathring{\mathcal{G}}_n; \\ f(s) & \text{if } s \in \partial\mathcal{G}_n. \end{cases} \tag{6.13}$$

The updates of both Bellman operators are depicted in Figure 6.6.

Hren and Rémi Munos (2008) used this Bellman operator B_n in their `OPD` algorithm to define a pair of bounds (L_n, U_n) at each iteration n. They use trivial bounds at the leaves, and backup these estimates up to the root by iteratively applying B_n. We can show that, under a *monotonicity* condition (satisfied by the trivial bounds 0 and V_{max}), applying B_n can only tighten a bound and converges in a finite time.

Definition 6.14 (Monotonicity)**.** *A pair of bounds (L, U) or $(\mathcal{L},\mathcal{U})$ is* monotonic *if they are respectively non-decreasing and non-increasing along transitions:*

$$\begin{aligned} &\forall a \in \mathcal{T}_n, \quad L(a) \leq B_n(L)(a), && U(a) \geq B_n(U)(a) \\ &\forall s \in \mathring{\mathcal{G}}_n, \quad \mathcal{L}(s) \leq \mathcal{B}_n(\mathcal{L})(s), && \mathcal{U}(s) \geq \mathcal{B}_n(\mathcal{U})(s) \end{aligned}$$

Lemma 6.15 (Properties of B_n)**.**

(*i*) *B_n preserves monotonicity and tightens monotonic bounds:*

$$\text{if } L \leq V \leq U, \text{ then } L \leq B_n(L) \leq V \leq B_n(U) \leq U;$$

(*ii*) *The sequence $B_n^k = \underbrace{B_n \circ \cdots \circ B_n}_{k \text{ times}}$ converges in a finite time $k = d_n$, where d_n is the depth of $\mathcal{T}_n$.*

Proof. We provide a proof in Section C.1.5. □

This enables Hren and Rémi Munos (2008) to define[3] non-trivial valid bounds on V:

$$L_n \triangleq B_n^{d_n}(0), \qquad U_n \triangleq B_n^{d_n}(V_{\max}), \tag{6.14}$$

where 0 is the null function. The corresponding `OPD` algorithm is described in Algorithm 6.2.

Likewise, we show that the graph version $\mathcal{B}_n$ verifies similar properties.

Lemma 6.16 (Properties of $\mathcal{B}_n$)**.**

(*i*) *$\mathcal{B}_n$ preserves monotonicity and tightens monotonic bounds:*

$$\text{if } \mathcal{L} \leq V \leq \mathcal{U}, \text{ then } \mathcal{L} \leq \mathcal{B}_n(\mathcal{L}) \leq V \leq \mathcal{B}_n(\mathcal{U}) \leq \mathcal{U};$$

(*ii*) *$\mathcal{B}$ is a γ-contraction, and we denote $\mathcal{B}_n^\infty \triangleq \lim_{k\to\infty} \mathcal{B}_n^k$.*

Proof. We provide a proof in Section C.1.6. □

[3] We use an iteration of operators while a recursive definition was used originally.

Algorithm 6.2: The *Optimistic Planning of Deterministic Systems* (`OPD`) algorithm from (Hren and Rémi Munos, 2008).

for *each iteration* n **do**
  Compute the bounds $L_n = B_n^{d_n}(0)$ and $U_n = B_n^{d_n}(V_{\max})$.
  $b_n \leftarrow \emptyset$
  while *the node* $b_n \in \mathring{\mathcal{T}}_n$ *is internal* **do**
    $b_n \leftarrow \underset{a' \in b_n A}{\arg\max}\, r(a') + \gamma U_n(a')$ ▷ `Optimistic sampling rule`
  for *action* $a \in \mathcal{A}$ **do** ▷ `Node expansion`
    Simulate $r \leftarrow r(s(b_n), a)$ and $s' \leftarrow P(s(b_n), a)$.
    Add a new leaf $b_n a$ to $\mathcal{T}_{n+1}$, with associated reward r.
return $\underset{a \in \mathcal{A}}{\arg\max}\, r(s, a) + \gamma L_n(a)$. ▷ `Conservative recommendation rule`

This motivates us to propose Algorithm 6.3, following the approach of Algorithm 6.2 adapted to a graph structure.

Algorithm 6.3: Our proposed *Graph-Based Optimistic Planning for Deterministic systems* (`GBOP-D`) algorithm.

for *each iteration* n **do**
3 Compute the bounds $\mathcal{L}_n = \mathcal{B}_n^{\infty}(0)$ and $\mathcal{U}_n = \mathcal{B}_n^{\infty}(V_{\max})$.
4 $s_n \leftarrow s_1$
5 **while** *the node* $s_n \in \mathring{\mathcal{G}}_n$ *is internal* **do**
7 $s_n \leftarrow \underset{s'}{\arg\max}\, r(s_n, a) + \gamma \mathcal{U}_n(s')$ ▷ `Optimistic sampling rule`
8 **for** *action* $a \in \mathcal{A}$ **do** ▷ `Node expansion`
9 Simulate $r \leftarrow r(s_n, a)$ and $s' \leftarrow P(s_n, a)$.
10 Get or create the node s' in $\mathcal{G}_{n+1}$, and add the transition $(s_n, a) \to s', r$.
11 **return** $\underset{a \in \mathcal{A}}{\arg\max}\, R(s, a) + \gamma \mathcal{L}_n(s(a))$. ▷ `Conservative recommendation rule`

Remark 6.17 (Termination and complexity)**.** *There are two procedures in `GBOP-D` that may not terminate in finite time when $\mathcal{G}_n$ contains a loop: the computation of $\mathcal{B}_n^{\infty}$ (line 1) and the sampling rule loop (line 2). We handle these steps carefully in Section C.2.2, where we discuss an approximate implementation in which these two procedures are stopped whenever they reach a desired accuracy ε, along with an analysis of the corresponding time complexity and impact on the performance.*

Though both algorithms share a similar design, we claim that using graphs provides substantial theoretical and practical performance improvements, and back up this statement in Sections 6.3.2 and 6.3.4.

6.3.2 Analysis

Comparing OPD and GBOP-D directly is difficult since they do not involve the same structure, which causes implicit differences in their behaviours. Studying them under a common framework makes these differences explicit. To leverage the analysis of OPD by Hren and Rémi Munos (2008), we will frame GBOP-D as a tree-based planning algorithm: the graph operator $\mathcal{B}$ will be represented as tree backup B applied on an *unrolled* tree $T(\mathcal{G}_n)$, defined below.

Background on the sample complexity of OPD

First, we recall the analysis of Hren and Rémi Munos (2008) and introduce some notations.

Lemma 6.18 (Sequence values)**.** *The value of a finite **sequence** of actions $a \in \mathcal{A}^h$, defined in Definition 6.1, verifies*

$$V(a) = G(s_1, a) + \gamma^h V(s(a)),$$

where $G(s,a) = \sum_{t=0}^{h-1} \gamma^t r_t$ is the return obtained by executing the sequence of actions a starting from the state s.

Proof. We provide a proof in Section C.1.7. □

This enables to define a measure of the difficulty of a planning problem.

Definition 6.19 (Difficulty measure)**.** *We define the near-optimal branching factor $\boldsymbol{\kappa}$ of an MDP as*

$$\boldsymbol{\kappa} = \limsup_{h\to\infty} |\mathcal{T}_h^\infty|^{1/h} \in [1, K] \tag{6.15}$$

where $\mathcal{T}_h^\infty = \left\{ a \in \mathcal{A}^h : V^\star - V(a) \leq \dfrac{\gamma^h}{1-\gamma} \right\}$ is the set of near-optimal nodes at depth h.

This problem-dependent measure $\boldsymbol{\kappa}$ is the branching factor of the subtree $T^\infty = \bigcup_h T_h^\infty$ of near-optimal nodes that can be sampled by OPD, and acts as an effective branching factor as opposed to the true branching factor K. When $\boldsymbol{\kappa}$ is small, fewer nodes must be explored at a given depth allowing the algorithm to plan deeper for a given budget n. Thus, it directly impacts the simple regret that can be achieved by OPD when run on a given MDP.

Theorem 6.20 (Regret bound of Hren and Rémi Munos, 2008)**.** *The Algorithm 6.2 enjoys the following regret bound:*

$$r_n = \tilde{\mathcal{O}}\left(n^{-\log\frac{1}{\gamma}/\log\kappa}\right),$$

where $f_n = \tilde{\mathcal{O}}(n^{-\alpha})$ means that for any $\alpha' < \alpha$, $f_n = \mathcal{O}(n^{-\alpha'})$, for all $\alpha \in \mathbb{R}_+ \cup \{+\infty\}$.

Proof. We provide a proof in Section C.1.8. □

The near-optimal branching factor κ is related (Sébastien Bubeck and Rémi Munos, 2010) to the near-optimality dimension studied in the online optimisation literature (see *e.g.* Sébastien Bubeck et al., 2009; Rémi Munos, 2011). It is typically small in problems where there is one single optimal trajectory, of which any deviation can be quickly dismissed as suboptimal. Conversely, κ is large when many sub-optimal trajectories cannot be distinguished easily based on their values, which requires the exploration of a large part of the tree T of branching factor K.

Motivation for an improved regret bound

We start by reformulating the sampling rule used for the `OPD` algorithm. To that end, notice that when some bounds (L, U) on the state values $V(s(a))$ are available, they also induce bounds $(\overline{L}, \overline{U})$ on values $V(a)$ of sequences of actions a of length h defined as

$$\underbrace{R(s_1, a) + \gamma^h L(a)}_{\overline{L}(a)} \leq V(a) \leq \underbrace{R(s_1, a) + \gamma^h U(a)}_{\overline{U}(a)}.$$

One can easily see that, since the (L_n, U_n) used in the optimistic sampling rule described in Algorithm 6.2 are invariant by B_n by definition, this rule can be equivalently expressed as

$$b_n \in \arg\max_{a \in \partial\mathcal{T}_n} \overline{U}_n(a). \tag{6.16}$$

Likewise, the conservative recommendation rule returns the first action of

$$a_n \in \arg\max_{a \in \partial\mathcal{T}_n} \overline{L}_n(a) \tag{6.17}$$

As shown in Figure 6.6, in a tree the Bellman operator B_n only propagates the information upward, and the leaves cannot be updated. Thus, $U_n = B_n^{d_n}(V_{\max})$ and $V_{\max}$ coincide on $\partial\mathcal{T}_n$ which means that the sampling rule of `OPD` can be summarized as using (6.16) with the trivial

upper-bound $U_n = V_{\max}$. Likewise, the recommendation rule simply uses (6.17) with the trivial lower-bound $L_n = 0$. Thus, OPD amounts to simply using the trivial bound $(0, V_{\max})$ on leaf nodes, and does not make use of all the available information in $\mathcal{T}_n$ to improve these bounds.

Let us now assume for the moment that we had access to tighter bounds (L, U) provided by an oracle:

$$0 \leq L \leq V \leq U \leq V_{\max}.$$

Definition 6.21 (A finer difficulty measure)**.** *We define the near-optimal branching factor according to the bounds (L, U) as*

$$\boldsymbol{\kappa(L,U)} \triangleq \limsup_{h\to\infty} \left|\mathcal{T}_h^\infty(\boldsymbol{L},\boldsymbol{U})\right|^{1/h} \in (1, K], \tag{6.18}$$

where $\mathcal{T}_h^\infty(L,U) = \left\{a \in \mathcal{A}^h : V^\star - V(a) \leq \gamma^h(U(a) - L(a))\right\}$.

Lemma 6.22. *This branching factor shrinks as the bounds (L, U) get tighter:*

$$L_2 \leq L_1 \leq V \leq U_1 \leq U_2 \implies \kappa(L_1,U_1) \leq \kappa(L_2,U_2).$$

In particular, $\boldsymbol{\kappa(L,U)} \leq \boldsymbol{\kappa}$.

Proof. We provide a proof in Section C.1.9. □

Theorem 6.23. *Let $L \leq V \leq U$ monotonic bounds, then planning with L and U in (6.16) and (6.17) yields the following simple regret bound:*

$$r_n = \tilde{\mathcal{O}}\left(n^{-\log\frac{1}{\gamma}/\log\boldsymbol{\kappa(L,U)}}\right).$$

Proof. We provide a proof in Section C.1.10. □

This theorem states that we can potentially improve the performance of the planning algorithm if we manage to find bounds (L, U) that are tighter than the trivial ones at the leaves $\partial\mathcal{T}_n$, which may be possible if we have already seen the states corresponding to this leaves, but it does not explain how to obtain such bounds. In the next subsection, we describe a method to build a sequence of increasingly tight bounds (L_n, U_n), at each planning iteration n. The corresponding regret bound, our main result, is stated in Theorem 6.27.

Unrolling the tree to tighten the bounds

In order to reproduce the behaviour of Algorithm 6.3 on a tree structure, we rely on the following observation: expanding a node s in $\mathcal{G}_n$ simultaneously expands all the paths leading to this node. To account for this observation in the analysis, we will consider an *unrolling* operator T that builds a potentially infinite tree $T(\mathcal{G}_n)$ containing every sequence of actions that can be traversed in a graph $\mathcal{G}_n$.

$$T(\mathcal{G}_n) = \{a \in \mathcal{A}^h : s_{t+1} \in \mathcal{G}_n \text{ with } s_{t+1} = P(s_t, a_t) \text{ for } 0 \leq t < h\} \tag{6.19}$$

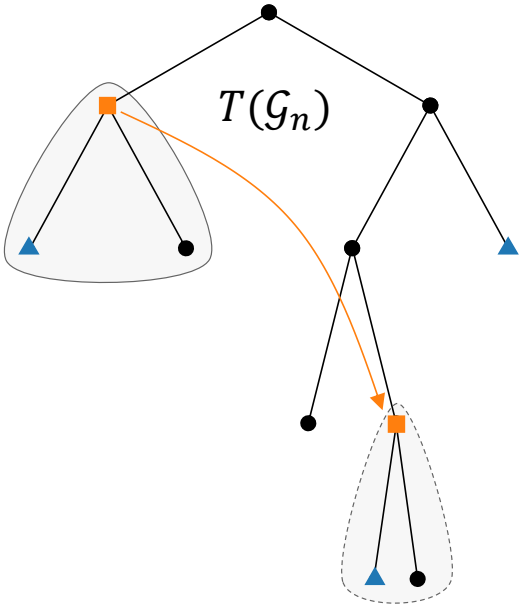


Figure 6.7 – The tree $T(\mathcal{G}_n)$ obtained by unrolling $\mathcal{G}_n$. Contrary to T_n shown in Figure 6.5, the red leaf a is expanded at the same time as the internal red node, which enables to tighten its value bounds $(L_n(a), U_n(a))$ by applying B_n.

We analyse GBOP-D though the prism of $T(\mathcal{G}_n)$, which is only used as a theoretical tool.

We can define the counterpart of the bounds $(\mathcal{L}_n, \mathcal{U}_n)$ in the same way as (6.14) applied to $T(\mathcal{G}_n)$ rather than T_n, except that the depth d_n of $T(\mathcal{G}_n)$ can now be infinite:

$$L_n = B_n^{\infty}(0), \qquad U_n = B_n^{\infty}(V_{\max}). \tag{6.20}$$

This definition is equivalent to that of GBOP-D in the sense that

Lemma 6.24 (Bound equivalence). *For any sequence of action $a \in T(\mathcal{G}_n)$, we have $L_n(a) = \mathcal{L}_n(s(a))$ and $U_n(a) = \mathcal{U}_n(s(a))$.*

Proof. We provide a proof in Section C.1.11. □

In $T(\mathcal{G}_n)$, the unrolling mechanics behave as if any leaf a sharing the same state $s(a)$ as an internal node a' was automatically expanded, and thus had its bound $L_n(a), U_n(a)$ tightened by the Bellman backup B_n to a sub-interval of the trivial bounds $(0, V_{\max})$ that are used in OPD.

The sampling and recommendation rules of GBOP-D also amount to running those of OPD on the tree $T(\mathcal{G}_n)$, except that the sampled sequence b_n and recommended sequence a_n can now have infinite depth since $T(\mathcal{G}_n)$ itself can be infinite (we say that a_n and b_n are represented by nodes of infinite depth). In the sequel, we analyse how these rules behave on $T(\mathcal{G}_n)$.

Lemma 6.25 (Expansion). *Any node a of depth h traversed by the optimistic sampling rule of GBOP-D at iteration n belongs to $T_h^\infty(L_n, U_n)$:*

$$V^\star - V(a) \leq \gamma^h (U_n(a) - L_n(a)). \tag{6.21}$$

In particular, if the sampling rule samples an infinite sequence $a \in \mathcal{A}^\infty$, it is an optimal sequence, and we write that (6.3) also holds for a with $h = \infty$.

Proof. We provide a proof in Section C.1.12. □

Lemma 6.26 (Recommendation). *The action a_n recommended by GBOP-D has a simple regret $r_n \leq \frac{\gamma^{d_n}}{1-\gamma}$, where $d_n \in \mathbb{R} \cup \{\infty\}$ is the maximal depth of expanded nodes in $T(\mathcal{G}_n)$.*

Proof. We provide a proof in Section C.1.13. □

Note that even though $T(\mathcal{G}_n)$ can be infinite, there is only one node b_t that is selected for expansion at each iteration $t \leq n$.

Regret guarantee

In Theorem 6.23, we assumed that some bounds (L, U) were revealed by an oracle and available from the onset for planning. In (6.20), we instead built a *sequence* of bounds $(L_n, U_n)_{n\geq 0}$ (6.20) that is non-increasing in the sense of inclusion, *i.e.*

$$0 \leq \cdots \leq L_{n-1} \leq L_n \leq V \leq U_n \leq U_{n-1} \leq \cdots \leq V_{\max}.$$

We can consider the sequence $\kappa_n = \kappa(L_n, U_n)$. It is non-increasing and lower-bounded by 1, thus converges. Let $\boldsymbol{\kappa_\infty = \lim_{n\to\infty} \kappa(L_n, U_n)} \in [1, K]$.

Theorem 6.27. *GBOP-D enjoys the following regret bound, with $\kappa_\infty \leq \kappa$:*

$$r_n = \tilde{\mathcal{O}}\left(n^{-\log \frac{1}{\gamma}/\log \kappa_\infty}\right).$$

Proof. We provide a proof in Section C.1.14. □

Intuitively, κ_∞ should be much lower than κ in problems where trajectories overlap a lot. For instance, it will be the case for problems where two actions cancel each other out (*e.g.* moving left or right), or are commutative (*e.g.* placing pawns on a board game). However, this is merely an intuition. We now show that there exist problem instances in which $\kappa_\infty < \kappa$, which is a legitimate concern since their non-existence would make Theorem 6.27 trivial.

Illustrative example

In Proposition 6.28, we consider a toy MDP $\mathcal{M}$ shown in Figure 6.8. The transitions are described visually while the rewards are defined as follows: let $0 \leq r^\star \leq \gamma$, and $r^- = r^\star - \frac{\gamma}{1-\gamma} S$, $r^+ = r^\star + S$ with $S = r^\star \left(\frac{1}{\gamma} - 1\right)$. Note that this choice ensures that $r^\star, r^-, r^+$ and S are all in $[0, 1]$.

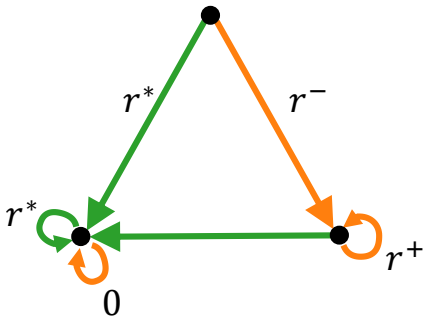


Figure 6.8 – A toy MDP with three states and $K \geq 2$ actions. We start in the top state. The first action a_1 is represented by green arrows, and all other actions $a_2, \dots, a_K$ are represented by orange arrows. The rewards are shown next to the transitions.

Proposition 6.28 (Branching factors)**.** *The MDP $\mathcal{M}$ verifies $\boldsymbol{\kappa = K - 1}$ and $\boldsymbol{\kappa_\infty = 1}$.*

Proof. We provide a proof in Section C.1.15. □

This result confirms that Theorem 6.27 is non-trivial since we exhibit a problem for which $\kappa_\infty < \kappa$ (when $K \geq 3$), and legitimates our attempt to improve planning performances by merging the tree into a graph.

6.3.3 Extension to Stochastic Systems

The approach developed in Sections 6.3.1 and 6.3.2 consists in using state similarity to tighten a pair of lower and upper bounds (L, U) for the value function V. Thus, any planning algorithm that is based on such bounds can benefit from this insight, and any theoretical result based on the validity and rate of convergence of these bounds will be preserved.

Confidence intervals for rewards. When the reward kernel $P(r \mid s, a)$ is stochastic[4], deviation inequalities can be used to design a confidence interval $[\ell_t(s,a), u_t(s,a)]$ over its expected value $\mathbb{E}[r|s,a]$. For instance, the Chernoff-Hoeffding deviation inequality was used to design confidence intervals in (Kocsis and Szepesvári, 2006; Sébastien Bubeck and Rémi Munos, 2010; Kaufmann and Koolen, 2017). In our recent works (Leurent and Maillard, 2020b; Jonsson et al., 2020), the tighter Kullback-Leibler confidence interval is preferred:

$$u_t(s,a) \triangleq \max\left\{v : \mathrm{kl}(\hat{r}_t(s,a), v) \leq \frac{\beta^r(n_t(s,a), n)}{n_t(s,a)}\right\},$$
$$\ell_t(s,a) \triangleq \min\left\{v : \mathrm{kl}(\hat{r}_t(s,a), v) \leq \frac{\beta^r(n_t(s,a), n)}{n_t(s,a)}\right\},$$

where $\hat{r}_t(s,a)$ is the sample mean, β^r is an exploration function and $\mathrm{kl}(u,v)$ is the binary Kullback-Leibler divergence between Bernoulli distributions: $\mathrm{kl}(u,v) = u\log\frac{u}{v} + (1-u)\log\frac{1-u}{1-v}$.

Confidence region for transitions. Likewise, when the transition kernel $P(s' \mid s, a)$ is stochastic, a confidence set on the probability vector $p(\cdot|s,a)$ can be defined as

$$\mathcal{C}_t(s,a) \triangleq \left\{p \in \Sigma_S : \mathrm{KL}(\widehat{p}_t(\cdot|s,a), p) \leq \frac{\beta^p(n_t(s,a), n)}{n_t(s,a)}\right\},$$

where $\widehat{p}_t(\cdot|s,a) \triangleq n_t(s,a,\cdot)/n_t(s,a)$ is the empirical distribution, Σ_S is the probability simplex over S, β^p is an exploration function and $\mathrm{KL}(p,q) = \sum_{s\in S} p(s)\log\frac{p(s)}{q(s)}$ is the Kullback-Leibler divergence between categorical distributions.

Bellman operator with stochasticity. In this section, we do not discuss the tuning of β^r, β^p, but simply assume that they are chosen such that the rewards and transitions belong to their

[4] We assumed known deterministic rewards until now, since it is typically chosen by the autonomous vehicle designer.

confidence regions with sufficiently high probability to obtain performance guarantees for the planning algorithm. For more details on such a choice, refer to Section 6.2.1 or (*e.g.* Jonsson et al., 2020). We modify the Definition 6.13 of the Bellman operator on graphs $\mathcal{B}_t$ as

$$\mathcal{B}_t^+(\mathcal{U})(s) = \max_{a\in\mathcal{A}} \left[u_t(s,a) + \gamma \max_{q\in\mathcal{C}_t(s,a)} \sum_{s'} q(s'|s,a)\mathcal{U}(s') \right],$$
$$\mathcal{B}_t^-(\mathcal{L})(s) = \max_{a\in\mathcal{A}} \left[\ell_t(s,a) + \gamma \min_{q\in\mathcal{C}_t(s,a)} \sum_{s'} q(s'|s,a)\mathcal{L}(s') \right],$$

for all $s \in \mathring{\mathcal{G}}_n$, where the maximum and minimum over these Kullback-Leibler confidence regions $\mathcal{C}_t(s,a)$ can be computed as explained in (Appendix A of Filippi, O. Cappé, and A. Garivier, 2010). Under the event that every confidence regions $[\ell_t(s,a), u_t(s,a)]$ and $\mathcal{C}_t(s,a)$ are valid at time t, the Lemma 6.16 still holds for $\mathcal{B}_t^-, \mathcal{B}_t^+$.

Structure of the planning algorithm In the deterministic setting, once a transition has been observed, it is known with certainty and does not need to be sampled ever again, which is why only external nodes $\partial\mathcal{G}_n$ are sampled in `GBOP-D`. Conversely, in the stochastic setting, the expected reward and transition probabilities must be estimated from samples, which implies that internal nodes $\mathring{\mathcal{G}}_n$ must be sampled as well. Then, it is common to adopt an episodic standpoint where we sample trajectories of a fixed horizon H, tuned depending on the budget n. This is the case in (*e.g.* Kearns, Mansour, and Ng, 2002; Kocsis and Szepesvári, 2006; Sébastien Bubeck and Rémi Munos, 2010; Feldman and Domshlak, 2014; Leurent and Maillard, 2020b; Jonsson et al., 2020). We also follow this scheme in our proposed Algorithm 6.4 algorithm.

Algorithm 6.4: *Graph-Based Optimistic Planning* (`GBOP`) algorithm.

for *trajectory* m *in* $[1, M]$ **do**
  for *time* t *in* $[1, H]$ **do**
    $n \leftarrow (m-1)H + h$.
    Compute the bounds $\mathcal{L}_n = (\mathcal{B}_n^-)^\infty(0)$ and $\mathcal{U}_n = (\mathcal{B}_n^+)^\infty(V_{\max})$.
    $b_t \leftarrow \arg\max_{a\in\mathcal{A}} r(s_t,a) + \gamma\mathcal{U}_n(s')$ ▷ `Optimistic sampling rule`
    Simulate $r_t, s_{t+1} \sim P(r, s_{t+1} \mid s_t, b_t)$.
    Get or create the node s_{t+1} in $\mathcal{G}_{n+1}$, and add an occurence of the transition (s_t, b_t, r_t, s_{t+1}).
return $\arg\max_{a\in\mathcal{A}} r(s,a) + \gamma\mathcal{L}_n(s(a))$. ▷ `Conservative recommendation rule`

6.3.4 Numerical Experiments

To evaluate the practical benefits of our approach, we compare graph-based and tree-based planning algorithms in various problems.

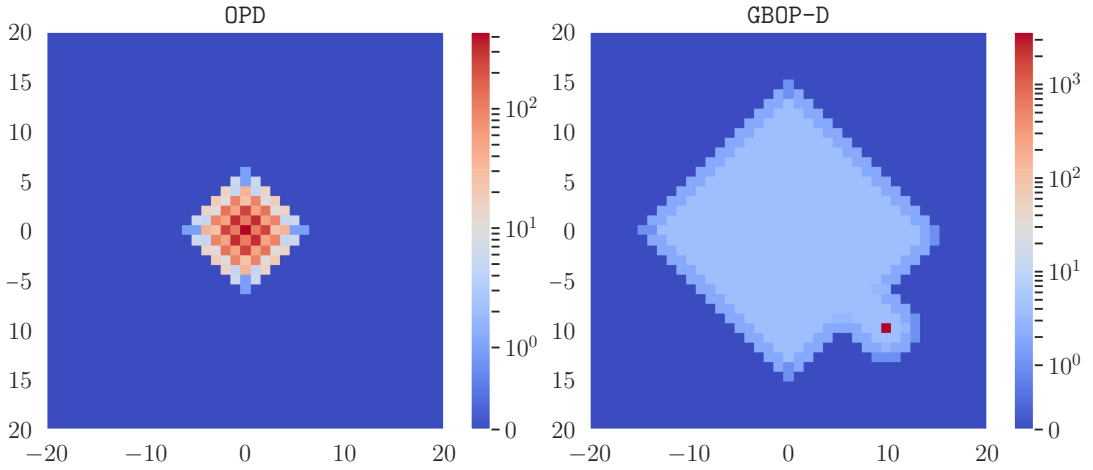


Figure 6.9 – State occupancies of tree-based *vs.* graph-based algorithms in a deterministic gridworld.

Parameters. In every experiment, we used $\gamma = 0.95$. In GBOP-D and GBOP, the fixed accuracy $\varepsilon = 1 \times 10^{-2}$ was used for computing $\mathcal{B}_n^\infty$, and the sampling rule was stopped after reaching the depth $d_n^+ = n$ (see Section C.2.2). Regarding the tuning of confidence intervals in GBOP, since the rewards are deterministic and the transitions are stochastic in both domains we used $\beta^r(n_t(s,a),n) = 0$ and $\beta^p(n_t(s,a),n) = \log n$, following the observations of Section 6.2.1. The maximal size B of the support of the transitions $P(s'|s,a)$ was also used ($B = 4$ in the gridworld domain and $B = 3$ in the sailing domain) to accelerate the computations of the confidence region $\mathcal{C}_t$ for transitions, as explained in (Jonsson et al., 2020).

Gridworld domain. We consider a grid in which the agent can move in $K = 4$ directions. The reward function is 0 everywhere, except in the vicinity of a goal located at $(10, 10)$, around which the reward decreases quadratically from 1 to 0 in a ball of radius 5. The Figure 6.9 shows number of times a state is sampled by OPD and GBOP-D, both run with a budget $n = 5460$ and discount $\gamma = 0.95$. In the absence of rewards, OPD samples sequences of actions uniformly (in a breadth-first search manner), which –because of the dynamics structure– results in a non-uniform occupancy of the state space S, where the trajectories concentrate near the starting state. In contrast, GBOP-D explores uniformly in S, sampling each state up to four times (from its four neighbours), until it finds the goal vicinity and finally samples the goal location indefinitely. We reproduce the experiment in the stochastic setting by adding noise on the transitions with probability $p = 10\%$, and comparing GBOP to UCT as we show in Figure 6.10. To quantify

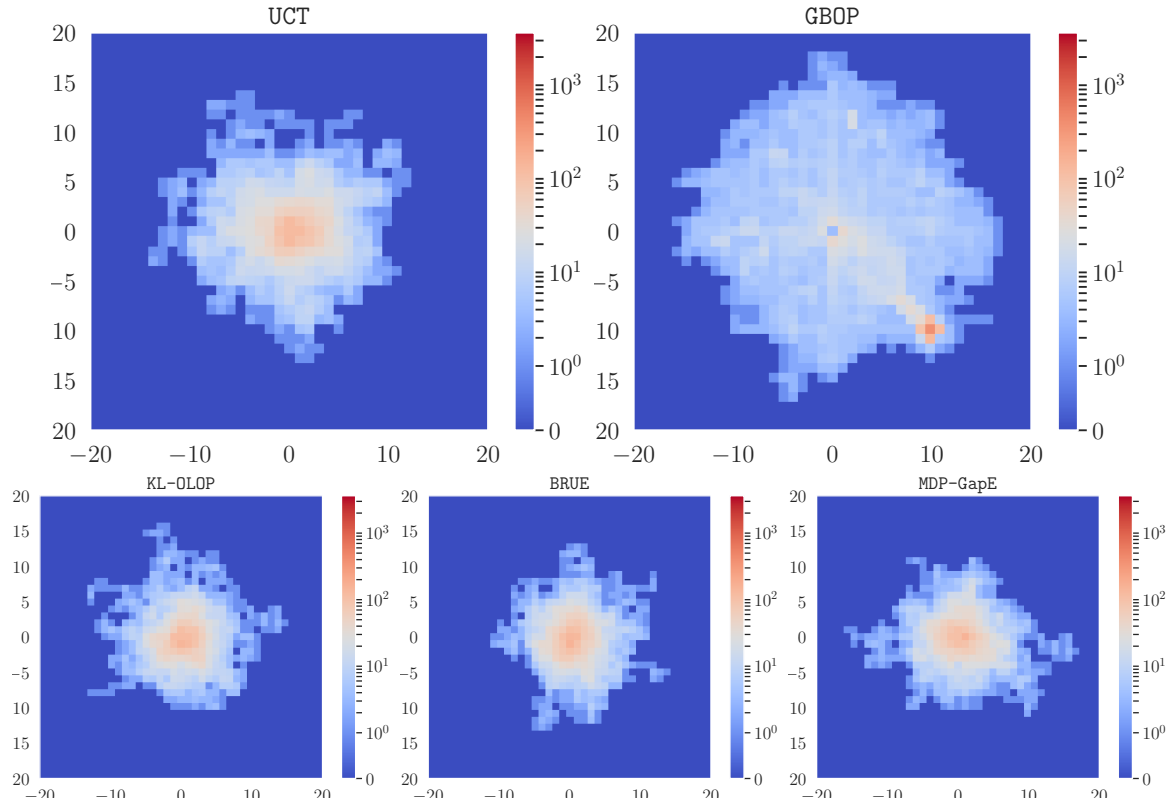
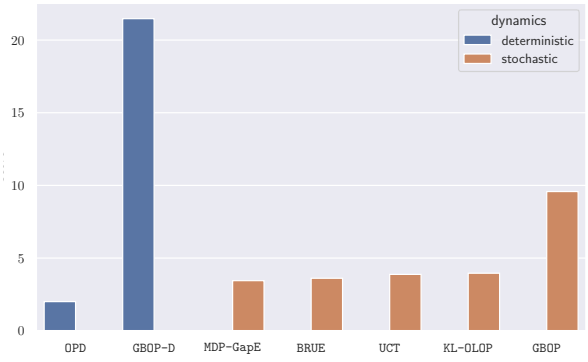


Figure 6.10 – State occupancies of tree-based *vs.* graph-based algorithms in a stochastic gridworld.

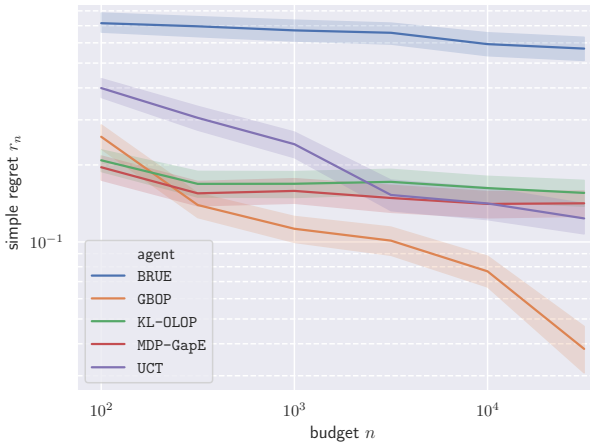
these qualitative differences, we define in Figure 6.11a an exploration score: the mean distance $d(s_t, s_1)$ of sampled states to the initial state (exploration) minus the distance $d(s_t, s_g)$ to the goal state (exploitation).

Sailing domain (Vanderbei, 1996). In a second experiment, a boat is sailing in $K = 8$ directions to reach a goal, and suffers a cost (move duration) that depends on the direction of the wind which follows stochastic dynamics. Figure 6.11b shows the evolution of the simple regret r_n of stochastic planning algorithms with respect to the number n of oracle calls. We show the mean regret and its 95% confidence interval computed over 500 simulations. The asymptotic log-log slope σ provides an empirical measurement of the effective branching factor $\kappa_e = \exp(-\log(1/\gamma)/\sigma)$ for each algorithm. We measure that for $n > 10^{3.5}$, $\sigma \approx -0.04$ and $\kappa_e \approx 3.6$ for `BRUE`, `KL-OLOP`, `MDPGapE`, `UCT`. In contrast, we measure $\sigma \approx -0.3$ and $\kappa_e \approx 1.2$ for `GBOP`, which suggests that our result of Theorem 6.27 might generalize to the stochastic setting.

In Figure 6.12, we compare the trees T_n expanded by `OPD` and `KL-OLOP` to the unrolled graph $T(\mathcal{G}_n)$ of `GBOP-D` in the deterministic gridworld domain. For clarity of the figure, we only display the nodes selected for expansion and not the entire $T(\mathcal{G}_n)$ as in Figure 6.7, since it is infinite and fractal. We observe that `OPD` explores uniformly in the space of sequences, which results



(**a**) Exploration score in the gridworld domain for several algorithms.



(**b**) Mean simple regret r_n in the sailing domain, with its 95% confidence interval.

Figure 6.11 – Benchmark of planning performances.

in a concentrated exploration in the state space, as seen in Figure 6.9. This phenomenon is similar to the concentration properties of martingales. KL-OLOP behaves similarly, but allocates its budget n in fewer trajectories of higher length than OPD, which results in a sparser tree and slightly better exploration (compare Figure 6.9 to Figure 6.10). In contrast, GBOP-D expands a very sparse and unbalanced tree $T(\mathcal{G}_n)$ which corresponds to uniform exploration in the state space, and allows to explore deeper for the same budget n (The tree is only shown up to depth 12, but continues much deeper since the optimal transition is sampled indefinitely once it is discovered). In particular, the paths towards the goal are sampled many times, while other algorithms are still balanced at the root in terms of number of visits.

Chapter conclusion

In this chapter, we studied the theoretical and practical performances of planning algorithms, that can be used to compute a near-optimal trajectory for a known stochastic non-linear system. We first introduced an enhanced version of the OLOP algorithm for open-loop online planning, whose design was motivated by an investigation of the over-conservative search behaviours of OLOP. We analysed its sample complexity and showed that the original regret bounds are preserved, while its empirical performances are increased by an order of magnitude in several numerical experiments. Second, we studied the value of merging information between similar states during planning. We showed that using a graph structure provides a benefit compared to tree structures in the deterministic setting, in the form of an improved regret bound that depends on a smaller problem difficulty. This improvement translates into enhanced performance in practice, and can be adapted to stochastic problems as we demonstrate experimentally.

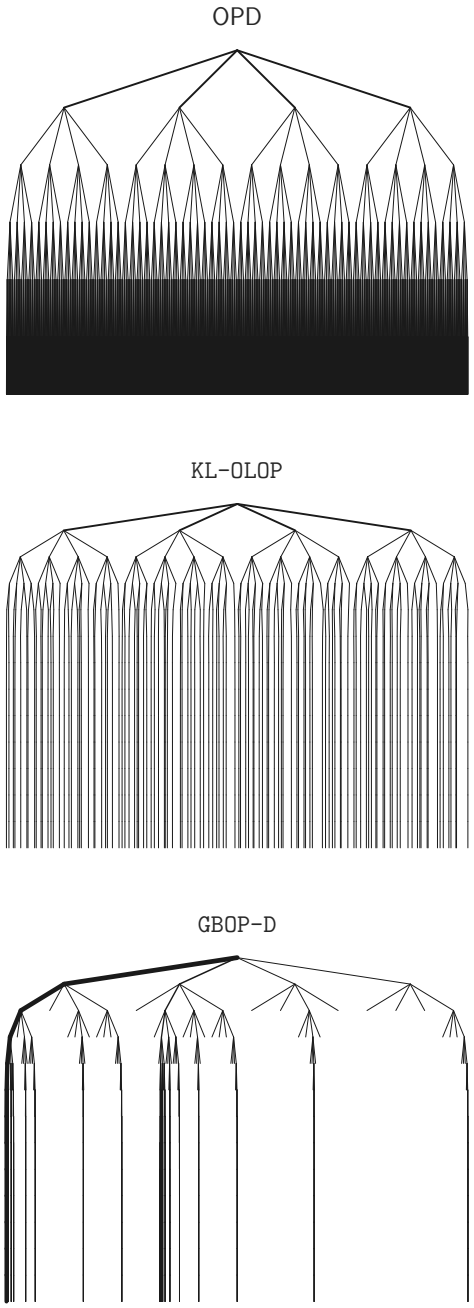


Figure 6.12 – Trees expanded by OPD, by KL-OLOP, and sequences of actions sampled by GBOP-D. The width of edges is proportional to the number of visits.

Chapter 7

Preparing for the Worst

Two roads diverged in a wood, and I–
I took the one less traveled by,
And that has made all the difference.

Robert Frost, *The Road Not Taken*.

Model-based algorithms often assume a *certainty equivalence* to decouple estimation and control: they use a point estimate $\hat{P}$ of the dynamics as if it was the true value. Unfortunately, this *model bias* can significantly degrade the performances. In this chapter, we address this issue by resorting to *robust* decision-making: instead of a mere point estimate, we build a *confidence region* that contains the true dynamics P with high probability, and consider the worst-case outcome with respect to this uncertainty. We propose an integrated framework leveraging non-asymptotic linear regression, interval prediction and tree-based planning to achieve robust stabilisation and minimax control with generic costs; along with an end-to-end analysis.[1]

Contents

[1] This chapter is based on three articles published in the *2019 and 2020 Conferences on Decision and Control* and the *2020 Conference on Neural Information Processing Systems* (Leurent, Denis Efimov, Raissi, et al., 2019; Leurent, Denis Efimov, and Maillard, 2020b; Leurent, Denis Efimov, and Maillard, 2020a).

7.1 Motivation

Remark 7.1 (Change of notations). *So far, we borrowed notations from the Reinforcement Learning community to describe states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$ and transitions $s_{t+1} = P(s_{t+1} \mid s_t, a_t)$. In this chapter, we change conventions and switch to notations from the Control community, which are better suited to describe continuous-time dynamics: states are denoted as $x(t)$, controls as $u(t)$, and dynamics as $\dot{x}(t) = f(x(t), u(t))$. The system dynamics are described in continuous time, but sensing and control are performed in discrete time with time-step $\mathrm{d}t > 0$. For any variable z, we use subscript to refer to these discrete times: $z_n = z(t_n)$ with $t_n = n\mathrm{d}t$ and $n \in \mathbb{N}$. We use bold symbols to denote temporal sequences $\mathbf{z} = (z_n)_{n\in\mathbb{N}}$.*

Model bias Despite the recent successes of Reinforcement Learning (*e.g.* Mnih et al., 2015; Silver, Hubert, et al., 2018), it has hardly been applied in real industrial issues. This could be attributed to two undesirable properties which limit its practical applications. First, it depends on a tremendous amount of interaction data that cannot always be simulated. This issue can be alleviated by model-based methods – which we consider in this part – that often benefit from better sample efficiencies than their model-free counterparts. Second, it relies on trial-and-error and random exploration, which is unacceptable in a critical setting where mistakes are costly and must be avoided at all times.

Since experiencing failures is out of the question, the only way to prevent them from the outset is to rely on some sort of prior knowledge. In this chapter, we assume that the system dynamics are *partially known*, in the form of differential equation with unknown parameters and inputs. More precisely,

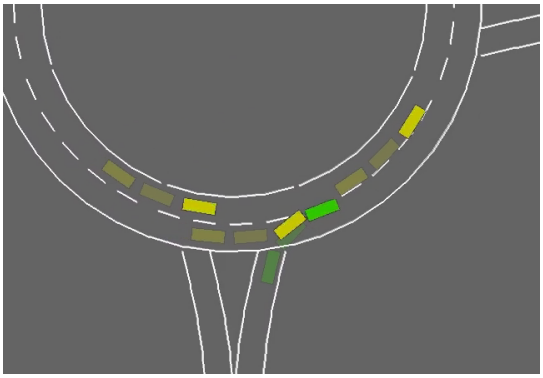
Assumption 7.2 (Structure I). *We consider a dynamical system with state $x \in \mathbb{R}^p$, acted on by controls $u \in \mathbb{R}^q$ and perturbations $\omega \in \mathbb{R}^r$, with dynamics in the form*

$$\dot{x}(t) = f_\theta\left(x(t), u(t), \omega(t)\right),$$

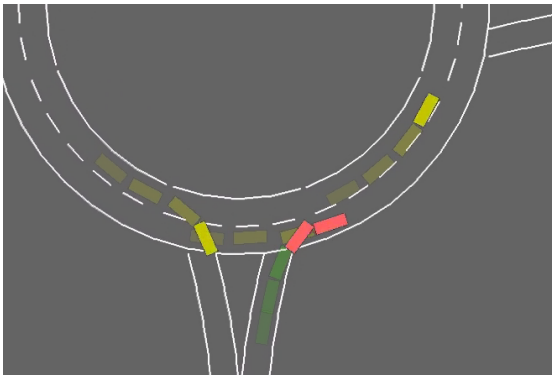
where the dynamics structure $f : \theta \to f_\theta$ is known, *and the parameter vector θ belongs to a known compact set $\Theta \subset \mathbb{R}^d$.*

We argue that this structure assumption is realistic given that most industrial applications to date have been relying on physical models to describe their processes and well-engineered controllers to operate them, rather than machine learning. Our framework relaxes this modelling effort by allowing some *structured uncertainty* around the nominal model.

We adopt a data-driven scheme to estimate the parameters θ more accurately as we interact with the true system. Most model-based reinforcement learning algorithms rely on the estimated dynamics $\hat{\theta}$ to derive the corresponding optimal controls (*e.g.* I. Lenz, Knepper, and Saxena, 2015; Levine, Finn, et al., 2016), but suffer from *model bias*: they ignore the error between the learned and true dynamics, which can dramatically degrade control performances (Doyle, 1978; Schneider, 1997).



(**a**) Optimal control (RL) operates near constraint saturation which produces risky behaviours.



(**b**) As a result, model bias easily causes accidents when predictions are slightly wrong (here by less than half a meter).

Figure 7.1 – Illustration of the issue of model bias when merging into a roundabout.[2]

In particular, this is likely to happen when maximising an objective under constraints, which naturally pushes the system to operate in the vicinity of constraint saturation and makes it prone to failure. For instance, consider the task of merging into a roundabout with flowing traffic, where the agent is rewarded for driving fast while avoiding collision, as defined in (3.6). This formulation is almost equivalent to the problem of maximising speed under a collision-free constraint. When planning with the oracle (true) dynamics, an optimal policy will generally operate as close as possible to the system constraints, *i.e.* it will *brush* other vehicles, resulting in dangerous behaviours as shown in Figure 7.1a. Consequently, when the model predictions are slightly wrong, collisions will happen, as shown in Figure 7.1b. In short, there is no such thing as *continuity* of the optimal policy with respect to the underlying dynamics, especially when the reward function is sharp. A reward-engineering solution to this issue would be to modify the reward function $R(s, a)$ to include a notion of *safety distance*, that the agent would be penalised for not respecting. However, this would require a tedious tuning of the penalty for the possible location and speed of every vehicle nearby, and would not generalise to other situations. In contrast, in this thesis we would rather specify a reward function as simple and straightforward as possible –avoid collisions–, and wish to see the notion of safety distance

[2] An illustrative video is also available at https://www.youtube.com/embed/8khqd3BJo0A?start=3&end=39.

emerge as a by-product of safe decision-making with respect to our uncertainty of other vehicles' behaviours.

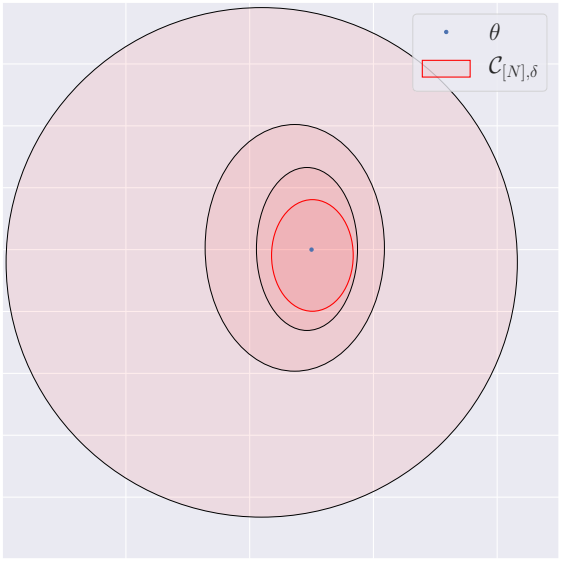


Figure 7.2 – The model estimation procedure. The confidence region $\mathcal{C}_{[N],\delta}$ shrinks with the number of samples N.

Embracing model ambiguity To address the issue of model bias, we turn to the framework of *robust* decision-making: At time step $N \in \mathbb{N}$, instead of merely considering a point estimate of the dynamics, the control scheme needs to rely on an entire *confidence region* $\mathcal{C}_{[N],\delta}$, illustrated in Figure 7.2, that contains the true dynamics parameters with high probability:

$$\mathbb{P}\left(\theta \in \mathcal{C}_{[N],\delta}\right) \geq 1 - \delta, \tag{7.1}$$

where $\delta \in (0, 1]$ is the confidence level. In order to derive an explicit form for this confidence region $\mathcal{C}_{[N],\delta}$, additional assumptions need to be taken regarding the relation between the state transition $\dot{x}$ and its parameter θ. Out of the various hypothesis classes that could represent f_θ, we require one that provides confidence regions for regression. Thus, we make a first assumption that f_θ is linearly dependent on θ, so as to leverage the statistical tools developed for non-asymptotic linear regression.

Assumption 7.3 (Structure II)**.** *We assume that* f_θ *takes the form*

$$\dot{x}(t) = \phi\left(x(t), u(t), \omega(t)\right)\theta + \psi\left(x(t), u(t), \omega(t)\right),$$

where ϕ and ψ are known functions that depend on $x(t), u(t), \omega(t)$ but not θ.

In **Section 7.2**, having observed a history $\mathcal{D}_{[N]} = \{(x_n, y_n, u_n)\}_{n\in[N]}$ of transitions, our first contribution extends the work of Abbasi-Yadkori, Pál, and Szepesvári (2011), who provide a confidence ellipsoid for the least-square estimator, to our setting of feature matrices rather than feature vectors.

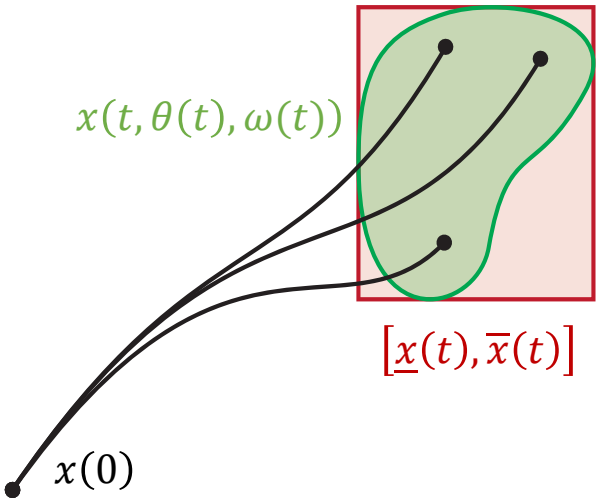


Figure 7.3 – The state prediction procedure. At each time step, we bound the set of reachable states $x(t)$ (in green) under model uncertainty $\mathcal{C}_{[N],\delta}$ inside the interval $[\underline{x}(t), \overline{x}(t)]$ (in red).

Propagation of uncertainty In order to inform the controls, the uncertainty $\mathcal{C}_{[N],\delta}$ about the dynamics needs to be propagated to the induced trajectories. To that end, we wish to derive an interval predictor $[\underline{x}(t), \overline{x}(t)]$ which takes the information on the current state x_N, the confidence region $\mathcal{C}_{[N],\delta}$, a planned control sequence $\mathbf{u}$ and admissible perturbation bounds $[\underline{\omega}(t), \overline{\omega}(t)]$; and verifies the *inclusion property* illustrated in Figure 7.3:

$$\underline{x}(t) \leq x(t) \leq \overline{x}(t), \ \forall t \geq t_N. \tag{7.2}$$

Yet, in order to obtain a closed-form for $[\underline{x}(t), \overline{x}(t)]$, we need to know how the uncertainty over $x(t)$ can affect that of the next state $x(t + dt)$, which requires further specifying the shape of the features ϕ and ψ. Again, we make an assumption that ϕ and ψ are linearly dependent on the states $x(t)$, controls $u(t)$ and perturbations $\omega(t)$, so as to draw on the theory of interval observers for linear systems. Thus, in this chapter we will consider a dynamical system in the following form.

Assumption 7.4 (Structure III)**.** *There exists a known feature tensor $\phi \in \mathbb{R}^{d\times p\times p}$ such that for all $\theta \in \Theta$,*

$$\dot{x}(t) = A(\theta)x(t) + Bu(t) + D\omega(t),\ t \geq 0, \tag{7.3}$$

with

$$A(\theta) = A + \sum_{i=1}^{d} \theta_i \phi_i, \tag{7.4}$$

where $A, \phi_1, \dots, \phi_d \in \mathbb{R}^{p\times p}$ are known.

For all n, we denote $\Phi_n = [\phi_1 x_n \dots \phi_d x_n] \in \mathbb{R}^{p\times d}$. The control matrix $B \in \mathbb{R}^{p\times q}$ and disturbance matrix $D \in \mathbb{R}^{p\times r}$ are known. We also assume access to the observation of $x(t)$ and to a noisy measurement of $\dot{x}(t)$ in the form

$$y(t) = \dot{x}(t) + C\nu(t), \tag{7.5}$$

where $\nu(t) \in \mathbb{R}^s$ is a measurement noise and $C \in \mathbb{R}^{p\times s}$ is known. Assumptions over the disturbance ω and noise ν will be detailed further, and we denote $\eta(t) = C\nu(t) + D\omega(t)$.

In **Section 7.3**, as a second contribution, we derive an *interval predictor* $[\underline{x}(t), \overline{x}(t)]$ for the system (7.3), and analyse its stability.

Robust stabilisation and constraint satisfaction As a third contribution and first application, we consider the problem of robustly stabilising the system (7.3) at a vicinity of the origin under parametric uncertainty and bounded perturbations, while also ensuring that

$$x(t) \in \mathbb{X},\ u(t) \in \mathbb{U} \quad \forall t \geq 0, \tag{7.6}$$

where $[\underline{x}_0, \overline{x}_0] \subset \mathbb{X} \subset \mathbb{R}^p$ and $\mathbb{U} \subset \mathbb{R}^q$ are given bounded constraint sets for the state and the control, respectively. In **Section 7.4**, we introduce a Dual-MPC scheme that achieves this result by relying on a stabilising control for the predicted intervals $[\underline{x}(t), \overline{x}(t)]$.

Minimax control beyond quadratic costs Yet, many tasks cannot be framed as stabilisation problems (*e.g.* obstacle avoidance). To achieve more flexible goals, these tasks can be better formulated with the *minimax control objective* (Ben-Tal, El Ghaoui, and Nemirovski, 2009; Bertsimas, Brown, and Caramanis, 2011; Gorissen, Yanıkoğlu, and den Hertog, 2015), that we consider as a second application. This objective aims to maximise the worst-case return V^r

with respect to the confidence region $\mathcal{C}_{[N],\delta}$ and admissible perturbations

$$\sup_{\mathbf{u}\in(\mathbb{R}^q)^{\mathbb{N}}} \underbrace{\inf_{\substack{\theta\in\mathcal{C}_{[N],\delta}\\ \omega\in[\underline{\omega},\overline{\omega}]^{\mathbb{N}}}} \left[\sum_{n=N+1}^{\infty} \gamma^n R(x_n(\mathbf{u},\omega))\right]}_{V^r(\mathbf{u})}, \tag{7.7}$$

where $x_n(\mathbf{u},\omega)$ is the state reached at step n under controls $\mathbf{u}$ and perturbations ω within the given admissible bounds $[\underline{\omega}(t),\overline{\omega}(t)]$, and R is an *arbitrary* bounded reward function. This choice of rich reward space is crucial to have sufficient flexibility to model non-convex and non-smooth functions that naturally arise in many practical problems involving combinatorial optimisation, branching decisions, etc., while quadratic costs are mostly suited for tracking a fixed reference trajectory (*e.g.* Vinodh Kumar and Jerome, 2013).

Unfortunately, minimax problems such as (7.7) are already notoriously hard when the reward R has a simple form. Without a restriction on the shape of functions R, solving the optimal – not to mention the robust – control objective is intractable and we cannot hope to derive an explicit solution. Thus, in **Section 7.5** we propose a robust MPC algorithm for solving (7.7) numerically. Facing a sequential decision problem with continuous states, we turn to the literature of tree-based planning algorithms studied in Chapter 6. However, these techniques are designed for a single known generative model rather than a confidence region for the system dynamics. We adapt them to the robust objective (7.7) by approximating it with a tractable surrogate $\hat{V}^r$ that exploits the interval predictions $[\underline{x}(t),\overline{x}(t)]$ to define a pessimistic reward $\underline{R}$. In our main result, we show that the best surrogate performance achieved during planning is guaranteed to be attained on the true system, and provide an upper bound for the approximation gap and simple regret of our framework in Theorem 7.30. This is our fourth contribution and the first result of this kind for minimax control with generic costs to the best of our knowledge.

Multi-model extension In **Section 7.6**, our fifth contribution extends the proposed framework to consider multiple modelling assumptions, while narrowing uncertainty through data-driven model rejection, and still ensuring safety via robust model-selection during planning.

Numerical experiments Finally, in **Section 7.7** we demonstrate the applicability of our approach in two numerical experiments: a simple illustrative example and a more challenging simulation for safe autonomous driving on HIGHWAY-ENV.

7.1.1 Related Work

The control of uncertain systems is a long-standing problem, to which a vast body of literature is dedicated. Existing work is mostly concerned with the problem of *stabilisation* around a fixed reference state or trajectory, including approaches such as $\mathcal{H}_\infty$ control (Basar and Bernhard, 1996), sliding-mode control (X.-Y. Lu and Spurgeon, 1997) or system-level synthesis (Dean et al., 2019; Dean et al., 2018). This chapter fits in the popular MPC framework, for which adaptive data-driven schemes have been developed to deal with model uncertainty (Sastry, Bodson, and Bartram, 1990; Tanaskovic et al., 2014; Amos et al., 2018), but lack guarantees. The family of tube-MPC algorithms seeks to derive theoretical guarantees of *robust constraint satisfaction*: as in Section 7.4, the state x and control u are constrained in a safe region $\mathbb{X} \times \mathbb{U}$ around the origin, often chosen convex (Fukushima, Kim, and Sugie, 2007; Adetola and Guay, 2008; Aswani et al., 2013; Turchetta, Berkenkamp, and Krause, 2016; Lorenzen, Allgöwer, and Cannon, 2017; Köhler et al., 2019; X. Lu and Cannon, 2019). Our work in Section 7.4 mainly differs in that it relies on intervals rather than zonotopes, for simplicity of implementation and computational efficiency.

Moreover, as we argue previously, many tasks cannot be framed as stabilisation problems (*e.g.* obstacle avoidance) and are better addressed with the minimax control objective, which can allow more flexible goal formulations. Minimax control has mostly been studied in two particular instances.

Finite states Minimax control of finite Markov Decision Processes with uncertain parameters was studied in (Iyengar, 2005; Nilim and El Ghaoui, 2005; Wiesemann, Kuhn, and Rustem, 2013), who showed that the main results of Dynamic Programming can be extended to their robust counterparts only when the dynamics ambiguity set verifies a certain rectangularity property. Since we consider continuous states, these methods do not apply.

Linear dynamics and quadratic costs Several approaches have been proposed for cumulative regret minimisation in the LQ problem. In the OFU paradigm, the best possible dynamics within a high-confidence region is selected under a controllability constraint, to compute the corresponding optimal control in closed-form by solving a Riccati equation. The results of (Abbasi-Yadkori and Szepesvári, 2011; Ibrahimi, Javanmard, and B. V. Roy, 2012; Faradonbeh, Tewari, and Michailidis, 2020) show that this procedure achieves a $\tilde{\mathcal{O}}\left(N^{1/2}\right)$ regret. Posterior sampling algorithms (Ouyang, Gagrani, and Jain, 2017; Abeille and Lazaric, 2018) select candidate dynamics randomly instead, and obtain the same result. Other works use noise injection for exploration, such as (Dean et al., 2019; Dean et al., 2018). However, neither optimism nor random exploration fits a critical setting, where ensuring safety requires instead to consider pessimistic outcomes. The work of Dean et al. (2019) is close to our setting: after an

offline estimation phase, they estimate a simple regret between a minimax controller and the optimal performance. Our work differs in that it addresses a generic shape cost. Another work of interest is that of Rosolia and Borrelli (2019) where worst-case generic costs are considered. However, they assume knowledge of the dynamics, and their rollout-based solution only produces inner-approximations and does not yield any guarantee. In this chapter, interval prediction is used to produce oversets, while a near-optimal control is found using a tree-based planning procedure.

7.2 Confident model estimation

The goal of this section is to derive a confidence region (7.1) for the parameters θ of the dynamics (7.3).

We abuse notations and define a virtual measurement signal, still denoted $y(t)$, that includes additional known terms

$$y(t) = \dot{x}(t) + C\nu(t) - Ax(t) - Bu(t),$$

to obtain a linear regression system $y_n = \Phi_n\theta + \eta_n$.

Regularised least square To derive an estimate on θ, we consider the weighted L_2-regularised regression problem with weights $\Sigma_p \in \mathbb{R}^{p\times p}$ and parameter $\lambda \in \mathbb{R}_*^+$:

$$\min_{\theta\in\mathbb{R}^d} \sum_{n=1}^{N} \|y_n - \Phi_n\theta\|^2_{\Sigma_p^{-1}} + \lambda\|\theta\|^2. \tag{7.8}$$

The solution can be obtained as:

Proposition 7.5 (Regularised solution)**.** *The solution to* (7.8) *is*

$$\theta_{N,\lambda} \triangleq G_{N,\lambda}^{-1} \sum_{n=1}^{N} \Phi_n^\intercal \Sigma_p^{-1} y_n, \tag{7.9}$$

$$\textit{where} \quad G_{N,\lambda} \triangleq \sum_{n=1}^{N} \Phi_n^\intercal \Sigma_p^{-1} \Phi_n + \lambda I_d \in \mathbb{R}^{d\times d}. \tag{7.10}$$

Proof. We provide a proof in Section D.1.1. □

Substituting y_n into (7.9) yields the regression error

$$\theta_{N,\lambda} - \theta = G_{N,\lambda}^{-1} \sum_{n=1}^{N} \Phi_n^\intercal \Sigma_p^{-1} \eta_n - \lambda G_{N,\lambda}^{-1} \theta. \tag{7.11}$$

To bound this error, we need the noise η_n to concentrate. We assume that

Assumption 7.6 (Bounded disturbance)**.** *At each time $t \geq 0$, the disturbance $\omega(t)$ is enclosed by* known *signals $\underline{\omega}(t) \leq \omega(t) \leq \overline{\omega}(t)$, whose amplitude verifies $\sum_{n=0}^{\infty} \gamma^n C_\omega(t_n) < \infty$, where*

$$C_\omega(t) \triangleq \sup_{\tau \in [0,t]} \|\overline{\omega}(\tau) - \underline{\omega}(\tau)\|_2.$$

Assumption 7.7 (Sub-Gaussian observations)**.** *At each time $t \geq 0$, the combined noise $\eta(t)$ is an independent sub-Gaussian noise with covariance proxy $\Sigma_p \in \mathbb{R}^{p \times p}$:*

$$\forall u \in \mathbb{R}^p,\ \mathbb{E}\left[\exp\left(u^\intercal \eta(t)\right)\right] \leq \exp\left(\frac{1}{2} u^\intercal \Sigma_p u\right).$$

Under these assumptions, we can derive a confidence ellipsoid for θ.

Theorem 7.8 (Confidence ellipsoid, a matricial version of Abbasi-Yadkori, Pál, and Szepesvári, 2011)**.** *Under Assumption 7.7, it holds with probability at least $1 - \delta$ that*

$$\|\theta_{N,\lambda} - \theta\|_{G_{N,\lambda}} \leq \beta_N(\delta), \tag{7.12}$$

with

$$\beta_N(\delta) \triangleq \sqrt{2 \log\left(\frac{\det(G_{N,\lambda})^{1/2}}{\delta \det(\lambda I_d)^{1/2}}\right)} + (\lambda d)^{1/2} \|\theta\|_\infty. \tag{7.13}$$

Proof. We provide a proof in Section D.1.2. □

7.3 State interval prediction

In this section, we view our dynamics (7.3) of Assumption 7.4 as an instance of the more general context of the Linear Parameter-Varying (LPV) representation of the dynamics (J.S. Shamma, 2012; Marcos and Balas, 2004; J. Shamma and Cloutier, 1993; Tan, 1997)

$$\dot{x}(t) = A(\boldsymbol{\theta(t)})x(t) + Bu(t) + D\omega(t),\ t \geq 0, \tag{7.14}$$

where the (known) matrix function $A : \Theta \to \mathbb{R}^{n\times n}$ is only locally bounded (continuous) and does not have to be linear in θ, and where unknown parameters $\theta(t)$ are free to evolve within the known set of admissible values Θ, $\theta \in \mathcal{L}_\infty^d$.

Moreover, we assume that Assumption 7.6 is verified, such that the input $\omega(t)$ belongs to a known bounded interval $[\underline{\omega}(t), \overline{\omega}(t)]$ for all $t \in \mathbb{R}_+$, which is the standard hypothesis for interval estimation (D. Efimov and Raïssi, 2016; Raïssi and D. Efimov, 2018). We also suppose in the following Assumption 7.9 that the system (7.14) generates stable trajectories with a bounded state x for the applied class of inputs $u(t)$, $\omega(t)$, and the initial conditions $x(0)$ are constrained to belong to a given interval $[\underline{x}_0, \overline{x}_0]$.

Assumption 7.9 (Bounded state)**.** *In the system* (7.14)*,* $x \in \mathcal{L}_\infty^n$.

In addition, $x(0) \in [\underline{x}_0, \overline{x}_0]$ *for some known* $\underline{x}_0, \overline{x}_0 \in \mathbb{R}^n$.

Note that since the function A and the set Θ are known, and $\theta \in \Theta$, then there exist matrices $\underline{A}, \overline{A} \in \mathbb{R}^{p\times p}$, which can be easily computed, such that

$$\underline{A} \leq A(\theta) \leq \overline{A}, \quad \forall \theta \in \Theta.$$

The objective of this section is to design an *interval predictor* for the system (7.14), which takes the information on the initial conditions $[\underline{x}_0, \overline{x}_0]$, the admissible bounds on the values of the exogenous input $[\underline{\omega}(t), \overline{\omega}(t)]$, the information about $A(\cdot)$ and Θ (*e.g.* the matrices $\underline{A}, \overline{A}$, but not the instant value of $\theta(t)$) and generates bounded interval estimates $\underline{x}(t), \overline{x}(t) \in \mathbb{R}^p$ such that

$$\underline{x}(t) \leq x(t) \leq \overline{x}(t), \quad \forall t \geq 0. \tag{7.2}$$

In the presence of uncertainty (unknown parameters θ or/and external disturbances $\omega(t)$) the design of a conventional estimator or predictor, approaching the ideal value of the state, can be realised under restrictive assumptions only. However, an interval estimation/prediction remains frequently feasible: using input-output information an algorithm evaluates the set of admissible values (interval) for the state at each instant of time (D. Efimov and Raïssi, 2016; Raïssi and D. Efimov, 2018). The interval length must be minimised via a parametric tuning of the system, and it is typically proportional to the size of the model uncertainty (Chebotarev et al., 2015). It is worth stressing that the interval estimation or prediction is not a relaxation of the original problem, in fact it is an improvement since the interval mean value can be used as the state pointwise estimate, while the interval bounds provide a simultaneous accuracy evaluation for given uncertainty.

There are many approaches to design interval/set-membership estimators and predictors (Jaulin, 2002; Kieffer and Walter, 2004; Olivier Bernard and J.-L. Gouzé, 2004; Moisan, O.

Bernard, and J. Gouzé, 2009), and this section focuses on the design based on the monotone systems theory (Olivier Bernard and J.-L. Gouzé, 2004; Moisan, O. Bernard, and J. Gouzé, 2009; Raïssi, Videau, and Zolghadri, 2010; Raïssi, D. Efimov, and Zolghadri, 2012; D. Efimov, L.M. Fridman, et al., 2012). In such a way the main difficulty for synthesis consists in ensuring cooperativity of the interval error dynamics by a proper design of the algorithm. As it has been shown in (Mazenc and O. Bernard, 2011; Raïssi, D. Efimov, and Zolghadri, 2012; Combastel, 2013), such a complexity of the design can be handled by applying an additional transformation of coordinates, which maps a stable system into a stable and monotone one. An approach for selection of a constant similarity transformation matrix representing a given interval of matrices to an interval of Metzler matrices (providing monotonicity) has been developed in (D. Efimov, Raïssi, Chebotarev, et al., 2013a; Chebotarev et al., 2015).

When designing a predictor, the main difficulty to overcome is the predictor stability, which contrarily to an observer cannot be imposed by a proper design of the gains. An interval inclusion of the uncertain components can be restrictive and transform an initially stable system to an unstable one. In other words, an important problem is to keep the interval predictor stability in the presence of uncertainties and saving the interval inclusion property of the estimates, then an unstable system can definitely enclose the trajectories of a stable one, but at the price of precision. To solve this problem, first, a generic predictor is proposed for an LPV system, whose estimates can be combined with the interval frequency-based estimator presented earlier. To analyse the stability of the predictor, which is modelled by a nonlinear Lipschitz dynamics, a novel non-conservative Lyapunov function is developed, whose features can be verified through solution of linear matrix inequalities (LMIs). Second, we revisit the case of Linear Time-Invariant (LTI) models and demonstrate an asymptotic accuracy improvement that can be achieved if a piece of additional information about external signals is given: the admissible interval of frequency spectrum. Finally, the utility of the developed theory is demonstrated with a safe motion planning application on HIGHWAY-ENV.

The section is organised as follows: in Section 7.3.1, we start by giving an introduction to the theory of interval estimation for LTI systems, before considering a motivating example. An improved interval predictor is presented in Section 7.3.2. The asymptotic accuracy is enhanced using frequency-based estimation in Section 7.3.3. Application of the developed theory to the problem of path planning for an autonomous vehicle is shown in Section 7.3.4.

7.3.1 Preliminaries

We first give some background on interval arithmetic and nonnegative systems, before considering a motivating example.

Interval arithmetic

Lemma 7.10 (D. Efimov, L.M. Fridman, et al., 2012)**.** *Let $x \in \mathbb{R}^n$ be a vector variable, $\underline{x} \leq x \leq \overline{x}$ for some $\underline{x}, \overline{x} \in \mathbb{R}^n$.*

(1) *If $A \in \mathbb{R}^{m\times n}$ is a constant matrix, then*

$$A^+\underline{x} - A^-\overline{x} \leq Ax \leq A^+\overline{x} - A^-\underline{x}. \tag{7.15}$$

(2) *If $A \in \mathbb{R}^{m\times n}$ is a matrix variable and $\underline{A} \leq A \leq \overline{A}$ for some $\underline{A}, \overline{A} \in \mathbb{R}^{m\times n}$, then*

$$\underline{A}^+\underline{x}^+ - \overline{A}^+\underline{x}^- - \underline{A}^-\overline{x}^+ + \overline{A}^-\overline{x}^- \leq Ax \leq \overline{A}^+\overline{x}^+ - \underline{A}^+\overline{x}^- - \overline{A}^-\underline{x}^+ + \underline{A}^-\underline{x}^-. \tag{7.16}$$

Furthermore, if $-\overline{A} = \underline{A} \leq 0 \leq \overline{A}$, then the inequality (7.16) can be simplified:

$$-\overline{A}(\overline{x}^+ + \underline{x}^-) \leq Ax \leq \overline{A}(\overline{x}^+ + \underline{x}^-).$$

Nonnegative systems

Definition 7.11 (Hurwitz)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is called* Hurwitz *if all its eigenvalues have negative real parts.*

Definition 7.12 (Metzler)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is called* Metzler *if all its elements outside the main diagonal are nonnegative.*

Any solution of the LTI system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + D\omega(t),\ t \geq 0, \\ y(t) &= Cx(t) + E\omega(t), \end{aligned} \tag{7.17}$$

with $x(t) \in \mathbb{R}^n$, $y(t) \in \mathbb{R}^p$ and a Metzler matrix $A \in \mathbb{R}^{p\times p}$, is elementwise nonnegative for all $t \geq 0$ provided that $x(0) \geq 0$, $u : \mathbb{R}_+ \to \mathbb{R}^q_+$, $\omega : \mathbb{R}_+ \to \mathbb{R}^r_+$ and $B \in \mathbb{R}^{p\times q}_+$, $D \in \mathbb{R}^{p\times r}_+$ (Farina and Rinaldi, 2000; Smith, 1995). The output solution $y(t)$ is nonnegative if $C \in \mathbb{R}^{s\times p}_+$ and $E \in \mathbb{R}^{s\times r}_+$. Such dynamical systems are called cooperative (monotone) or nonnegative if only initial conditions in $\mathbb{R}^n_+$ are considered (Farina and Rinaldi, 2000; Smith, 1995).

Lemma 7.13 (Raïssi, D. Efimov, and Zolghadri, 2012)**.** *Given the matrices $A \in \mathbb{R}^{p\times p}$, $Y \in \mathbb{R}^{p\times p}$ and $C \in \mathbb{R}^{s\times p}$. If there is a matrix $L \in \mathbb{R}^{p\times s}$ such that the matrices $A - LC$ and Y have the same eigenvalues, then there is a matrix $S \in \mathbb{R}^{p\times p}$ such that $Y = S(A - LC)S^{-1}$ provided that the pairs $(A - LC, \chi_1)$ and (Y, χ_2) are observable for some $\chi_1 \in \mathbb{R}^{1\times p}$, $\chi_2 \in \mathbb{R}^{1\times p}$.*

This result allows to represent the system (7.17) in its nonnegative form via a similarity transformation of coordinates.

Lemma 7.14 (D. Efimov, Raïssi, Chebotarev, et al., 2013a)**.** *Let $D \in \Xi \subset \mathbb{R}^{p\times p}$ be a matrix variable satisfying the interval constraints $\Xi = \{D \in \mathbb{R}^{p\times p} : D_a - \Delta \leq D \leq D_a + \Delta\}$ for some $D_a^T = D_a \in \mathbb{R}^{p\times p}$ and $\Delta \in \mathbb{R}_+^{p\times p}$. If for some constant $\mu \in \mathbb{R}_+$ and a diagonal matrix $\Upsilon \in \mathbb{R}^{p\times p}$ the Metzler matrix $Y = \mu E_{p\times p} - \Upsilon$ has the same eigenvalues as the matrix D_a, then there is an orthogonal matrix $S \in \mathbb{R}^{p\times p}$ such that the matrices $S^T DS$ are Metzler for all $D \in \Xi$ provided that $\mu > p||\Delta||_{max}$.*

In the last lemma, the existence of similarity transformation is proven for an interval of matrices, *e.g.* in the case of LPV dynamics.

A motivating example

Following the result of Lemma 7.10, there is a straightforward solution to the problem:

Proposition 7.15. *Let Assumptions 7.6 and 7.9 be satisfied for the system (7.14), then the interval predictor*

$$\begin{aligned}
\dot{\underline{x}}(t) &= \underline{A}^+\underline{x}^+(t) - \overline{A}^+\underline{x}^-(t) - \underline{A}^-\overline{x}^+(t) + \overline{A}^-\overline{x}^-(t) + Bu(t) + D^+\underline{\omega}(t) - D^-\overline{\omega}(t),\\
\dot{\overline{x}}(t) &= \overline{A}^+\overline{x}^+(t) - \underline{A}^+\overline{x}^-(t) - \overline{A}^-\underline{x}^+(t) + \underline{A}^-\underline{x}^-(t) + Bu(t) + D^+\overline{\omega}(t) - D^-\underline{\omega}(t),\\
&\text{with } \underline{x}(0) = \underline{x}_0,\ \overline{x}(0) = \overline{x}_0,
\end{aligned} \tag{7.18}$$

ensures the inclusion property (7.2).

Proof. The relations (7.2) can easily be obtained by recursively applying Lemma 7.10 at each time step. □

Unfortunately, the stability analysis of the system (7.18) is more tricky. Indeed, (7.18) is a purely nonlinear system (since $\underline{x}^+$, $\underline{x}^-$, $\overline{x}^+$ and $\overline{x}^-$ are globally Lipschitz functions of the state

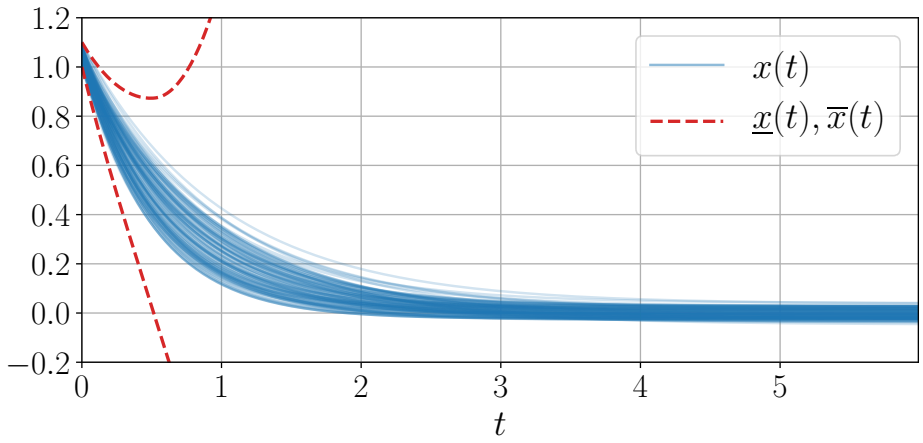


Figure 7.4 – The results of prediction by (7.18): even in such a simplistic setting, the predictor is unstable and diverges quickly.

$\underline{x}$ and $\overline{x}$), whose robust stability with respect to the bounded external inputs $\underline{\omega}$ and $\overline{\omega}$ can be assessed if a suitable Lyapunov function is found. And it is easy to find an example, where the matrices $\underline{A}$ and $\overline{A}$ are stable, but the system (7.18) is not:

Example (motivating)**.** Consider a scalar system

$$\dot{x}(t) = -\theta(t)x(t) + \omega(t),\ t \geq 0,$$

where $x(t) \in \mathbb{R}$ with $x(0) \in [\underline{x}_0, \overline{x}_0] = [1.0, 1.1]$, $\theta(t) \in \Theta = [\underline{\theta}, \overline{\theta}] = [0.5, 1.5]$ and $\omega(t) \in [\underline{\omega}, \overline{\omega}] = [-0.1, 0.1]$ for all $t \geq 0$. Obviously, Assumptions 7.6 and 7.9 are satisfied, and this uncertain dynamics produces bounded trajectories (to prove this consider a Lyapunov function $V(x) = x^2$). Then the interval predictor (7.18) takes the form

$$\begin{aligned}\dot{\underline{x}}(t) &= -\overline{\theta}\overline{x}^+(t) + \underline{\theta}\overline{x}^-(t) + \underline{\omega},\\ \dot{\overline{x}}(t) &= -\underline{\theta}\underline{x}^+(t) + \overline{\theta}\underline{x}^-(t) + \overline{\omega}.\end{aligned}$$

The results of simulation are shown in Figure 7.4. As we can conclude, additional consideration and design are needed to properly solve the posed problem.

7.3.2 Interval predictor design

Note that, in related papers (Ait Rami, Cheng, and Prada, 2008; Raïssi, Videau, and Zolghadri, 2010; Bolajraf, Rami, and Helmke, 2011; D. Efimov, Raïssi, Chebotarev, et al., 2013a; D. Efimov, Raïssi, and Zolghadri, 2013; Chebotarev et al., 2015), various interval observers for LPV systems have been proposed, but in those works the cooperativity and stability of the estimation error dynamics are ensured by a proper selection of observer gains and/or by design of control

algorithms, which can be dependent on $\underline{x}$, $\overline{x}$ and guarantee the observer robust stability. For an interval predictor there is no such a freedom, and a careful selection of hypotheses has to be made in order to provide a desired solution. We will additionally assume the following:

Assumption 7.16. *There exist a matrix $A_0 \in \mathbb{R}^{p\times p}$ and matrices $\Delta A_i \in \mathbb{R}^{p\times p}$, $i \in [M]$ for some $M \in \mathbb{N}_+$ such that the following relations are satisfied for all $\theta \in \Theta$:*

$$A(\theta) = A_0 + \sum_{i=1}^{M} \lambda_i(\theta)\Delta A_i,$$
$$\sum_{i=1}^{M} \lambda_i(\theta) = 1;\ \lambda_i(\theta) \in [0,1],\ i \in [M].$$

Therefore, it is assumed that the matrix $A(\theta)$ for any $\theta \in \Theta$ can be embedded in a polytope defined by M known vertices ΔA_i with the given centre A_0.

To connect this assumption to the results obtained from model estimation in previous section, we can enclose the confidence ellipsoid $\mathcal{C}_{[N],\delta}$ on θ obtained from (7.12) at time $t = t_N$ into a polytope $\mathcal{P}$ on $A(\theta)$. For simplicity, we present here a simple but coarse strategy: bound the ellipsoid by its enclosing sphere, and then the sphere by its enclosing hypercube. We obtain

$$\mathcal{P} = \left\{ A_N + \sum_{i=1}^{M} \lambda_i \Delta A_{N,i} : \lambda \geq 0, \sum_{i=1}^{M} \lambda_i = 1 \right\}, \tag{7.19}$$

where

$$\begin{aligned} A_N &= A + \theta_{N,\lambda}^{\mathsf{T}}\phi, & M &= 2^d \\ \Delta A_{N,i} &= \sqrt{\frac{\beta_N(\delta)}{\lambda_{\max}(G_{N,\lambda})}} h_i{}^{\top}\phi, & h_i &\in \{-1,1\}^d. \end{aligned}$$

Another strategy presented in Section D.2 produces a much tighter polytope, at the price of an increased computational cost required by the diagonalisation of $G_{N,\lambda}$.

In the rest of this section, we will temporarily consider that N is fixed and consider the interval prediction problem for $t \geq t_N$. For simplicity of notations, we will keep denoting A_N and $\Delta A_{N,i}$ as A_0 and δA_i, and $t_N = 0$.

For the given centre A_0 to admit useful properties of nonnegative systems, we will further require that it is Metzler. More precisely, according to the results of Lemmas 7.13 and 7.14, this can be imposed by applying a properly designed similarity transformation, which maps a matrix (interval of matrices) to a Metzler one. Design of such a transformation is not considered in this chapter, and we will just suppose the following:

Assumption 7.17. *There exists a nonsingular matrix $Z \in \mathbb{R}^{p\times p}$ such that $Z^{-1}A_0Z$ is Metzler.*

In practice, this assumption is often verified. It is for instance the case whenever A_0 is diagonalizable, or a method from (D. Efimov, Raïssi, Chebotarev, et al., 2013b) computes a similarity transformation Z when the system is observable with respect to a scalar output. To simplify the notation, we further assume that $Z = I_p$ so that the system (7.14) has already been put in the right form:

$$\dot{x}(t) = [A_0 + \sum_{i=1}^{M} \lambda_i(\theta(t))\Delta A_i]x(t) + Bu(t) + D\omega(t).$$

Denote

$$\Delta A_+ = \sum_{i=1}^{M} \Delta A_i^+,\ \Delta A_- = \sum_{i=1}^{M} \Delta A_i^-,$$

then the following interval predictor can be designed:

Theorem 7.18. *Let Assumptions 7.6, 7.9, 7.16 and 7.17 be satisfied for the system (7.14), then an interval predictor*

$$\begin{aligned}
\dot{\underline{x}}(t) &= A_0\underline{x}(t) - \Delta A_+\underline{x}^-(t) - \Delta A_-\overline{x}^+(t) + Bu(t) + D^+\underline{\omega}(t) - D^-\overline{\omega}(t),\\
\dot{\overline{x}}(t) &= A_0\overline{x}(t) + \Delta A_+\overline{x}^+(t) + \Delta A_-\underline{x}^-(t) + Bu(t) + D^+\overline{\omega}(t) - D^-\underline{\omega}(t),\\
&\textit{with } \underline{x}(0) = \underline{x}_0,\ \overline{x}(0) = \overline{x}_0
\end{aligned} \tag{7.20}$$

ensures the inclusion property (7.2). If there exist diagonal matrices P, Q, Q_+, Q_-, Z_+, Z_-, Ψ_+, Ψ_-, Ψ, $\Gamma \in \mathbb{R}^{2p\times 2p}$ such that the following LMIs are satisfied:

$$\begin{gathered}
P + \min\{Z_+, Z_-\} > 0,\ \Upsilon \preceq 0,\ \Gamma > 0,\\
Q + \min\{Q_+, Q_-\} + 2\min\{\Psi_+, \Psi_-\} > 0,
\end{gathered}$$

where

$$\Upsilon = \begin{bmatrix} \Upsilon_{11} & \Upsilon_{12} & \Upsilon_{13} & P \\ \Upsilon_{12}^\top & \Upsilon_{22} & \Upsilon_{23} & Z_+ \\ \Upsilon_{13}^\top & \Upsilon_{23}^\top & \Upsilon_{33} & Z_- \\ P & Z_+ & Z_- & -\Gamma \end{bmatrix},$$

$$\Upsilon_{11} = \mathcal{A}^\top P + P\mathcal{A} + Q,\ \Upsilon_{12} = \mathcal{A}^\top Z_+ + PR_+ + \Psi_+,$$

$$\Upsilon_{13} = \mathcal{A}^\top Z_- + PR_- + \Psi_-,\ \Upsilon_{22} = Z_+R_+ + R_+^\top Z_+ + Q_+,$$
$$\Upsilon_{23} = Z_+R_- + R_+^\top Z_- + \Psi,\ \Upsilon_{33} = Z_-R_- + R_-^\top Z_- + Q_-,$$
$$\mathcal{A} = \begin{bmatrix} A_0 & 0 \\ 0 & A_0 \end{bmatrix},\ R_+ = \begin{bmatrix} 0 & -\Delta A_- \\ 0 & \Delta A_+ \end{bmatrix},\ R_- = \begin{bmatrix} \Delta A_+ & 0 \\ -\Delta A_- & 0 \end{bmatrix},$$

then the predictor (7.20) *is input-to-state stable with respect to the inputs* $\underline{\omega}$, $\overline{\omega}$.

Note the requirement that the matrix P has to be diagonal is not restrictive, since for a Metzler matrix $\mathcal{A}$, its stability is equivalent to existence of a diagonal solution P of the Lyapunov equation $\mathcal{A}^\top P + P\mathcal{A} \prec 0$ (Farina and Rinaldi, 2000).

Proof. First, let us demonstrate (7.2), to this end note that

$$-\Delta A_i^- \leq \lambda_i \Delta A_i = \lambda_i \Delta A_i^+ - \lambda_i \Delta A_i^- \leq \Delta A_i^+$$

for any $\lambda_i \in [0, 1]$, then using Lemma 7.10 we obtain

$$-\Delta A_i^+ \underline{x}^- - \Delta A_i^- \overline{x}^+ \leq \lambda_i \Delta A_i x \leq \Delta A_i^+ \overline{x}^+ + \Delta A_i^- \underline{x}^-$$

provided that $\underline{x} \leq x \leq \overline{x}$. Hence,

$$-\Delta A_+ \underline{x}^- - \Delta A_- \overline{x}^+ \leq \sum_{i=1}^{N} \lambda_i \Delta A_i x \leq \Delta A_+ \overline{x}^+ + \Delta A_- \underline{x}^-$$

and introducing usual interval estimation errors $\underline{e} = x - \underline{x}$ and $\overline{e} = \overline{x} - x$ and calculating their dynamics we get:

$$\dot{\underline{e}}(t) = A_0 \underline{e}(t) + \underline{r}_1(t) + \underline{r}_2(t),$$
$$\dot{\overline{e}}(t) = A_0 \overline{e}(t) + \overline{r}_1(t) + \overline{r}_2(t),$$

where

$$\underline{r}_1 = \sum_{i=1}^{M} \lambda_i \Delta A_i x + \Delta A_+ \underline{x}^- + \Delta A_- \overline{x}^+,$$
$$\underline{r}_2 = D\omega - D^+ \underline{\omega} + D^- \overline{\omega},$$
$$\overline{r}_1 = -\sum_{i=1}^{M} \lambda_i \Delta A_i x + \Delta A_+ \overline{x}^+ + \Delta A_- \underline{x}^-,$$
$$\overline{r}_2 = D^+ \overline{\omega} - D^- \underline{\omega} - D\omega.$$

Non-negativity or $\underline{r}_2$ and $\overline{r}_2$ follows from Assumption 7.6 and Lemma 7.10. The signals $\underline{r}_1$ and $\overline{r}_1$ are also nonnegative provided that (7.2) holds and due to the calculations above. Note that the relations (7.2) are satisfied for $t = 0$ by construction and Assumption 7.9, then since the matrix A_0 is Metzler by Assumption 7.16, we have that $\dot{\underline{e}}_i(0) \in \mathbb{R}^p_+$ or $\dot{\overline{e}}_i(0) \in \mathbb{R}^p_+$ provided that $e_i(0) = 0$ or $e_i(0) = 0$, respectively, for any $i \in [p]$ (the error cannot become negative). Next, repeating these arguments it is possible to show that $\underline{e}(t) \geq 0$ and $\overline{e}(t) \geq 0$ for all $t \geq 0$ (Smith, 1995), which confirms the relations (7.2).

Second, let us consider the stability of (7.20), and for this purpose define the extended state vector as $X = [\underline{x}^\top \ \ \overline{x}^\top]^\top$, whose dynamics admit the differential equation

$$\dot{X}(t) = \mathcal{A}X(t) + R_+X^+(t) - R_-X^-(t) + \delta(t),$$

where

$$\delta(t) = \begin{bmatrix} -D^- & D^+ \\ D^+ & -D^- \end{bmatrix} \begin{bmatrix} \overline{\omega}(t) \\ \underline{\omega}(t) \end{bmatrix}.$$

is a bounded input vector, whose norm is proportional to $\underline{\omega}, \overline{\omega}$. Consider a candidate Lyapunov function

$$\begin{aligned} V(X) &= X^\top PX + X^\top Z_+X^+ - X^\top Z_-X^- \\ &= \sum_{k=1}^{2n} P_{k,k}X_k^2 + (Z_+)_{k,k}X_kX_k^+ - (Z_-)_{k,k}X_kX_k^- \\ &= \sum_{k=1}^{2n} P_{k,k}X_k^2 + (Z_+)_{k,k}|X_k|X_k^+ + (Z_-)_{k,k}|X_k|X_k^-, \end{aligned}$$

which is positive definite provided that

$$P + \min\{Z_+, Z_-\} > 0,$$

and whose derivative for the system dynamics takes the form

$$\begin{aligned} \dot{V} &= 2\dot{X}^\top PX + 2\dot{X}^\top Z_+X^+ - 2\dot{X}^\top Z_-X^- \\ &= \begin{bmatrix} X \\ X^+ \\ -X^- \\ \delta \end{bmatrix}^\top \Upsilon \begin{bmatrix} X \\ X^+ \\ -X^- \\ \delta \end{bmatrix} - X^\top QX - (X^+)^\top Q_+X^+ \\ &\quad - (X^-)^\top Q_-X^- - 2(X^+)^\top \Psi X^- - 2(X^+)^\top \Psi_+X \\ &\quad - 2(-X^-)^\top \Psi_-X + \delta^\top \Gamma\delta. \end{aligned}$$

Note that

$$\begin{aligned}(X^+)^\top \Psi X^- &= 0,\\ (X^+)^\top \Psi_+ X &\geq 0,\\ (-X^-)^\top \Psi_- X &\geq 0\end{aligned}$$

for any diagonal matrix Ψ and

$$\Psi_+ \geq 0,\ \Psi_- \geq 0.$$

Hence, if $\Upsilon \preceq 0$, as it is assumed in the theorem, we obtain that

$$\begin{aligned}\dot{V} \leq & -X^\top Q X - (X^+)^\top Q_+ X^+ - (X^-)^\top Q_- X^-\\ & -2(X^+)^\top \Psi_+ X - 2(-X^-)^\top \Psi_- X + \delta^\top \Gamma \delta\\ \leq & -X^\top \Omega X + \delta^\top \Gamma \delta,\end{aligned}$$

where

$$\Omega = Q + \min\{Q_+, Q_-\} + 2\min\{\Psi_+, \Psi_-\} > 0$$

is a diagonal matrix. The substantiated properties of V and its derivative imply that (7.20) is input-to-state stable (Khalil, 2002) with respect to the input δ (or, by its definition, with respect to $(\underline{\omega}, \overline{\omega})$). □

Remark 7.19 (Global constraints). *The LMIs of the above theorem are not conservative, since the restriction on positive definiteness of involved matrix variables is not imposed on all of them separately, but on their combinations:*

$$\begin{aligned}P + \min\{Z_+, Z_-\} > 0,\ \Gamma > 0,\\ Q + \min\{Q_+, Q_-\} + 2\min\{\Psi_+, \Psi_-\} > 0,\end{aligned}$$

then some of them can be sign-indefinite or negative-definite ensuring the fulfilment of the last inequality $\Upsilon \preceq 0$.

Remark 7.20 (Asymptotic linearity). *Assume that* $-\underline{\omega} = \overline{\omega} = const \neq 0$ *and the conditions of Theorem 7.18 are satisfied, then asymptotically* $\underline{x}$ *and* $\overline{x}$ *are negative and positive, respectively. Therefore, the dynamics of* (7.20) *takes the form for sufficiently high values of* $t \geq 0$*:*

$$\begin{aligned}\dot{\underline{x}}(t) &= (A_0 - \Delta A_+)\underline{x}(t) - \Delta A_- \overline{x}(t) + Bu(t) + D^+ \underline{\omega} - D^- \overline{\omega},\\ \dot{\overline{x}}(t) &= (A_0 + \Delta A_+)\overline{x}(t) + \Delta A_- \underline{x}(t) + Bu(t) + D^+ \overline{\omega} - D^- \underline{\omega},\end{aligned}$$

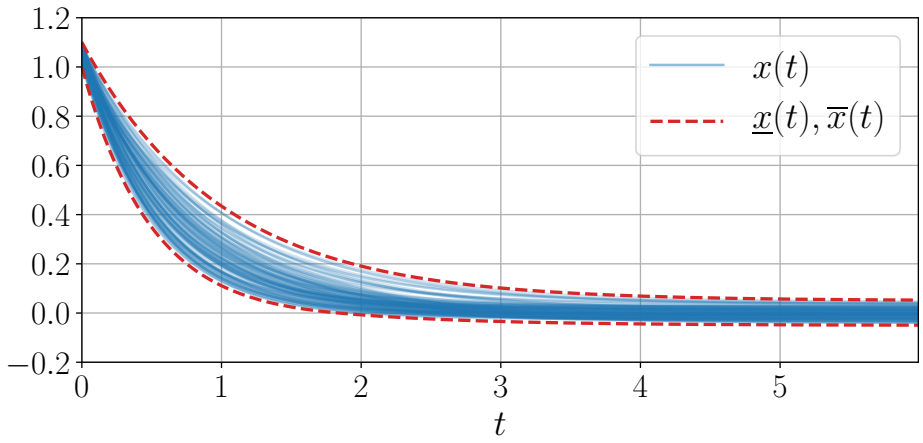


Figure 7.5 – The results of prediction by (7.20): the new predictor is stable and produces tight bounds.

which is a linear system

$$\dot{X}(t) = \begin{bmatrix} A_0 - \Delta A_+ & -\Delta A_- \\ \Delta A_- & A_0 + \Delta A_+ \end{bmatrix} X(t) + \begin{bmatrix} B \\ B \end{bmatrix} u(t) + \begin{bmatrix} -D^- & D^+ \\ D^+ & -D^- \end{bmatrix} \begin{bmatrix} \overline{\omega} \\ \underline{\omega} \end{bmatrix}, \quad (7.21)$$

where as before $X = [\underline{x}^\top \ \overline{x}^\top]^\top$.

Example (motivating, continue)**.** Let us apply the predictor (7.20) to the motivating example:

$$\begin{aligned} \dot{\underline{x}}(t) &= -\overline{\theta}\underline{x}(t) - (\overline{\theta} - \underline{\theta})\underline{x}^-(t) + \underline{\omega}, \\ \dot{\overline{x}}(t) &= -\overline{\theta}\overline{x}(t) + (\overline{\theta} - \underline{\theta})\overline{x}^+(t) + \overline{\omega}, \end{aligned}$$

where $A_0 = -\overline{\theta}$ is chosen, then $\Delta A_+ = \overline{\theta} - \underline{\theta}$, $\Delta A_- = 0$ and all conditions of Theorem 7.18 are verified. The results of simulation are shown in Figure 7.5. As we can see the new predictor generates very reasonable and bounded estimates.

7.3.3 Frequency-based interval estimation

The domain of convergence of the linear system (7.21), and hence of (7.20), can be tightened under an additional hypothesis that $\omega(t)$ has a known and bounded frequency spectrum. Assume that there exist two signals $\underline{\omega}, \overline{\omega} : \mathbb{R}_+ \to \mathbb{R}^r$ and two vectors $\underline{x}_0, \overline{x}_0 \in \mathbb{R}^p$ such that

$$\begin{aligned} \underline{\omega}(t) \leq \omega(t) \leq \overline{\omega}(t), \quad \forall t \geq 0, \\ \underline{x}_0 \leq x(0) \leq \overline{x}_0, \end{aligned}$$

and there is a matrix $L \in \mathbb{R}^{p\times s}$ such that $A - LC$ is Hurwitz and Metzler, then an interval observer for the system (7.17) can be written as follows (Raïssi, D. Efimov, and Zolghadri, 2012):

$$\begin{aligned}
\dot{\underline{x}}(t) &= (A-LC)\underline{x}(t) + Bu(t) + Ly(t) + (D-LE)^{+}\underline{\omega}(t) - (D-LE)^{-}\overline{\omega}(t),\\
\dot{\overline{x}}(t) &= (A-LC)\overline{x}(t) + Bu(t) + Ly(t) + (D-LE)^{+}\overline{\omega}(t) - (D-LE)^{-}\underline{\omega}(t),\\
&\text{with } \underline{x}(0) = \underline{x}_0,\ \overline{x}(0) = \overline{x}_0,
\end{aligned} \tag{7.22}$$

guaranteeing the desired interval relations

$$\underline{x}(t) \leq x(t) \leq \overline{x}(t), \quad \forall t \geq 0.$$

This solution uses only information about amplitude of the external input ω, and its precision can be largely improved if we assume that there is also information about admissible frequency spectrum in ω:

Lemma 7.21. *Let there exist $f_1, f_2 \in \mathbb{N}_+$ and $T, W > 0$ such that*

$$\omega(t) = a_0 + \sum_{f=f_1}^{f_2} a_f \sin\left(\frac{2\pi f}{T}t + \phi_s\right),$$

for some $a_0, a_f, \phi_f \in \mathbb{R}$ with $f \in [f_1, f_2]$ and $||d|| \leq W$. Then for any $x(0) \in \mathbb{R}^n$ in (7.17) and any $\varepsilon > 0$ there exists $\tau > 0$ such that

$$|x_i(t)| \leq \sup_{w\in[\frac{2\pi}{T}f_1, \frac{2\pi}{T}f_2]} |e_i(jwI_n - A)^{-1}DE_r|W + \varepsilon \quad \forall t \geq \tau,$$

where j corresponds to the imaginary unit, provided that the matrix A is Hurwitz.

Proof. The solution of the system (7.17) can be written as follows:

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\sigma)}(Bu(\sigma) + D\omega(\sigma))\mathrm{d}\sigma,$$

where the first term $(e^{At}x(0))$ is converging asymptotically to zero since the matrix A is Hurwitz by hypothesis. And in order to estimate the second term, the Bode magnitude plot can be used, which provides the asymptotic amplitude of the state for the given frequency input. Under the introduced hypotheses, the frequency of the input lies in the interval $[\frac{2\pi}{T}f_1, \frac{2\pi}{T}f_2]$ and the amplitude is upper-bounded by W, then there exist constants $\tau > 0$ and $\varepsilon > 0$, related with $x(0)$, such that the claim of the lemma is true. □

The interval observer (7.22), if we assume that $\overline{\omega}(t) = -\underline{\omega}(t) = WE_r$ and $L = 0$, asymptotically will converge to the interval $[-|e_i A^{-1} B E_r| W, |e_i A^{-1} B E_r| W]$ (that corresponds to the result of Lemma 7.21 with $f_1 = f_2 = 0$), which is the estimate from Bode plot given for the frequency 0, and it is a well-known fact that for many stable systems the Bode magnitude plot is a decreasing function of the frequency. Therefore, if the information about frequency spectrum is known and it is separated from zero, then the asymptotic interval accuracy can be significantly improved. Of course, Lemma 7.21 can be applied iteratively for a decreasing sequence of $\varepsilon > 0$ and an increasing one in $\tau > 0$.

Example. Let us illustrate these conclusions on a simple example:

$$A = \begin{bmatrix} -1 & 1 \\ 0.1 & -1 \end{bmatrix}, \; B = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \; L = 0,$$
$$\overline{\omega}(t) = -\underline{\omega}(t) = 1,$$
$$\overline{x}_0 = [2 \; 1]^\top, \; \underline{x}_0 = [-1 \; -2]^\top.$$

And assume that $\omega(t) = d_0 \sin(ft + \phi_0), \; f = 7$, then the system trajectories and intervals are shown in Figure 7.6.

7.3.4 Prediction for a self-driving vehicle

We consider the problem of safe decision-making for autonomous highway driving (Leurent, 2018)[3].

As described in Chapter 3, an autonomous vehicle is driving on a highway populated with N other agents, and uses Model Predictive Control to plan a sequence of decisions. To that end, it relies on parametrized dynamical model for each agent to predict the future trajectory of each traffic participant:

$$\dot{x}_i = f_i(x, \theta_i), \; i \in [N_v],$$

where f_i are described in Chapter 3, $x_i \in \mathbb{R}^4$ is the state of a vehicle, $x = [x_1^\top, \dots, {x_{N_v}}^\top]^\top \in \mathbb{R}^{4N_v}$ and $\theta_i \in \mathbb{R}^5$ is the corresponding vector of unknown parameters. Crucially, this system describes the couplings and interactions between vehicles, so that the autonomous agent can properly anticipate their reactions. However, we assume that we do not have access to the true values of the behavioural parameters $\theta = [\theta_1, \dots, \theta_{N_v}]^\top$; instead, we merely know that these parameters lie in a set of admissible values $\Theta \subset \mathbb{R}^{5N_v}$. In order to act safely in the face of uncertainty, the agent must consider all possible vehicle trajectories in order to take its decisions. In this section, we focus on how to compute these trajectory enclosures through interval prediction.

[3] Videos and source code of all experiments are available at https://eleurent.github.io/interval-prediction/.

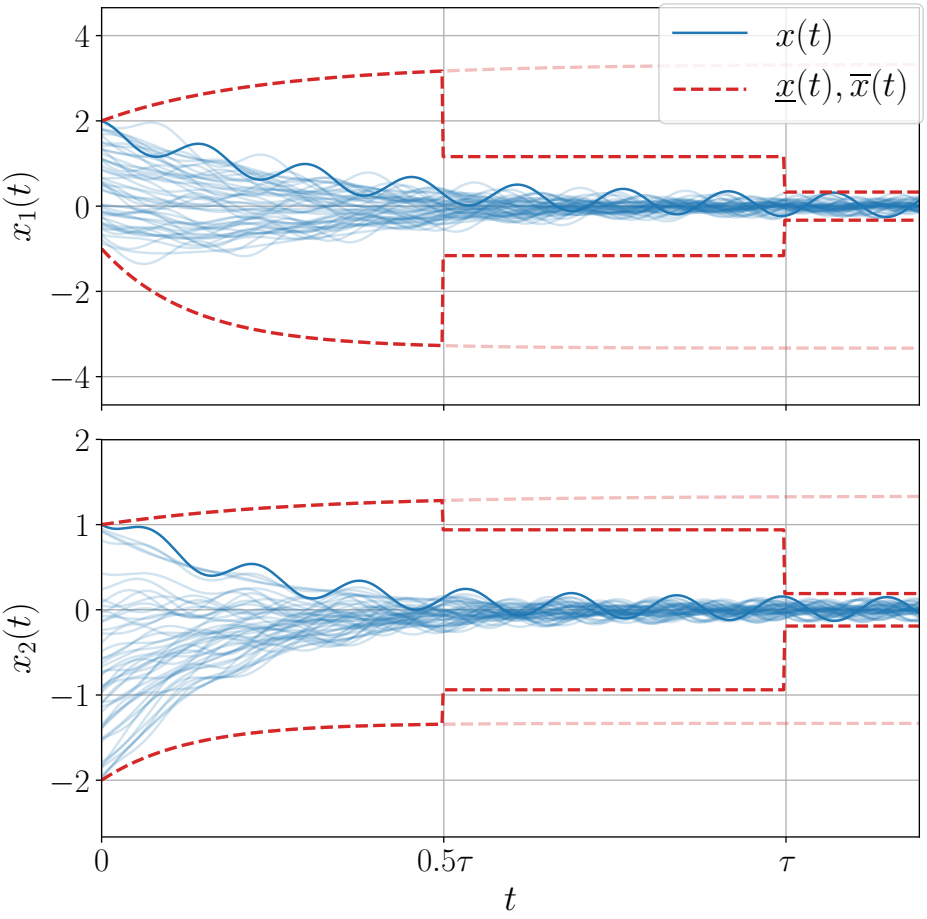


Figure 7.6 – The results of prediction for different values of the frequency. Taking $\tau = \frac{4}{\min_{i=\overline{1,n}} |\lambda_i(A)|} = 5.85$ and $\varepsilon = 0.05$, the trajectories of the interval observer (7.22) are presented for $t \leq 0.5\tau$, and as we can conclude, these estimates are rather conservative. Next, for $t \in [0.5\tau, \tau]$ the estimates given in Lemma 7.21 for the case $s_1 = s_2 = 0$ are shown, which are already more accurate. Finally, for $t \geq \tau$ the estimates of Lemma 7.21 are presented for $s_1 = s_2 = s$, which demonstrate a definite improvement.

LPV formulation

The system presented in Chapter 3 is non-linear and must be cast into the LPV form. We approximate the non-linearities induced by the trigonometric operators through equilibrium linearisation around $y_i = y_{L_i}$ and $\psi_i = \psi_{L_i}$.

This yields the following longitudinal dynamics:

$$\begin{aligned}\dot{p}_i^x &= v_i,\\ \dot{v}_i &= \theta_{i,1}(v_0 - v_i) + \theta_{i,2}(v_{f_i} - v_i) + \theta_{i,3}(p_{f_i}^x - p_i^x - d_0 - v_i T),\end{aligned}$$

where $\theta_{i,2}$ and $\theta_{i,3}$ are set to 0 whenever the corresponding features are not active.

It can be rewritten in the form

$$\dot{x} = A(\theta)(x - x_c) + d.$$

For example, in the case of two vehicles only,

$$x = \begin{bmatrix} p_i^x \\ p_{f_i}^x \\ v_i \\ v_{f_i} \end{bmatrix}, \quad x_c = \begin{bmatrix} -d_0 - v_0 T \\ 0 \\ v_0 \\ v_0 \end{bmatrix}, \quad d = \begin{bmatrix} v_0 \\ v_0 \\ 0 \\ 0 \end{bmatrix}$$

$$A(\theta) = \begin{array}{c} \\ i \\ f_i \\ i \\ f_i \end{array} \begin{array}{c} \begin{array}{cccc} i & f_i & i & f_i \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\theta_{i,3} & \theta_{i,3} & -\theta_{i,1} - \theta_{i,2} - \theta_{i,3} & \theta_{i,2} \\ 0 & 0 & 0 & -\theta_{f_i,1} \end{bmatrix} \end{array}$$

The lateral dynamics are in a similar form:

$$\begin{bmatrix} \dot{p}_i^y \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} 0 & v_i \\ -\frac{\theta_{i,4}\theta_{i,5}}{v_i} & -\theta_{i,5} \end{bmatrix} \begin{bmatrix} p_i^y - p_{L_i}^y \\ \psi_i - \psi_{L_i} \end{bmatrix} + \begin{bmatrix} v_i \psi_{L_i} \\ 0 \end{bmatrix}$$

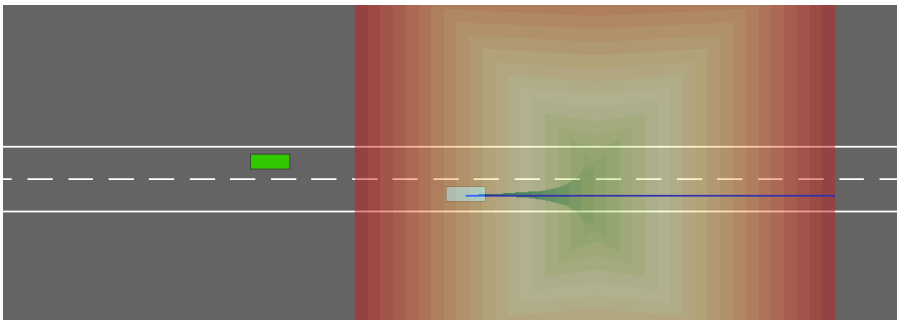
Here, the dependency in v_i is seen as an uncertain parametric dependency, *i.e.* $\theta_{i,6} = v_i$, with constant bounds assumed for v_i using an overset of the longitudinal interval predictor.

Change of coordinates

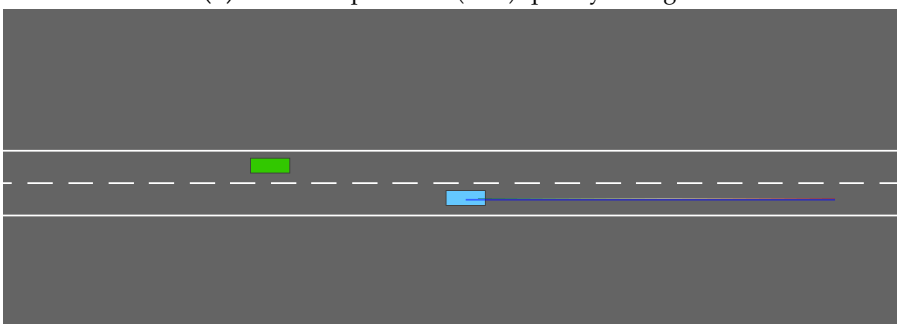
In both cases, the obtained polytope centre A_0 is non-Metzler. We use Lemmas 7.13 and 7.14 to compute a similarity transformation of coordinates. Precisely, we ensure that the polytope is chosen so that its centre A_0 is diagonalisable having real eigenvalues, and perform an eigendecomposition to compute its change of basis matrix S. The transformed system $x' = S^{-1}(x - x_c)$ verifies Assumption 7.16 as required to apply the interval predictor of Theorem 7.18. Finally, the obtained predictor is transformed back to the original coordinates x by using the interval arithmetic of Lemma 7.10.

Results

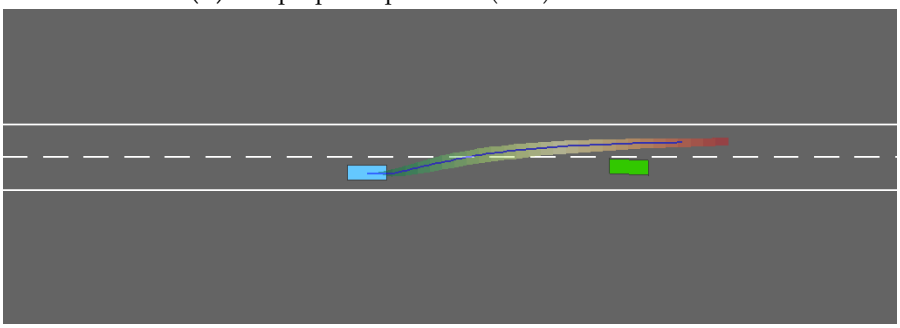
We show the resulting intervals in Figure 7.7. The target vehicle with uncertain behaviour is in blue, while the ego-vehicle is in green. Its trajectory interval is computed over a duration of two seconds and represented by an area filled with a colour gradient representing time. The ground-truth trajectory is shown in blue. In Figure 7.7a, we observe that the direct predictor (7.18) is unstable and quickly diverges to cover the whole road, thus hindering any sensible decision-making. In prior work (Leurent, Blanco, et al., 2018), we had to circumvent this issue by subdividing Θ and $[\underline{x}, \overline{x}]$ to reduce the initial overestimations and merely delay the divergence (Adrot and Flaus, 2003), at the price of a heavy computational load. In stark contrast, we see in Figure 7.7b that the novel predictor (7.20) is very stable even over long horizons, which allows the ego-vehicle to plan an overtaking manoeuvre. Until then, there was little uncertainty in the predicted trajectory for the target vehicle was isolated, but as the ego-vehicle cuts into its lane in Figure 7.7c, we start seeing the effects of uncertain interactions between the two vehicles, in both longitudinal and lateral directions. Note that this formulation naturally exhibits socially aware predictions, and accounts for these interactions. Our framework is also quite flexible in representing different assumptions on the admissible behaviours. For instance, we show in Figure 7.7d a simulation in which we model right-hand traffic where drivers are expected to keep to the rightmost lane. In such a situation, it is reasonable to assume that in the absence of any obstacle in front, a vehicle driving on the middle lane will either stay there or return to the right lane, but has no incentive to change to the left lane. This simple assumption on L_i can easily be incorporated in the interval predictor, and enables the emergence of a realistic behaviour when running the robust decision-making procedure: the ego-vehicle cannot pass another vehicle by its right side, and can only overtake it by its left side. These behaviours displaying safe reasoning under uncertainty are showcased in the attached videos.



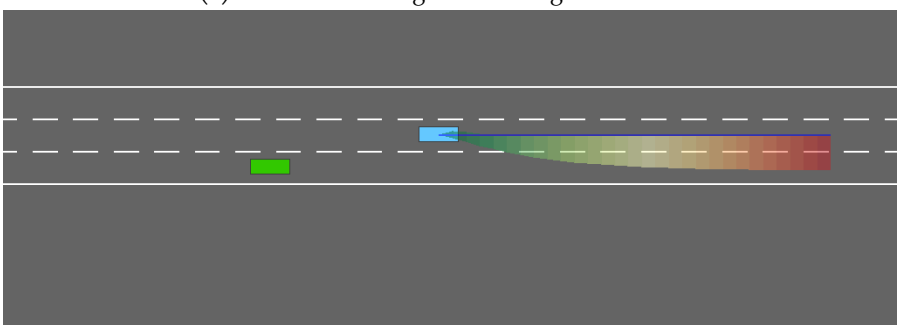
(**a**) The naive predictor (7.18) quickly diverges



(**b**) The proposed predictor (7.20) remains stable



(**c**) Prediction during a lane change manoeuvre



(**d**) Prediction with uncertainty in the followed lane L_i

Figure 7.7 – State intervals obtained by the two methods in different conditions.

Section conclusion

The prediction problem for uncertain LPV systems is solved by designing an interval predictor, which is described by nonlinear differential equations, and whose stability is evaluated using a new Lyapunov function. The corresponding robust stability conditions are expressed in terms of LMIs. An approach is presented to improve the asymptotic accuracy of interval estimation or prediction in LTI systems provided that the exogenous inputs have a known spectrum of frequencies. The proficiency of the methods is demonstrated in application to a problem of safe motion planning for a self-driving car.

7.4 Robust stabilisation and constraint satisfaction

In this section, as a first application of the estimation and prediction tools introduced above, we set out to design a robust control $u(t)$ that stabilises (7.3), (7.5) at a vicinity of the origin under Assumptions 7.4, 7.6 and 7.9 such that

$$x(t) \in \mathbb{X},\ u(t) \in \mathbb{U} \quad \forall t \geq 0, \tag{7.6}$$

where $[\underline{x}_0, \overline{x}_0] \subset \mathbb{X} \subset \mathbb{R}^p$ and $\mathbb{U} \subset \mathbb{R}^q$ are given bounded constraint sets for the state and the control, respectively.

7.4.1 Stabilising control for (7.18) and (7.20)

Note that both interval predictors, (7.18) and (7.20), admit a representation in the form

$$\dot{\xi}(t) = \mathcal{A}_0\xi(t) + \mathcal{A}_1\xi^+(t) + \mathcal{A}_2\xi^-(t) + \mathcal{B}u(t) + \delta(t), \tag{7.23}$$

where $\xi(t) = [\underline{x}^\top(t)\ \overline{x}^\top(t)]^\top \in \mathbb{R}^{2p}$ is the extended state vector of the predictors,

$$\delta(t) = \begin{bmatrix} D^+ & -D^- \\ -D^- & D^+ \end{bmatrix} \begin{bmatrix} \underline{\omega}(t) \\ \overline{\omega}(t) \end{bmatrix} \in \mathbb{R}^{2p}$$

is the external known input, $\mathcal{B} = [B^\top\ B^\top]^\top$,

$$\mathcal{A}_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},\ \mathcal{A}_1 = \begin{bmatrix} \underline{A}^+ & -\underline{A}^- \\ -\overline{A}^- & \overline{A}^+ \end{bmatrix},\ \mathcal{A}_2 = \begin{bmatrix} -\overline{A}^+ & \overline{A}^- \\ \underline{A}^- & -\underline{A}^+ \end{bmatrix} \quad \text{for (7.18)},$$

and

$$\mathcal{A}_0 = \begin{bmatrix} A_0 & 0 \\ 0 & A_0 \end{bmatrix},\ \mathcal{A}_1 = \begin{bmatrix} 0 & -\Delta A_- \\ 0 & \Delta A_+ \end{bmatrix},\ \mathcal{A}_2 = \begin{bmatrix} -\Delta A_+ & 0 \\ \Delta A_- & 0 \end{bmatrix} \quad \text{for (7.20)}.$$

Note that (7.23) is a nonlinear system due to the presence of globally Lipschitz nonlinearities $\xi^+(t)$ and $\xi^-(t)$.

Due to the inclusion property (7.2), the boundedness of $\xi(t)$ implies the same property of $x(t)$. Therefore, in order to regulate (7.3) it is required to design a state feedback $u(t)$ minimizing the asymptotic amplitude of the state $\xi(t)$ for given input $\delta(t)$ (D. Efimov, Raïssi, and Zolghadri, 2013). In other words, it is necessary to design a control $u(t)$ that input-to-state stabilises (7.23). It is proposed to look for such a control in the form

$$u(t) = K_0\xi(t) + K_1\xi^+(t) + K_2\xi^-(t) + S\delta(t), \tag{7.24}$$

where $K_0, K_1, K_2 \in \mathbb{R}^{q\times 2p}$ and $S \in \mathbb{R}^{q\times 2p}$ are the gains to be designed ((7.24) contains a nonlinear feedback). The selection of S is simple, it has to minimise the norm of $\mathcal{B}S + I_{2p}$, and it can be made independently of K_0, K_1, K_2. Therefore, denoting $\tilde{\delta}(t) = (\mathcal{B}S + I_{2p})\delta(t)$ the closed-loop system (7.23), (7.24) takes the form:

$$\dot{\xi}(t) = \mathcal{D}_0\xi(t) + \mathcal{D}_1\xi^+(t) + \mathcal{D}_2\xi^-(t) + \tilde{\delta}(t), \tag{7.25}$$

where $\mathcal{D}_i = \mathcal{A}_i + \mathcal{B}K_i$ for $i \in [3]$, and the restrictions, which the gains K_0, K_1, K_2 have to respect, are given below:

Theorem 7.22. *If there exist diagonal matrices $P, Q, Q_+, Q_-, Z_+, Z_-, \Psi_+, \Psi_-, \Psi, \Gamma \in \mathbb{R}^{2p\times 2p}$ such that the following linear matrix inequalities are satisfied:*

$$P + \min\{Z_+, Z_-\} > 0,\ \Upsilon \preceq 0,\ \Gamma > 0,$$
$$Q + \min\{Q_+, Q_-\} + 2\min\{\Psi_+, \Psi_-\} > 0,$$

where

$$\Upsilon = \begin{bmatrix} \Upsilon_{11} & \Upsilon_{12} & \Upsilon_{13} & P \\ \Upsilon_{12}^\top & \Upsilon_{22} & \Upsilon_{23} & Z_+ \\ \Upsilon_{13}^\top & \Upsilon_{23}^\top & \Upsilon_{33} & -Z_- \\ P & Z_+ & -Z_- & -\Gamma \end{bmatrix},$$
$$\Upsilon_{11} = \mathcal{D}_0^\top P + P\mathcal{D}_0 + Q,\ \Upsilon_{12} = \mathcal{D}_0^\top Z_+ + P\mathcal{D}_1 + \Psi_+,$$
$$\Upsilon_{13} = P\mathcal{D}_2 - \mathcal{D}_0^\top Z_- - \Psi_-,\ \Upsilon_{22} = Z_+\mathcal{D}_1 + \mathcal{D}_1^\top Z_+ + Q_+,$$
$$\Upsilon_{23} = Z_+\mathcal{D}_2 - \mathcal{D}_1^\top Z_- + \Psi,\ \Upsilon_{33} = -Z_-\mathcal{D}_2 - \mathcal{D}_2^\top Z_- + Q_-,$$

then (7.25) is input-to-state stable with respect to $\underline{\omega}, \overline{\omega}$.

Note that the requirement that the matrix P has to be diagonal is not restrictive, since for a Metzler matrix $\mathcal{D}_0$ (the case of (7.18) and (7.20)), its stability is equivalent to existence of a diagonal solution P of the Lyapunov equation $\mathcal{D}_0^\top P + P\mathcal{D}_0 \prec 0$ (Farina and Rinaldi, 2000).

Proof. Consider a candidate Lyapunov function

$$\begin{aligned} V(\xi) &= \xi^\top P \xi + \xi^\top Z_+ \xi^+ - \xi^\top Z_- \xi^- \\ &= \sum_{k=1}^{2p} P_{k,k}\xi_k^2 + (Z_+)_{k,k}\xi_k\xi_k^+ - (Z_-)_{k,k}\xi_k\xi_k^- \\ &= \sum_{k=1}^{2p} P_{k,k}\xi_k^2 + (Z_+)_{k,k}|\xi_k|\xi_k^+ + (Z_-)_{k,k}|\xi_k|\xi_k^-, \end{aligned}$$

which is positive definite provided that

$$P + \min\{Z_+, Z_-\} > 0$$

since all terms in V are quadratic-like, and whose derivative for the system (7.25) dynamics takes the form

$$\begin{aligned} \dot{V} &= 2\dot{\xi}^\top P\xi + 2\dot{\xi}^\top Z_+\xi^+ - 2\dot{\xi}^\top Z_-\xi^- \\ &= \begin{bmatrix} \xi \\ \xi^+ \\ \xi^- \\ \tilde{\delta} \end{bmatrix}^\top \Upsilon \begin{bmatrix} \xi \\ \xi^+ \\ \xi^- \\ \tilde{\delta} \end{bmatrix} - \xi^\top Q\xi - (\xi^+)^\top Q_+\xi^+ \\ &\quad - (\xi^-)^\top Q_-\xi^- - 2(\xi^+)^\top \Psi\xi^- - 2(\xi^+)^\top \Psi_+\xi \\ &\quad - 2(-\xi^-)^\top \Psi_-\xi + \tilde{\delta}^\top\Gamma\tilde{\delta}. \end{aligned}$$

Note that

$$\begin{aligned} (\xi^+)^\top \Psi \xi^- &= 0, \\ (\xi^+)^\top \Psi_+ \xi &\geq 0, \\ (-\xi^-)^\top \Psi_- \xi &\geq 0 \end{aligned}$$

for any diagonal matrix Ψ and $\Psi_+ \geq 0$, $\Psi_- \geq 0$. Hence, if $\Upsilon \preceq 0$, as it is assumed in the theorem, we obtain that

$$\begin{aligned} \dot{V} \leq & -\xi^\top Q\xi - (\xi^+)^\top Q_+\xi^+ - (\xi^-)^\top Q_-\xi^- \\ & - 2(\xi^+)^\top \Psi_+\xi - 2(-\xi^-)^\top \Psi_-\xi + \tilde{\delta}^\top\Gamma\tilde{\delta} \\ \leq & -\xi^\top \Omega\xi + \tilde{\delta}^\top\Gamma\tilde{\delta}, \end{aligned}$$

where

$$\Omega = Q + \min\{Q_+, Q_-\} + 2\min\{\Psi_+, \Psi_-\} > 0,$$

is a diagonal matrix. The substantiated properties of V and its derivative imply that (7.25) is input-to-state stable (E. D. Sontag, 2001; Dashkovskiy, D.V. Efimov, and E. Sontag, 2011) with respect to the input $\tilde{\delta}$ (or, by its definition, with respect to $(\underline{\omega}, \overline{\omega})$). □

Following the proof of Theorem 7.22, for all $\xi \in \mathbb{R}^{2p}$,

$$\xi^\top (P + \min\{Z_+, Z_-\})\xi \le V(\xi) \le \xi^\top (P + Z_+^+ + Z_-^+)\xi,$$

then

$$\dot{V} \le -\alpha V + \tilde{\delta}^\top \Gamma \tilde{\delta}$$

for all $\xi, \tilde{\delta} \in \mathbb{R}^{2p}$, where

$$\alpha = \min_{i \in [2p]} \lambda_i \left(\Omega (P + Z_+^+ + Z_-^+)^{-1} \right),$$

and we can define the set (recall that the signal $\tilde{\delta}(t)$ is known for all $t \ge 0$)

$$\mathbb{X}_f = \{\xi \in \mathbb{R}^{2p} : V(\xi) \le \alpha^{-1} \sup_{t \ge 0} |\tilde{\delta}^\top(t) \Gamma \tilde{\delta}(t)|\}, \tag{7.26}$$

as the set that asymptotically attracts all trajectories in (7.25).

The conditions of Theorem 7.22 assume that the control gains K_0, K_1, K_2 are given, let us find these gains as solutions of linear matrix inequalities:

Corollary 7.23. *If there exist diagonal matrices P, $\tilde{Q}$, $\tilde{Q}_+$, $\tilde{Q}_-$, Z_+, Z_-, $\tilde{\Psi}_+$, $\tilde{\Psi}_-$, $\tilde{\Psi}$, $\Gamma \in \mathbb{R}^{2p \times 2p}$ and matrices $U_0, U_1, U_2 \in \mathbb{R}^{q \times 2p}$ such that the following linear matrix inequalities are satisfied:*

$$P > 0,\ Z_+ > 0,\ Z_- > 0,\ \Pi \preceq 0,\ \Gamma > 0,$$
$$\tilde{Q} + \min\{\tilde{Q}_+, \tilde{Q}_-\} + 2\min\{\tilde{\Psi}_+, \tilde{\Psi}_-\} > 0,$$

where

$$\Pi = \begin{bmatrix} \Pi_{11} & \Pi_{12} & \Pi_{13} & I \\ \Pi_{12}^\top & \Pi_{22} & \Pi_{23} & I \\ \Pi_{13}^\top & \Pi_{23}^\top & \Pi_{33} & -I \\ I & I & -I & -\Gamma \end{bmatrix},$$
$$\Pi_{11} = P^{-1}\mathcal{A}_0^\top + \mathcal{A}_0 P^{-1} + U_0^\top \mathcal{B}^\top + \mathcal{B}U_0 + \tilde{Q},$$
$$\Pi_{12} = \mathcal{A}_1 Z_+^{-1} + \mathcal{B}U_1 + P^{-1}\mathcal{A}_0^\top + U_0^\top \mathcal{B}^\top + \tilde{\Psi}_+,$$
$$\Pi_{13} = \mathcal{A}_2 Z_-^{-1} + \mathcal{B}U_2 - P^{-1}\mathcal{A}_0^\top - U_0^\top \mathcal{B}^\top - \tilde{\Psi}_-,$$

$$\Pi_{22} = Z_+^{-1}\mathcal{A}_1^\top + \mathcal{A}_1 Z_+^{-1} + U_1^\top \mathcal{B}^\top + \mathcal{B}U_1 + \tilde{Q}_+,$$
$$\Pi_{23} = \mathcal{A}_2 Z_-^{-1} + \mathcal{B}U_2 - Z_+^{-1}\mathcal{A}_1^\top - U_1^\top \mathcal{B}^\top + \tilde{\Psi},$$
$$\Pi_{33} = \tilde{Q}_- - Z_-^{-1}\mathcal{A}_2^\top - \mathcal{A}_2 Z_-^{-1} - U_2^\top \mathcal{B}^\top - \mathcal{B}U_2,$$

then (7.25) *for* $K_0 = U_0 P$, $K_1 = U_1 Z_+$ *and* $K_2 = U_2 Z_-$ *is input-to-state stable with respect to the inputs* $\underline{\omega}, \overline{\omega}$.

Proof. Note that the conditions $P > 0$, $Z_+ > 0$, $Z_- > 0$ imply $P + \min\{Z_+, Z_-\} > 0$, and

$$\Upsilon = \begin{bmatrix} P & 0 & 0 & 0 \\ 0 & Z_+ & 0 & 0 \\ 0 & 0 & Z_- & 0 \\ 0 & 0 & 0 & I_{2p} \end{bmatrix} \Pi \begin{bmatrix} P & 0 & 0 & 0 \\ 0 & Z_+ & 0 & 0 \\ 0 & 0 & Z_- & 0 \\ 0 & 0 & 0 & I_{2p} \end{bmatrix}$$

under substitution $U_0 = K_0 P^{-1}, U_1 = K_1 Z_+^{-1}, U_2 = K_2 Z_-^{-1}, \tilde{Q} = P^{-1} Q P^{-1}, \tilde{Q}_+ = Z_+^{-1} Q_+ Z_+^{-1}$, $\tilde{Q}_- = Z_-^{-1} Q_- Z_-^{-1}$, $\tilde{\Psi} = Z_-^{-1} \Psi Z_+^{-1}$, $\tilde{\Psi}_+ = P^{-1}\Psi_+ Z_+^{-1}$ and $\tilde{\Psi}_- = P^{-1}\Psi_- Z_-^{-1}$. Hence, $\Upsilon \preceq 0$ provided that $\Pi \preceq 0$. The inequalities $\tilde{Q} + \min\{\tilde{Q}_+, \tilde{Q}_-\} + 2\min\{\tilde{\Psi}_+, \tilde{\Psi}_-\} > 0$ and $Q + \min\{Q_+, Q_-\} + 2\min\{\Psi_+, \Psi_-\} > 0$ are equivalent due to the diagonal structure of all matrices. Therefore, under introduced restrictions all conditions of Theorem 7.22 are verified for $K_0 = U_0 P$, $K_1 = U_1 Z_+$ and $K_2 = U_2 Z_-$. □

The requirements imposed on P, Z_+, Z_- in this corollary are more restrictive than the conditions of Theorem 7.22, but it allows the gains K_0, K_1, K_2 to be efficiently calculated.

Under conditions of Theorem 7.22, the control (7.24) ensures stabilisation of the predictor (7.23) (*i.e.* (7.18) or (7.20)) in a vicinity $\mathbb{X}_f$ of the origin. The size of the vicinity is proportional to the system (7.3) uncertainty (it can be optimized by the choice of K_0, K_1, K_2), and due to (7.2), it implies that the system (7.3) also will reach a neighbourhood of the origin under the control (7.24). Hence, the posed control problem would be solved provided that (7.6) holds. In order to ensure the robust constraint satisfaction we consider an MPC design in the next section.

7.4.2 Robust constraint satisfaction

For brevity, the results of this section are given for the predictor (7.20) only (and the case of (7.18) can be considered by skipping Assumption 7.17 in the formulation). We need the following hypothesis in this section:

Assumption 7.24. *There exist $K_0, K_1, K_2 \in \mathbb{R}^{q\times 2p}$ satisfying the conditions of Theorem 7.22 for the matrices A_0 and ΔA_i with $i \in [M]$ calculated in (7.19) for $\mathcal{C}_{[N],\delta} = \Theta$, and*

$$\mathbb{X}_f \subset \mathbb{X}^2,$$

where the corresponding set $\mathbb{X}_f$ is given in (7.26), and

$$K_0\xi + K_1\xi^+ + K_2\xi^- + S\delta(t) \in \mathbb{U}$$

for any $\xi \in \mathbb{X}_f$ and $t \geq 0$.

These properties guarantee that there exists a control (7.24) that can be always applied to stabilise the predictor (7.20) (the worst-case estimate set Θ is used to calculate the system matrices) and into the set $\mathbb{X}_f$ the restrictions (7.6) also hold for such a control. Recall that we denote $\mathrm{d}t > 0$ the time step and $t_n = n\mathrm{d}t$ for $n \in \mathbb{N}_+$, and define $T > \mathrm{d}t$ as the MPC prediction horizon, *i.e.* at each $t_n \geq 0$, to design the input $u(t)$, an optimal control problem is solved on the interval $[t_n, t_n + T]$, and this optimal control problem is resolved again after $\mathrm{d}t$ units of time (on the interval $[t_n, t_{n+1} = t_n + \mathrm{d}t)$ the obtained optimal control is applied). Then the developed MPC algorithm can be formalized as follows for any $t_n \geq 0$:

1. Take the confidence region $\mathcal{C}_{[N],\delta}$ from (7.12) and calculate the matrices A_0 and ΔA_i with $i \in [2^d]$ for (7.19).

2. Find

$$\mathcal{U} = \underset{u:[t_n,t_n+T]\to\mathbb{R}^q}{\arg\min} \xi(t_n+T)^\top W_1\xi(t_n+T) + \int_{t_n}^{t_n+T} \xi^\top(s)W_2\xi(s) + u(s)^\top W_3 u(s)\mathrm{d}s, \quad (7.27)$$

   where $W_i \in \mathbb{R}^{2p\times 2p}$ are given positive definite symmetric matrices, such that the following constraints are satisfied:

   (a) $\xi : [t_n, t_n + T] \to \mathbb{R}^{2p}$ is a solution of (7.20) with $t = t_n$.

   (b) $\xi(s) \in \mathbb{X}^2$ and $u(s) \in \mathbb{U}$ for $s \in [t_n, t_n + T]$;

   (c) $\xi(t_n + T) \in \mathbb{X}_f$.

3. For $t \in [t_n, t_{n+1})$ select

$$u(t) = \begin{cases} \mathcal{U}(t) & \xi(t_n) \notin \mathbb{X}_f \\ (7.24) & \xi(t_n) \in \mathbb{X}_f \end{cases}, \quad (7.28)$$

   where K_0, K_1, K_2 are taken from Assumption 7.24.

As we can conclude, the idea of the proposed dual MPC scheme (see also Michalska and D. Q. Mayne, 1993; D. Mayne, Rawlings, et al., 2000; D. Mayne, Raković, et al., 2009) is to use an open-loop optimal control to reach a neighbourhood of the origin $\mathbb{X}_f$ ensuring a robust constraint satisfaction (7.6), where a closed-loop control (7.24) can be applied, which provides asymptotic performances (stability and robustness, also with constraint satisfaction due to Assumption 7.24 and the definition of the terminal set (7.26)).

The main result of the section is as follows:

Theorem 7.25. *Let $\underline{x}_0, \overline{x}_0 \in \mathbb{X}$, and Assumptions 7.4, 7.6, 7.9, 7.17 and 7.24 hold with $\overline{\omega}, \overline{\omega} - \underline{\omega}$ being non-increasing functions of $t \geq 0$. Then the closed-loop system given by (7.3), (7.5), (7.20) and (7.28) has the following properties:*

1. *Input-to-state stability for $\underline{x}$, $\overline{x}$ and practical input-to-state stability for x with respect to $\underline{\omega}, \overline{\omega}$ in the terminal set $\mathbb{X}_f$;*
2. *Recursive feasibility with reaching $\mathbb{X}_f$ in a finite time;*
3. *Constraint satisfaction (7.6).*

Proof. Recall that $\theta \in \mathcal{C}_{[N],\delta}$ for all $t_N \geq 0$ due to the result of Theorem 7.8, and we can enforce that the size of the set $\mathcal{C}_{[N],\delta}$ does not grow with time by iteratively taking the intersection: $\mathcal{C}_{[N],\delta} \triangleq$ "$\mathcal{C}_{[N],\delta}$ from (7.12)" $\cap\, \mathcal{C}_{[N-1],\delta}$.

Note that if for some $t_n \geq 0$ the initial conditions $(\underline{x}^\top(t_n), \overline{x}^\top(t_n))^\top \in \mathbb{X}_f \subset \mathbb{X}^2$, then the control (7.28) equals to (7.24). According to the definition (7.26) of $\mathbb{X}_f$ and Assumption 7.24, $\xi(t) \in \mathbb{X}_f$ and $u(t) \in \mathbb{U}$ for all $t \geq t_n$, and the system is input-to-state stable with respect to $\xi(t) = [\underline{x}^\top(t)\ \overline{x}^\top(t)]^\top$ due to the result of Theorem 7.22. Since $|x(t)| \leq |\xi(t)|$ under (7.2) for $t \geq t_n$ and $|\xi(t_n)| \leq |x(t_n)| + \zeta$ with $\zeta > 0$ (that is well defined for all initial conditions in $\mathbb{X}_f$), the practical input-to-state stability for the variable $x(t)$ follows. The point *1.* is proven.

Now, let $(\underline{x}^\top(0), \overline{x}^\top(0))^\top \in \mathbb{X}^2 \setminus \mathbb{X}_f$ and assume that there is a solution of the optimal control problem (7.27). Applying such a control through (7.28) for $t \in [0, \mathrm{d}t)$, we have that $\xi(t) \in \mathbb{X}$ and $u(t) \in \mathbb{U}$ on this time interval. At $t = t_1 = \mathrm{d}t$, if again $(\underline{x}^\top(t_1), \overline{x}^\top(t_1))^\top \in \mathbb{X}^2 \setminus \mathbb{X}_f$, then it recursively exists a solution to (7.27) since the set $\mathcal{C}_{[N],\delta}$ is shrinking by its design and the signals $\overline{\omega}(t), \overline{\omega}(t) - \underline{\omega}(t)$ are non-increasing by hypotheses of the theorem (*i.e.*, the solution obtained at t_n is a sub-optimal branch of the solution calculated at t_{n-1} for all $n \geq 1$). Thus, recursive feasibility follows. Note that $\mathbb{X}_f$ is a neighbourhood of the origin, and the given in (7.27) cost with positive definite matrices W_1, W_2 and W_3 is minimized inside $\mathbb{X}_f$. Using this and sub-optimality arguments, since $\xi(t_n + T) \in \mathbb{X}_f$ in (7.27) (provided that the optimal control $\mathcal{U}$ is applied) and $[\underline{x}(t_n), \overline{x}(t_n)] \subset [\underline{x}(t_{n-1} + \mathrm{d}t), \overline{x}(t_{n-1} + \mathrm{d}t)]$ for all $n \geq 1$, there is a

finite time instant $t_k \geq T$ such that $(\underline{x}^\top(t_k), \overline{x}^\top(t_k))^\top \in \mathbb{X}_f$, and the system further stays there. The point *2.* is substantiated.

The point *3.* is a consequence of the previous analysis: under the control (7.28) the constrains (7.6) are always satisfied. □

Remark 7.26. *At each t_N, $N \in \mathbb{N}_+$, the gains K_0, K_1, K_2 can be recalculated for the currently estimated set $\mathcal{C}_{[N],\delta}$. If next, the obtained in (7.26) set $\mathbb{X}_f$ satisfies $\mathbb{X}_f \subset \mathbb{X}^2$, and $K_0\xi + K_1\xi^+ + K_2\xi^- + S\delta(t) \in \mathbb{U}$ for any $\xi \in \mathbb{X}_f$ and $t \geq 0$, as in Assumption 7.24, then the new set $\mathbb{X}_f$ and these updated gains K_0, K_1, K_2 can be used in (7.28).*

We illustrate the efficiency of the proposed MPC approach by numerical experiments.

7.4.3 Numerical experiment

We tackle the problem of the robust adaptive lateral control of an autonomous vehicle for a lane-keeping application, implemented in HIGHWAY-ENV. In contrast with Chapter 3 where we adopted a kinematic standpoint and simply assumed that the vehicle acceleration could be controlled directly, we dive deeper into a dynamical description subjected to *unknown* tire friction forces. We still represent the state of a rigid vehicle by its position (p_x, p_y), its yaw angle ψ, its velocity (v_x, v_y) in the body frame and yaw rate r, and additionally denote its mass as m, its moment of inertia as I_z, and its front and rear axle positions as a, b. We consider the Dynamical Bicycle Model described in (Awan, 2014, Chapter 3.2): the vehicle is moving at constant longitudinal speed u, and the lateral force F_y of a tire with slip angle α is assumed to be linear with an unknown friction coefficient C_α: $F_y = C_\alpha \alpha$. The slip angle of the front and rear tires are respectively denoted as α_f, α_r, along with the corresponding friction coefficients. Under a small angle approximation for ψ, α_f, α_r, Newton's second law of motion yields the linear dynamics (7.3) with

$$x = \begin{bmatrix} p_y \\ \psi \\ v_y \\ r \end{bmatrix}, \quad \theta = \begin{bmatrix} C_{\alpha_f} \\ C_{\alpha_r} \end{bmatrix}, \quad A = \begin{bmatrix} 0 & v_x & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -v_x \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{2}{m} \\ \frac{a}{I_z} \end{bmatrix},$$

$$\phi_1 = \frac{-2}{m v_x I_z} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & I_z & aI_z \\ 0 & 0 & am & a^2 m \end{bmatrix}, \quad \phi_2 = \frac{-2}{m v_x I_z} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & I_z & -bI_z \\ 0 & 0 & -bm & b^2 m \end{bmatrix}.$$

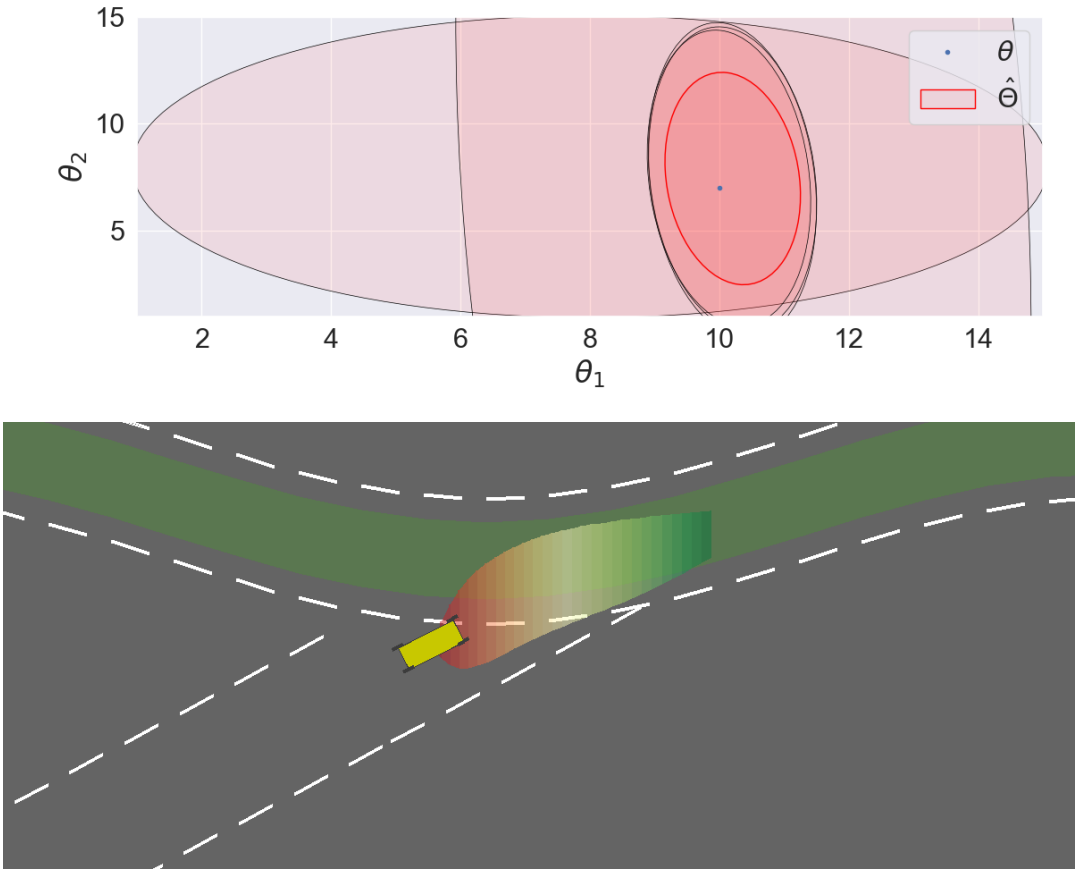


Figure 7.8 – Top: the model estimation showing the confidence region $\mathcal{C}_{[N],\delta}$ from (7.23) at different times t_N. **Bottom**: a lane keeping application, where a car must follow a lane-center curve under unknown friction and perturbations. X_f is shown in green, and $\xi(t)$ as an area with a color gradient.

Instead of simply stabilising the vehicle state x, we track the lateral position $y_r(t)$ of the lane centre. However, we do not have access to a full state reference $x_r(t) = [y_r(t), \psi_r(t), v_{y,r}(t), r_r(t)]^\intercal$ consistent with the dynamics (7.3). Thus, we define the state $\tilde{x} = x - [y_r(t), 0, 0, 0]^\intercal$ and consider the remaining unknown terms $[0, \psi_r(t), v_{y,r}(t), r_r(t)]$ and $u_r(t)$ as perturbations $\omega(t)$, bounded since x_r, u_r are assumed to belong to $\mathbb{X} = \pm[3, 2, 6, 6]^\intercal$ and $\mathbb{U} = \pm[10]$.

The Figure 7.8 depicts our approach. The confidence region $\mathcal{C}_{[N],\delta}$ from (7.12) is shown in the top graph, and shrinks with time. To simplify verification of Assumption 7.17 for this example, an auxiliary preliminary feedback has been applied shifting the eigenvalues of the closed-loop system. The robust stability of this feedback is assessed with the LMI of Theorem 7.22, and we compute the corresponding basin of attraction X_f from (7.26), represented in green in the bottom subfigure. Then, we use a sampling-based MPC scheme (Homem-de-Mello and Bayraksan, 2014) to solve (7.27) and bring $\xi(t)$ into X_f in $T = 3\,\mathrm{s}$. The associated interval prediction $\xi(t)$ from (7.23) is represented with a colour gradient from $t = t_n$ (red) to $t = t_n + T$

(green). Once the vehicles enters X_f, we finally switch to the closed-loop feedback (7.24) following (7.28) for the rest of the simulation[4].

7.5 Minimax control with generic costs

As we discussed in Section 7.1, the ability to stabilise a system in the neighbourhood of the origin is not sufficient to tackle many tasks for which there exist no clear notion of equilibrium, and where the system must continually and dynamically adapt to its surroundings. This especially includes tasks akin to obstacle avoidance, which constitutes a substantial part of motion planning.

Therefore, in this section and as a second application of the tools introduced in Sections 7.2 and 7.3, we tackle the arguably more expressive objective of minimax control (7.7) of an arbitrary bounded reward function $R : \mathbb{R}^p \to [0, 1]$, recalled here:

$$\sup_{bu \in (\mathbb{R}^q)^{\mathbb{N}}} \inf_{\substack{\theta \in \mathcal{C}_{[N],\delta} \\ \omega \in [\underline{\omega}, \overline{\omega}]^{\mathbb{R}}}} \left[\sum_{n=N+1}^{\infty} \gamma^n R(x_n(\mathbf{u}, \omega)) \right]. \tag{7.7}$$

Evaluation In order to evaluate this robust objective V^r, we approximate it thanks to the interval prediction $[\underline{x}(t), \overline{x}(t)]$ of Section 7.3.

Definition 7.27 (Surrogate objective)**.** *Let*

$$\hat{V}^r(\mathbf{u}) \triangleq \sum_{n=N+1}^{\infty} \gamma^n \underline{R}_n(\mathbf{u}), \tag{7.29}$$

$$\text{where} \quad \underline{R}_n(\mathbf{u}) \triangleq \min_{x \in [\underline{x}_n(\mathbf{u}), \overline{x}_n(\mathbf{u})]} R(x), \tag{7.30}$$

and $\underline{x}_n(\mathbf{u}), \overline{x}_n(\mathbf{u})$ follow the dynamics defined in (7.20).

This amounts to changing the reward function, except that the worst case is assessed over the whole past trajectory, which makes this pessimistic reward $\underline{R}_n$ *not Markovian*.

Theorem 7.28 (Lower bound)**.** *The surrogate objective (7.29) is a lower bound of the true objective (7.7):*

$$\hat{V}^r(\mathbf{u}) \leq V^r(\mathbf{u})$$

[4] A video is available at https://youtu.be/axurBzHRLGY

Algorithm 7.1: Integrated framework for confident estimation, interval prediction and minimax control

Data: confidence level δ, structure (A, ϕ), reward R, $\mathcal{D}_{[0]} \leftarrow \emptyset$, $\mathbf{a}_1 \leftarrow \emptyset$
for $N = 0, 1, 2, \dots$ **do**
  $\mathcal{C}_{[N],\delta} \leftarrow$ Model Estimation$(\mathcal{D}_{[N]})$. (7.12)
  for *each planning step* $k \in \{N, \dots, N+K\} = N + [K]$ **do**
    $[\underline{x}_{k+1}, \overline{x}_{k+1}] \leftarrow$ Interval Prediction$(\mathcal{C}_{[N],\delta}, \mathbf{a}_k b)$ for each action $b \in \mathcal{A}$. (7.20)
    $\mathbf{a}_{k+1} \leftarrow$ Pessimistic Planning$(\underline{R_{k+1}}([\underline{x}_{k+1}, \overline{x}_{k+1}]))$. (7.31)
  Execute the recommended control u_{N+1}, and add the transition $(x_{N+1}, y_{N+1}, u_{N+1})$ to $\mathcal{D}_{[N+1]}$.

Proof. We provide a proof in Section D.1.3. □

A direct consequence of Theorem 7.28 is that since all our approximations are conservative, if we manage to find a control sequence such that no *"bad event"* (*e.g.* a collision) happens according to the surrogate objective $\hat{V}^r$, then we are guaranteed that they will not happen either when the controls are executed on the true system.

Planning To optimise $\hat{V}^r$ (7.29), we cannot use Dynamic Programming algorithms since the state space is continuous and the pessimistic rewards are non-Markovian. Instead, as we did in Chapter 6, we turn to tree-based planning algorithms, which optimise a sequence of actions based on the corresponding sequence of rewards, without requiring Markovity nor state enumeration.

Though there exist works addressing continuous action spaces (Buşoniu, Páll, and Rémi Munos, 2018; Weinstein and Littman, 2012), we resort to a first approximation and discretise the continuous decision space $\mathbb{R}^q$ by adopting a hierarchical control architecture: at each time, the agent can select a high-level *action* a from a finite space $\mathcal{A}$. Each action $a \in \mathcal{A}$ corresponds to the selection of a low-level controller π_a, that we take affine: $u(t) = \pi_a(x(t)) \triangleq -K_a x(t) + u_a$. For instance, a tracking a subgoal x_g can be achieved with $\pi_g = K(x_g - x)$. This discretisation induces a suboptimality, but it can be mitigated by diversifying the controller basis. The robust objective (7.7) becomes

$$\sup_{\mathbf{a} \in \mathcal{A}^{\mathbb{N}}} V^r(\mathbf{a}),$$

where $x_n(\mathbf{a}, \omega)$ stems from (7.3) with $u_n = \pi_{a_n}(x_n)$.

This enables us to consider the OPD algorithm (Hren and Rémi Munos, 2008) tailored for the case when the relationship between actions and rewards is deterministic. Indeed, the stochasticity of perturbations and measurements is encased in $\hat{V}^r$: given the observations up to time N, both the predictor dynamics (7.20) and the pessimistic rewards (7.30) are deterministic.

At each planning iteration $k \in [K]$, OPD progressively builds a tree $\mathcal{T}_{k+1}$ by forming upper-bounds $U_a(k)$ over the value of sequences of actions a, and expanding[5] the leaf a_k with highest upper-bound:

$$a_k = \arg\max_{a \in \mathcal{L}_k} U_a(k), \quad U_a(k) = \sum_{n=0}^{h(a)-1} \underline{R}_n(a) + \frac{\gamma^{h(a)}}{1-\gamma} \tag{7.31}$$

where $\mathcal{L}_k$ is the set of leaves of $\mathcal{T}_k$, $h(a)$ is the length of the sequence a, and $\underline{R}_n(a)$ the pessimistic reward (7.30) obtained at time n by following the controls $u_n = \pi_{a_n}(x_n)$.

Algorithm 7.1 shows the full integration of the three procedures of estimation, prediction and control.

Lemma 7.29 (Planning performance of Hren and Rémi Munos, 2008)**.** *The simple regret of the OPD algorithm (7.31) applied to the surrogate objective (7.29) after K planning iterations is*

$$\text{if } \kappa > 1, \quad \hat{V}^r(a^\star) - \hat{V}^r(a_K) = \mathcal{O}\left(K^{-\frac{\log 1/\gamma}{\log \kappa}}\right);$$
$$\text{if } \kappa = 1, \quad \hat{V}^r(a^\star) - \hat{V}^r(a_K) = \mathcal{O}\left(\gamma^{(1-\gamma)^{\log_\gamma(\kappa/|\mathcal{A}|)}K/c}\right)$$

where κ is a problem-dependent measure of the proportion of near-optimal paths:

$$\kappa = \limsup_{h\to\infty} \left| \left\{ a \in A^h : \hat{V}^r(a) \geq \hat{V}^r(a^\star) - \frac{\gamma^{h+1}}{1-\gamma} \right\} \right|^{1/h}.$$

Proof. We provide a proof in Section C.1.8. □

Hence, by using enough computational budget K for planning we can get as close as we want to the optimal surrogate value $\hat{V}^r(a^\star)$, at a polynomial rate. Unfortunately, there exists a gap between $\hat{V}^r$ and the true robust objective V^r, which stems from three approximations: (i) the true reachable set was approximated by an enclosing interval in (7.2); (ii) the time-invariance of the dynamics uncertainty $A(\theta) \in \mathcal{C}_{[N],\delta}$ was handled by the interval predictor (7.20) as if it were a time-varying uncertainty $A(\theta(t)) \in \mathcal{C}_{[N],\delta}, \forall t$; and (iii) the lower-bound $\sum \min \leq \min \sum$ used to define the surrogate objective (7.29) is not tight. However, this gap can be bounded with additional assumptions.

Theorem 7.30 (Regret bound)**.** *Under two conditions:*

1. *a Lipschitz regularity assumption for the reward function R;*

[5] The expansion of a leaf node a refers to the simulation of its children transitions $a\mathcal{A} = \{ab, b \in \mathcal{A}\}$

2. *a stability condition: there exist* $P > 0, Q_0 \in \mathbb{R}^{p\times p}$, $\rho > 0$, *and* $N_0 \in \mathbb{N}$ *such that*

$$\forall N > N_0, \quad \begin{bmatrix} A_N^\intercal P + P A_N + Q_0 & P|D| \\ |D|^\intercal P & -\rho I_r \end{bmatrix} < 0;$$

we can bound the suboptimality of Algorithm 7.1 with planning budget K *as:*

$$V(a^\star) - \hat{V}^r(a_K) \leq \underbrace{\Delta_\omega}_{\substack{\text{robustness to} \\ \text{perturbations}}} + \underbrace{\mathcal{O}\left(\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})}\right)}_{\text{estimation error}} + \underbrace{\mathcal{O}\left(K^{-\frac{\log 1/\gamma}{\log \kappa}}\right)}_{\text{planning error}}$$

with probability at least $1-\delta$, *where* $V(a)$ *is the optimal expected return when executing an action* $a \in \mathcal{A}$, $a_\star$ *is an optimal action, and* Δ_ω *is a constant which corresponds to an irreducible suboptimality suffered from being robust to instantaneous disturbances* $\omega(t)$.

Proof. We provide a proof in Section D.1.4. □

It is difficult to check the validity of the stability condition 2. since it applies to matrices A_N produced by the algorithm rather than to the system parameters. A stronger but easier to check condition is that the polytope (7.19) at some iteration becomes included in a set where this property is uniformly satisfied. For instance, if the features are sufficiently excited, the estimation converges to a neighbourhood of the true dynamics $A(\theta)$. This also allows to further bound the input-dependent estimation error term.

Corollary 7.31 (Asymptotic near-optimality)**.** *Under an additional persistent excitation* (*PE*) *assumption*

$$\exists \underline{\phi}, \overline{\phi} > 0 : \forall n \geq n_0, \quad \underline{\phi}^2 \leq \lambda_{\min}(\Phi_n^\intercal \Sigma_p^{-1} \Phi_n) \leq \overline{\phi}^2, \tag{7.32}$$

the stability condition 2. of Theorem 7.30 can be relaxed to apply to the true system: there exist P, Q_0, ρ *such that*

$$\begin{bmatrix} A(\theta)^\intercal P + P A(\theta) + Q_0 & P|D| \\ |D|^\intercal P & -\rho I_r \end{bmatrix} < 0;$$

and the regret bound in Theorem 7.30 takes the more explicit form:

$$V(a_\star) - \hat{V}^r(a_K) \leq \underbrace{\Delta_\omega}_{\substack{\text{robustness to} \\ \text{perturbations}}} + \underbrace{\mathcal{O}\left(\frac{\log\left(N^{d/2}/\delta\right)}{N}\right)}_{\text{estimation error}} + \underbrace{\mathcal{O}\left(K^{-\frac{\log 1/\gamma}{\log \kappa}}\right)}_{\text{planning error}}$$

which ensures asymptotic near-optimality when $N \to \infty$ and $K \to \infty$.

Proof. We provide a proof in Section D.1.5. □

7.6 Multi-model selection

The procedure we developed in Sections 7.2, 7.3 and 7.5 relies on strong modelling assumptions, such as the dynamics structure in Assumption 7.4. But what if they are wrong?

Model adequacy One of the major benefits of using the family of linear models, compared to richer model classes, is that they provide strict conditions allowing to quantify the adequacy of the modelling assumptions to the observations.

Given $N-1$ observations, Section 7.2 provides a polytopic confidence region (7.19) that contains $A(\theta)$ with probability at least $1-\delta$. Since the dynamics are linear, we can propagate this confidence region to the next observation: y_N must belong to the Minkowski sum of a polytope representing model uncertainty $\mathcal{P}(A_0 x_N + B u_N, \Delta A_1 x_N, \dots, \Delta A_{2^d} x_N)$ and a polytope $\mathcal{P}(0_p, \underline{\eta}, \overline{\eta})$ bounding the perturbation and measurement noises. Delos and Teissandier (2015) provide a way to test this membership in polynomial time using linear programming. Whenever it is not verified, we can confidently reject the (A, ϕ)-modelling Assumption 7.4. This enables us to consider a rich set of potential features $\left((A^1, \phi^1), \dots, (A^M, \phi^M)\right)$ rather than relying on a single assumption, and only retain those that are consistent with the observations so far. Then, every remaining hypothesis must be considered during planning.

Robust selection We temporarily ignore the parametric uncertainty on θ to simply consider several candidate dynamics models, which all correspond to different modelling assumptions. We also restrict to deterministic dynamics, which is the case of (7.20).

Assumption 7.32 (Multi-model ambiguity)**.** *The true dynamics f lies within a finite set of candidate models $f^1, \dots, f^M$.*

$$\exists m \in [M] : \dot{x}(t) = f^m(x(t), u(t)), \forall t \geq 0$$

Preparing for the Worst

We want to adapt our planning algorithm in order to balance these concurrent hypotheses in a robust fashion, *i.e.* maximise a robust objective with discrete ambiguity

$$\underbrace{\sup_{a\in\mathcal{A}^{\mathbb{N}}} \min_{m\in[M]} \sum_{n=N+1}^{\infty} \gamma^n R_n^m}_{V^r(a)} \tag{7.33}$$

where R_n^m is the reward obtained by following the action sequence a up to step n under the dynamics f^m. This objective could be optimised in the same way as in Section 7.5, but this would result in a coarse and lossy approximation. Instead, we exploit the finite uncertainty structure of Assumption 7.32 to asymptotically recover the true V^r by modifying the OPD algorithm in the following way:

Definition 7.33 (Robust UCB)**.** *We replace the upper-bound (7.31) on sequence values in OPD by*

$$U_a^r(k) \triangleq \min_{m\in[M]} \sum_{n=0}^{h-1} \gamma^n R_n^m + \frac{\gamma^h}{1-\gamma} \tag{7.34}$$

An illustration of the computation of the robust upper-bounds is presented in Figure 7.10. Note that it is not equivalent to solving each control problem independently and following the action with highest worst-case value:

Remark 7.34. *In the definition of $U_a^r(k)$ (D.2) and $L_a^r(k)$ (D.4) it is essential that the minimum over the models is only taken at the end of trajectories, in the same way as for the robust objective (7.33) in which the worst-case dynamics is only determined after the action sequence has been fully specified. Assume that $L_a^r(k)$ is instead naively defined as*

$$L_a^r(k) = \min_{m\in[1,M]} L_a^m(k),$$

This would not recover the robust policy, as we show in Figure 7.9 with a simple counter-example.

We analyse the sample complexity of the corresponding robust planning algorithm.

Proposition 7.35 (Robust planning performance)**.** *The robust version of OPD (7.34) enjoys the same regret bound as OPD in Lemma 7.29, with respect to the multi-model objective (7.33).*

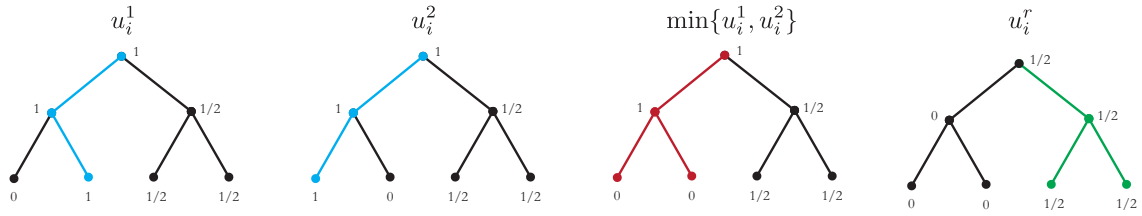


Figure 7.9 – From left to right: two simple models and corresponding u-values with optimal sequences in blue; the naive version of the robust values returns sub-optimal paths in red; our robust U-value properly recovers the robust policy in green.

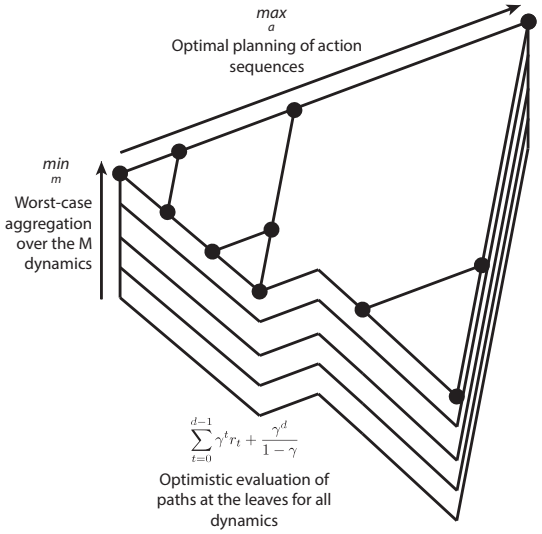


Figure 7.10 – The computation of robust U-values in (7.34). The simulation of trajectories for every dynamics model f^m is represented as stacked versions of the expanded tree $\mathcal{T}_k$.

Proof. We provide a proof in Section D.1.6. □

The regret depends on the number K of node expansions, but each expansion now requires M times more simulations than in the single-model setting. The solution of the robust objective (7.33) with discrete ambiguity $f \in \{f^m\}_{m\in[M]}$ can be recovered exactly, asymptotically when the planning budget K goes to infinity. This contrasts the results obtained in Section 7.5 for the robust objective (7.7) with continuous ambiguity $\theta \in \mathcal{C}_{[N],\delta}$, for which `OPD` only recovers the surrogate approximation $\hat{V}^r$, as discussed in Theorem 7.30. Finally, the two approaches of Sections 7.5 and 7.6 can be merged by using the pessimistic reward (7.30) in (7.34).

7.7 Experiments

Obstacle avoidance with unknown friction We first consider a simple illustrative example, shown in Figure 7.3: the control of a 2D system with position (p_x, p_y) and velocity (v_x, v_y) moving by means of a force (u_x, u_y) in an environment with unknown anisotropic friction.

$$\begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{v_x} \\ \dot{v_y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\theta_x & 0 \\ 0 & 0 & 0 & -\theta_y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u_x \\ u_y \end{bmatrix}$$

Note that the Assumption 7.17 for A_N is always verified. The reward is non-smooth and encodes the task of navigating to reach a goal state x_g while avoiding collisions with obstacles: $R(x) = \delta(x)/(1 + \|x - x_g\|_2)$ where $\delta(x)$ is 0 whenever x collides with an obstacle, 1 otherwise. The actions $\mathcal{A}$ are constant controls in the up, down, left and right directions. The environment is illustrated in Figure 7.11.

For the reasons mentioned above, no robust baseline applies to our setting. We compare Algorithm 7.1 to the non-robust adaptive control approach that plans with the estimated dynamics $\theta_{N,\lambda}$, and thus enjoys the same prior knowledge of dynamics structure and reward. This highlights the benefits of the robust formulation solely rather than stemming from algorithm design. We show in Table 7.1 the results of 100 simulations of a single episode: the robust agent performs worse than the nominal agent on average, but manages to ensure safety while the nominal agent collides with obstacles in 4 % of simulations[6]. We also compare to a standard model-free approach, DQN, which does not benefit from the prior knowledge of the system dynamics, and is instead trained over multiple episodes. The reported performance is that of the final policy obtained after training for 3000 episodes, during which 897 ± 64 collisions occurred $(29.9 \pm 2.1\%)$. We study the evolution of the suboptimality $V(x_N) - \sum_{n>N} \gamma^{n-N} R(x_n)$

[6] A video is available at https://youtu.be/jr2yi6Lf0bM

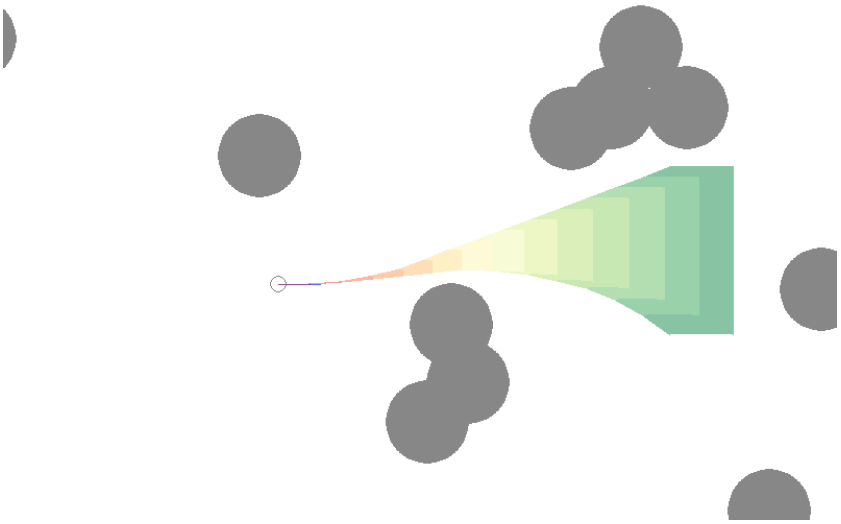


Figure 7.11 – Algorithm 7.1 running on the obstacle avoidance environment: we show the predicted state interval at each prediction time step (from red to green).

Table 7.1 – Performances on the obstacle task. We give the frequency of collision, minimum and average return achieved on a single episode, repeated with 100 random seeds. The robust agent performs worse than the nominal agent on average, but manages to ensure safety and attains a better worst-case performance.

Performance	failures	min	avg $\pm$ std
Oracle	0%	11.6	14.2 ± 1.3
Nominal	4%	2.8	$\mathbf{13.8} \pm 2.0$
Algorithm 7.1	**0%**	**10.4**	13.0 ± 1.5
DQN (trained)	6%	1.7	12.3 ± 2.5

with respect to the number of samples N, by comparing the empirical returns from a state x_N to the value $V(x_N)$ that the agent would get by acting optimally from x_N with knowledge of the dynamics. Although the assumptions of Theorem 7.30 are not satisfied (e.g. non-smooth reward), the mean suboptimality of the robust agent, shown in Figure 7.12, still decreases polynomially with N: Algorithm 7.1 gets *more efficient* as it is *more confident* while *ensuring safety* at all times. In comparison, the nominal agent enjoys a smaller suboptimality on average, but higher in the worst-case.

Behavioural planning for an autonomous vehicle We consider the HIGHWAY-ENV environment for simulated driving decision problems. An autonomous vehicle with state $x_0 \in \mathbb{R}^4$ is approaching an intersection among N_v other vehicles with states $x_i \in \mathbb{R}^4$, resulting in a joint traffic state $x = [x_0, \dots, x_{N_v}]^\top \in \mathbb{R}^{4N_v+4}$. These vehicles follow parametrized behaviours $\dot{\chi}_i = f_i(x, \theta_i)$ with unknown parameters $\theta_i \in \mathbb{R}^5$. We appreciate a first advantage of the

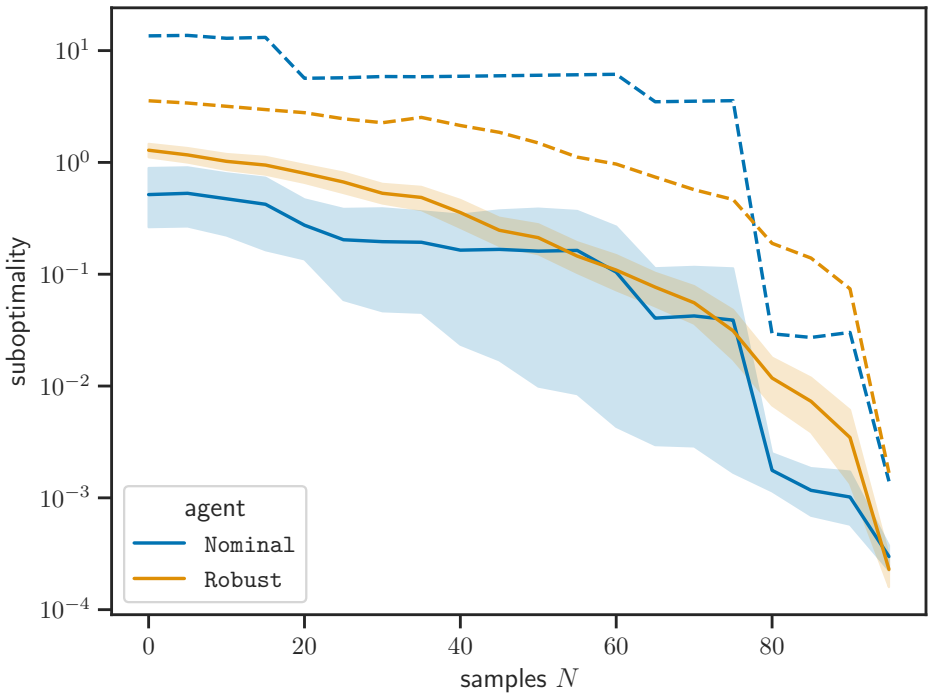


Figure 7.12 – The mean (solid), 95% confidence interval for the mean (shaded) and maximum (dashed) simple regret with respect to N.

structure imposed in Assumption 7.4: the uncertainty space of θ is $\mathbb{R}^{5N_v}$. In comparison, the traditional LQ setting where the whole state matrix A is estimated would have resulted in a much larger parameter space $\theta \in \mathbb{R}^{16{N_v}^2}$. The system dynamics f, which describes the interactions between vehicles, can only be expressed in the form of Assumption 7.4 given the knowledge of the desired route for each vehicle, with features ϕ expressing deviations to the centerline of the followed lane. Since these intentions are unknown to the agent, we adopt the multi-model perspective of Section 7.6 and consider one model per possible route for every observed vehicle before an intersection. In Table 7.2, we compare Algorithm 7.1 to a nominal agent planning with two different modelling assumptions: Nominal 1 has access to the true followed route for each vehicle, while Nominal 2 does not and picks the model with minimal prediction error. We show the multi-model rejection and robust selection procedure through the display of several trajectory hulls for all possible destinations of the observed vehicles[7]. Again we also compare to a DQN baseline trained over 3000 episodes, causing 1058 ± 113 collisions while training ($35 \pm 4\%$). As before, the robust agent has a higher worst-case performance and avoids collisions at all times, at the price of a decreased average performance.

[7] A video is available at https://youtu.be/DhoJAmJDau4

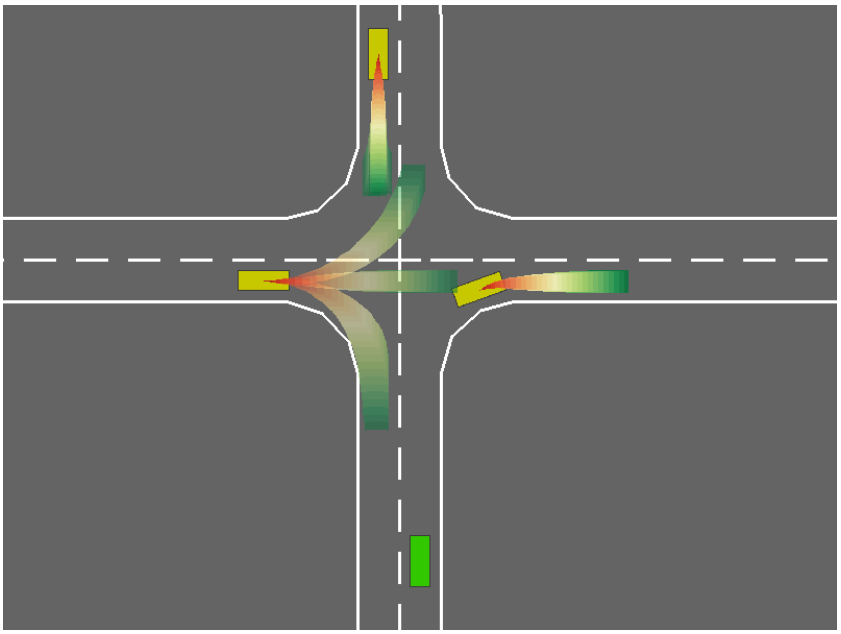


Figure 7.13 – The intersection crossing task. We show the trajectory intervals corresponding to behavioural uncertainty for each observed vehicle, and the multi-model assumption over the followed route.

Table 7.2 – Performances on the driving task. We make the same observations as in Table 7.1.

Performance	failures	min	avg $\pm$ std
Oracle	0%	6.9	7.4 ± 0.5
Nominal 1	4%	5.2	$\mathbf{7.3} \pm 1.5$
Nominal 2	33%	3.5	6.4 ± 0.3
Algorithm 7.1	**0%**	**6.8**	7.1 ± 0.3
DQN (trained)	3%	5.4	6.3 ± 0.6

Chapter conclusion

We propose a framework for the robust estimation, prediction and control of a partially known linear system. After deriving a confidence region for the state matrix through non-asymptotic linear regression, we design an interval predictor guaranteed to contain the induced trajectory, and whose stability is guaranteed upon satisfaction of an LMI. We leverage these tools in two applications. First, the robust stabilisation of the system at the origin under state and control constraints, that we achieve with a dual MPC and feedback that both exploit the predicted intervals. Second, the minimax control of a generic (non-quadratic) cost function, for which we provide a tree-based planning algorithm whose predicted performance is guaranteed and whose regret is bounded. The applicability of the method is further improved by a multi-model extension and demonstrated on several simulated driving applications, namely: socially-aware

trajectory prediction for an observed vehicle, steering control under unknown tire friction, and safe intersection crossing among drivers with uncertain destination and driving styles.

Part Conclusion

Review of our Requirements

Again, in Table 7.3 we discuss whether the methods developed in Part III address the challenges identified in Chapter 1.

Criterion		Description
Social Awareness	✓	In Chapters 6 and 7, the planning algorithm relies on a predictive model $\dot{x}_i = f_i(x, u)$ in which the motion of a vehicle $i \in [N_v]$ is coupled to that of other vehicles $x = [x_0, x_1, \dots x_{N_v}]$ in the scene, to account for driving interactions.
Sample Efficiency	✓	For *model estimation*, we rely in Chapter 7 on a structured parametrised model $\dot{x} = A(\theta)x + Bu$ which incorporates priors over car-like kinematics (non-holonomic) and human driving behaviours (lane following, cruise control). The *planning* step can benefit from the contributions of Chapter 6 on optimistic tree-based planning, such as merging the states of overlapping sampled trajectories.
Safety	✓	Two other notions of risk were introduced in Chapter 7: first as a hard constraint $x \in \mathbb{X}$, $u \in \mathbb{U}$ under bounded disturbances and parametric model uncertainty; and second, as a minimax objective that produces risk-averse behaviours by evaluating the worst possible outcome under said disturbances and model uncertainty.
Balance between safety and efficiency	✓	The conservativeness of the robust MPC algorithms of Chapter 7 can be tuned by adjusting the confidence level δ to trade-off the error probability with the size of the confidence region $\mathcal{C}_{[N],\delta}$; or by considering additional modelling assumptions (A, ϕ) in the multi-model perspective, which will increase robustness at the expense of efficiency.

Table 7.3 – Do the methods of Part III comply with the specifications of Chapter 1?

Chapter 8

General Conclusion and Perspectives

Nos équipiers
Sur les voies
Ralentissez

Vinci Autoroutes[1]

8.1 Conclusion on our contributions

In this thesis, we proposed a learning-based approach to the problem of behavioural planning for autonomous vehicles, with a focus on scenes with several drivers interacting. Following an in-breadth (Chapter 2), as well as in-depth (Chapter 3), initial investigation, we identified a set of key issues which make this problem challenging. We now recall these subjects, and precise how we strived to address them both in the model-free approach of Part II and the model-based approach of Part III.

Coupled social dynamics In dense traffic, the dynamics of distinct vehicles are locally coupled, due to how drivers react and adapt to their surroundings. Consequently, predicting the course or acting in a driving scene requires a *social awareness* skill: the ability to accurately understand and exploit these couplings. In Chapter 4, this function was performed by a *social attention* mechanism in the policy architecture, which enables the agent to filter out irrelevant objects from a complex driving scene and retain only those that represent a risk of collision. In Chapter 5, this coupling was made explicit by describing the motion of a vehicle i through a dynamical model $\dot{x_i} = f_i(x)$ taking the whole traffic state x as input.

[1] Writings collected by @pooredward.

Uncertainty due to human drivers Another critical difficulty lies in the uncertainty of human behaviours. In RL, the traditional approach to account for uncertainty is to incorporate stochasticity in the system dynamics, as we did in Chapter 5 where the objectives are formulated in terms of *expected* rewards and costs. Incidentally, we also observed in Chapter 4 that our attention-based architecture is highly sensitive to ambiguous and disambiguated information, such as vehicles' destinations. In Chapter 7 however, we adopted another view and assumed that the dynamics were (close to) deterministic, but dependent on some *unknown* parameters –both continuous and discrete– that could be estimated along the way.

Safety To deal with this uncertainty, we studied three models of safety. In Chapter 5, following the CMDP framework, we formalised risk as the expected discounted sum of an additional *cost signal* $C(s, a)$ –separate from the rewards $R(s, a)$– *constrained* to remain below a threshold β. In Chapter 7, we introduced a novel *interval predictor* allowing us to bound the set of reachable trajectories given the current parametric uncertainty over the dynamics, which required circumventing the instability of previous methods. This enabled us to cast safety as a *robust stabilisation* and *constraint satisfaction* problem, ensuring that the system stays at all time within a safe space $\mathbb{X} \times \mathbb{U}$. Finally, to go beyond stabilisation problems, we considered a third formalisation of safety as a *worst-case* outcome, and proposed an algorithm for the minimax control of a *generic* reward function $R(s, a)$. In addition to being tractable, each component of this algorithm is theoretically grounded, allowing us to obtain an end-to-end guarantee that the best performance predicted during planning is achievable on the underlying system, as well as the first regret bound in this setting, extending the state-of-the-art so far limited to quadratic costs.

Trade-off between safety and efficiency As we just saw, safety is always defined with respect to some *admissible* uncertainty. The larger the set of scenarios one is willing to consider and protect against, the more conservative they need to be to ensure safety. In particular, situations that require interacting with other agents are always susceptible to lead to accidents, when considering (unlikely) adversarial scenarios. In that sense, *absolute safety is not achievable*, or only at the cost of usability. To strike the right level of safety, we need to consider the right level of uncertainty, the right set of possible outcomes. In Chapter 7, the size of this ambiguity set is controlled by adjusting the confidence level δ for continuous parameters (*e.g.* driving style), and by adding or removing (A, ϕ)-modelling assumptions from the multi-model extension, for discrete parameters (*e.g.* potential destinations or lanes for a vehicle). In Chapter 5, we embrace this trade-off even more explicitly. Rather than trying to adjust the scope and size of the uncertainty, we instead directly control its influence on both the efficiency of the policy (rewards) and its safety (costs), by training a *budgeted* policy $\overline{\pi}$ that takes as input the desired

level of risk δ. While this setting was previously studied for finite states and known dynamics only, we extended it to continuous states and unknown dynamics.

Sample efficiency As for most reinforcement learning problems, we were also concerned about minimising the number of samples required to reach optimality. To that end, we exploited the specificities and structures of the behavioural planning problem in several ways. In Chapter 4, we embedded an inductive bias into the policy architecture by enforcing its invariance to permutations of the scene description, and observed that this fastens learning. In Chapter 7, some structure was similarly imposed –on the dynamics model this time– in the form of a parametrised linear model, which allowed to reduce the dimension of the hypothesis space significantly. We were also able to provide a bound on the simple regret relating the agent performance to the number of observed transition samples. In Chapter 6, we looked into the sample efficiency of the planning procedure, and specifically tree-based planning algorithms. First, in the case of stochastic dynamics representing human behaviours, we proposed a modification of the `OLOP` algorithm that improves its empirical sample complexity by an order on magnitude, while retaining its theoretical guarantees. Second, we showed that merging similar nodes in the lookahead tree enables to decrease the near-optimal branching factor featured in the regret bound of the algorithm. This translated into substantial empirical improvements in simple path planning tasks, where distinct sequences of actions lead to overlapping trajectories.

8.2 Outstanding issues and perspectives

In this section, I will adopt a more personal and subjective standpoint, and discuss which are the main barriers between research and industrialisation. Indeed, though we never intended for this thesis to lead directly to practical applications, they remain the long term goal that motivates our work, and I must now examine our contributions again in this light.

A first and general concern of mine is that, beside the warm comfort of the well-behaved theoretical frameworks in which we place ourselves, the sheer complexity of the real world can be overwhelming. While any single aspect –*partial observability, temporal abstraction, non-stationarity, risk aversion*, you name it– can be isolated and studied independently, the question of how to merge all these approaches into one single integrated product seems arduous, if not hopeless. Yet, any candidate algorithm not addressing any of these issues would be unfit for deployment. In the sequel, I will not be so ambitious but reflect instead on a more reasonable question: are the methods that we developed suitable for a real-world application?

Reinforcement Learning in continuous states Let us start with our work in Part II. Following a model-free perspective with continuous states, we resorted to function approximation

using neural networks. Unfortunately, Deep Learning interacts with Reinforcement Learning algorithms in ways that are yet to be understood, but already infamous for their brittleness. In Chapter 4, even our best policies still suffer a prohibitive rate of collisions of 6 %, considerably higher than the required performances. As we discussed, this may be attributed to the reward function that would not penalise collisions enough, but reward engineering is tedious and might in turn lead to over-conservative policies. The budgeted approaches of Chapter 5 were meant to address this issue, but at the price of increased complexity, and our negative result of Theorem 5.9 raises concerns about convergence in the general case. Another weakness lies in our use of a very naive exploration policy: the ε-greedy, which takes random actions at a fixed frequency, a strategy widely considered inefficient and damaging, yet one that we had no choice but to resort to in the absence of a better solution. Guided exploration strategies tailored for regret minimisation, and especially following the OFU principle, have been studied in the context of finite state-action spaces (Auer, Jaksch, and Ortner, 2009; Azar, Osband, and Rémi Munos, 2017). These methods typically require the ability to count the number of state visits, which is not suitable for continuous states. The question of how they can be extended thus constitutes a promising research perspective. First steps have recently been made in that direction, by either relying on approximated *pseudo-counts* (Tang et al., 2017), or by deriving similar regret bounds under linear function approximation (Jin et al., 2020).

Trial without error? Assume for a moment that the research community was able to solve the aforementioned problem and came up with exciting new algorithms for continuous state space with promising regret bounds. There remains an inevitable and fatal limitation: *the foundations of Reinforcement Learning are intrinsically based on trial and error*. Unfortunately, this is not an acceptable paradigm for the development of safety-critical problems such as Autonomous Driving. For reference, when we applied the model-free methods of Part II to very simple tasks, they converged in about 50k interaction samples, which represents about 15 h of driving, throughout which the agents experienced about two thousand collisions. More generally, having vehicles *exploring* on the roads among human drivers is morally inconceivable. Is there any chance at all to come up with a learning algorithm that does not require causing accidents while training?

Safety guarantees As a first candidate, the line of work on *safe* control is committed to developing algorithms that are guaranteed never to reach an unsafe state, or with a provably bounded probability of failure. Likewise, in Chapter 7, we managed to obtain some theoretical guarantees: a robust constraint satisfaction result, and a lower-bound on the worst-case outcome, that increases towards near-optimal performance with the number of samples. However, it is evident that these results are only worth as much as their underlying assumptions, which may turn out to be: very little. Indeed, it seems dubious that the complexity of human behaviours

can be accurately described by linear dynamics Assumption 7.4, and our own proposed system largely overlooks vast areas of the driving task. With Assumption 7.24, the safe region $\mathbb{X}$ cannot be chosen freely, but must contain the basin of attraction $\mathbb{X}_f$ whose size grows with uncertainty. Finally, the assumptions of Theorem 7.30 do not hold even in our simple simulations: the behaviours of observed vehicles are not persistently excited (*i.e.* continually changing lanes, or braking behind some vehicle), and the reward function is discontinuous. More generally, safety analyses can never protect against unmodeled events, such as a tree or a package falling down the road. Yet, having to model the world in its full complexity is daunting, especially since theoretical analysis imposes an additional constraint on the modelling effort, often at the expense of empirical performance. Fortunately, all is not bleak and it has been observed in numerous occasions that even when the guarantees do not hold, the founding principles of an algorithm can lead to designs that still exhibit the desired properties. Beside this issue of the practical validity of theoretical assumptions, it seems to me that none of the most popular safety frameworks is really suitable for AD. First, the ubiquitous concept of stabilisation does not really apply to a task of motion planning amidst other vehicles, for which there is no obvious equilibrium state. Second, the robust constraint satisfaction paradigm may be more relevant if the constraint space $\mathbb{X}$ could be defined as the collision-free space, but that space is non-convex which is not handled by most algorithms, *e.g.* in the Tube MPC family. Third, the minimax setting where worst-case outcomes are considered can become worthless in situations of large uncertainty where any decision can possibly result in a collision. In that case, the loss of sensitivity to probabilities caused to the $\min$ formulation means that decisions that are *less likely* to lead to a collision will not even be preferred to those that are *more likely* to do so. This ability requires to replace the worst-case evaluation by a more sophisticated measure of the outcomes distribution, such as the VaR or CVaR. This perspective seems a promising research direction to me, since to my knowledge no algorithm achieves guaranteed performance for these risk measures with continuous states, but it is also presumably a very demanding one.

Simulation and beyond? Another enticing way to avoid trial-and-error in the real world is to rely on simulation. Of course, the effort of modelling a complex world remains, but dropping the analysability requirement relaxes the modelling constraints. We can safely expect that simulations will continue to play an increasingly significant part in AD technologies, for both offline pre-training and online planning. This can be the occasion to divert our research efforts from the traditional regret minimisation objective, which is groundless in a simulated environment where failures are free but samples are costly. In contrast, a promising research direction is the study of the more appropriate *pure exploration* setting, which aims at relating the policy suboptimality to the number of samples used. I already took part in collaborations exploring this direction (Jonsson et al., 2020; Kaufmann, Ménard, et al., 2020; Ménard et al., 2020) and hope to pursue this path further. Finally, relying on simulation introduces the

additional question of how to adapt knowledge from simulation to the real world. A fine-tuning training process in real conditions would involve experiencing real failures again, though hopefully in reduced numbers, which could be realistic under human interventions (Saunders et al., 2018; Kendall et al., 2019). Another path of interest to me could be to leverage offline RL methods (Thomas, Theocharous, and Ghavamzadeh, 2015; Laroche, Trichelair, and Combes, 2019), that could enable to safely improve pre-trained policies using real driving data, especially around nominal states that can be confidently estimated.

Appendix A

The HIGHWAY-ENV software

Contents

A.1 General presentation

HIGHWAY-ENV is a collection of environments for behavioural planning tasks in autonomous driving.

Each environment specifies a full MDP to describe a particular decision-making problem that an autonomous vehicle may face.

Origins When I started my Ph.D., there existed more ambitious open-source simulators that relied on heavy physics engines and 3D graphics, such as TORCS (Wymann et al., 2015), Airsim (Shah et al., 2017) and CARLA (Dosovitskiy et al., 2017). However, those were better suited for low-level sensing and control, *e.g.* training of visuomotor policies in a single-agent setting. On the other hand of the spectrum, SUMO (Lopez et al., 2018) was rather meant for high-scale traffic optimisation and lacked details and flexibility on local dynamics. In contrast, I needed a simulator focused on high-level decisions and vehicle-to-vehicle interactions. Consequently, I launched HIGHWAY-ENV with the intent of having a minimalist simulator, implemented fully in Python for fast prototyping and easy interfacing with RL libraries.

Usage We show below a basic use of HIGHWAY-ENV, with a code snipped showing the interaction between the environment, which generates observations and rewards, and the agent which provides actions according to its policy.

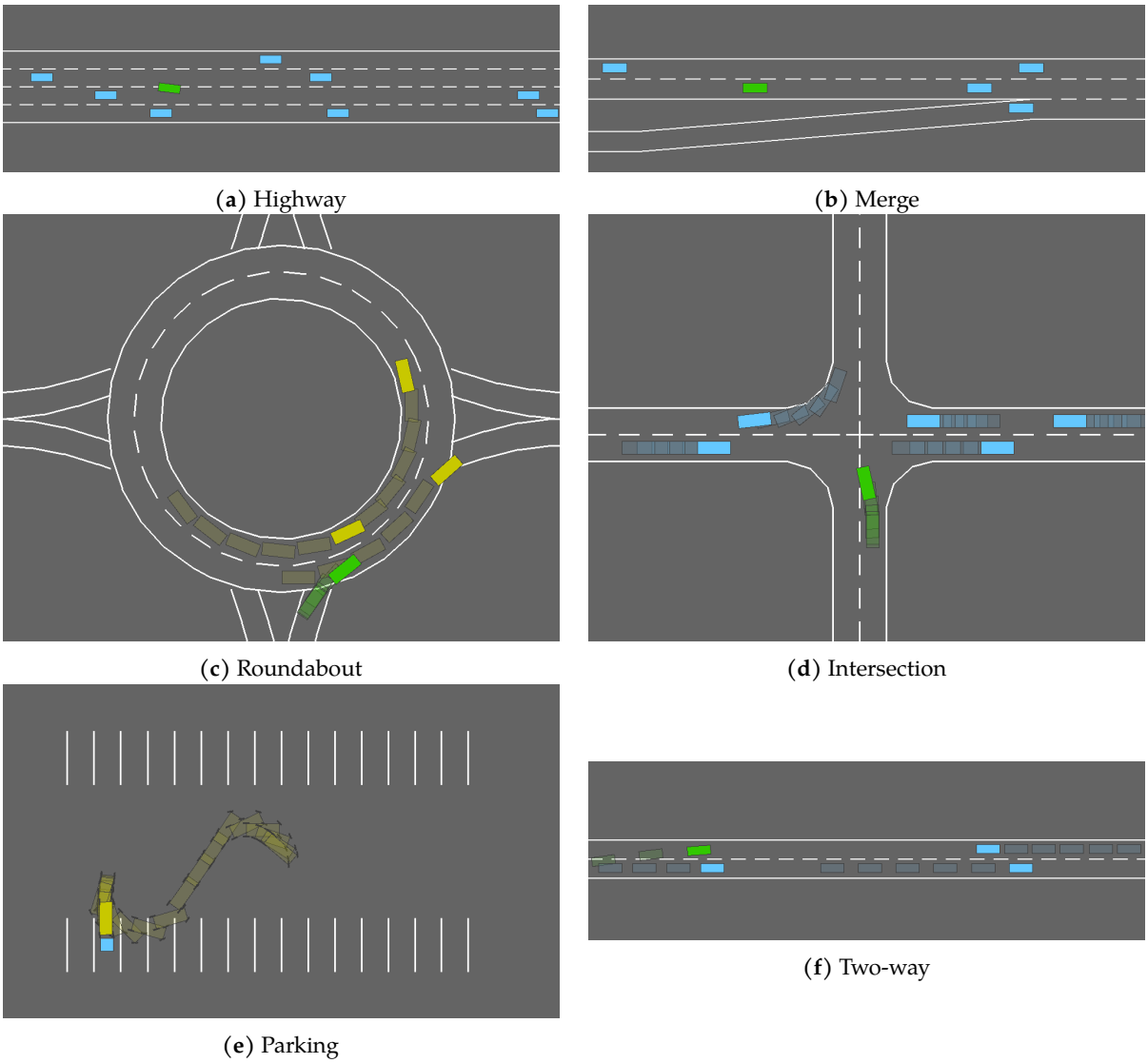
```
import gym
import highway_env

env = gym.make('highway-v0')

done = False
while not done:
    action = ... # Your agent code here
    obs, reward, done, info = env.step(action)
    env.render()
```

Listing A.1 – Create, step and render the `highway-v0` environment.

The environments To this day, HIGHWAY-ENV comes with six different scenes, configured with suitable observation space $\mathcal{S}$, action space $\mathcal{A}$ and reward function R, illustrated in Table A.1.

- **Highway.** The vehicle is driving on a highway populated with other drivers. The goal is to drive as fast as possible while avoiding collisions with other vehicles, through a discrete meta-action space of manoeuvres.
- **Merge.** The task is similar to `highway-v0`, but a vehicle is incoming from an access ramp and must be able to merge successfully in traffic. To that end, the ego-vehicle must change lane or adapt its velocity so that the merging vehicle has sufficient space. This task is inspired by a practical use case for ADAS systems at Renault.
- **Roundabout.** The vehicle must cross a roundabout as fast as possible while avoiding collisions. It requires reasoning about the uncertain destinations of other vehicles.
- **Intersection.** This environment is similar to roundabout, but with an increased density of vehicles and types of conflicts. Only the throttle is controlled, and the steering is performed automatically to track the ego-vehicle destination.
- **Parking.** A goal-conditioned continuous control environment: the desired parking spot location is part of the observation, and the ego-vehicle must plan cusp-shaped manoeuvres to reach it with the proper heading.
- **Two-way.** This environment is similar to highway, except that the ego-vehicle can change to a lane facing the opposite direction, with incoming vehicles. This enables to highlight an efficiency-safety trade-off for risk-sensitive decision-making.

Features Several parts of the environment can be configured.



(**a**) Highway

(**b**) Merge

(**c**) Roundabout

(**d**) Intersection

(**e**) Parking

(**f**) Two-way

Table A.1 – The different environments available in HIGHWAY-ENV.

- **Observations.** Several types of observations are available, such as the *list of features* and *occupancy grid* described in Chapter 4. Other types include RGB images, time-to-collision maps, and goal locations.
- **Actions.** In addition to the discrete meta-action space $\mathcal{A}$ described in Chapter 3, a continuous space $\mathcal{A} = \mathbb{R}^2$ for throttle and steering can also be selected.
- **Dynamics.** Other vehicles can follow the IDM and MOBIL behavioural models as described in Chapter 3 or its linearised version of Section 7.3. The ego-vehicle can be controlled with either the Kinematic Bicycle Model as in Chapter 3 or the Dynamic Bicycle Model as in Section 7.4.

- **Rewards.** The rewards and penalties associated with speed or collisions are configurable.
- **Graphics.** Graphics are rendered using the pygame library, window size and resolution can be configured.

Development process The project closely follows the insights of Chapter 3 in its definition of the traffic state, actions, dynamics, and rewards. See the user guide in the documentation for more details.

It also complies by the OpenAI gym standard interface for RL environments. Continuous integration (CI) is performed with pytest unit tests automatically triggered with GitHub actions. The documentation is built from the source code comments using Sphinx, and hosted on Read the Docs. Finally, multiple examples can be run directly from the browser through Google Colab notebooks.

A.2 Outreach

In this section, we document the dissemination of the library among several communities.

A.2.1 Academia

We highlight that several researchers have reported using or referred to HIGHWAY-ENV in their publications and submissions:

- Minne Li, Lisheng Wu, Jun WANG, and Haitham Bou Ammar (Dec. 2019). Multi-View Reinforcement Learning. In *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vancouver, BC, Canada: Curran Associates, Inc., pp. 1420–1431
- Yaodong Yang, Rasul Tutunov, Phu Sakulwongtana, and Haitham Bou-Ammar (May 2020). α^α-Rank: Practically Scaling α-Rank through Stochastic Optimisation. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*. Ed. by Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith. Auckland, New-Zealand: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1575–1583
- Sriram N N, Buyu Liu, Francesco Pittaluga, and Manmohan Chandraker (Aug. 2020). SMART: Simultaneous Multi-Agent Recurrent Trajectory Prediction. In *16th European Conference on Computer Vision (ECCV 2020)*. Glasgow, United Kingdom

- Mengdi Xu, Wenhao Ding, Jiacheng Zhu, Zuxin Liu, Baiming Chen, and Ding Zhao (2020). *Task-Agnostic Online Reinforcement Learning with an Infinite Mixture of Gaussian Processes*. preprint
- Xiaoteng Ma, Li Xia, Zhengyuan Zhou, Jun Yang, and Qianchuan Zhao (June 2020). DSAC: Distributional Soft Actor Critic for Risk-Sensitive Reinforcement Learning. In *Reinforcement Learning for Real Life Workshop at ICML 2019*. Long Beach, CA, USA
- Jincheng Mei, Yangchen Pan, Martha White, Amir-massoud Farahmand, and Hengshuai Yao (2020). *Beyond Prioritized Replay: Sampling States in Model-Based RL via Simulated Priorities*. preprint
- Haifeng Zhang et al. (Feb. 2020). Bi-level Actor-Critic for Multi-agent Coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. New York, pp. 7325–7332
- Angelos Mavrogiannis, Rohan Chandra, and Dinesh Manocha (2020). *B-GAP: Behavior-Guided Action Prediction for Autonomous Navigation*. preprint.
- Ming Zhou et al. (Nov. 2020). SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. In *Conference on Robot Learning (CoRL)*
- Justin K. Terry et al. (2020). *PettingZoo: Gym for Multi-Agent Reinforcement Learning*. preprint.
- Parv Kapoor, Anand Balakrishnan, and Jyotirmoy V. Deshmukh (2020). *Model-based Reinforcement Learning from Signal Temporal Logic Specifications*. preprint.
- S. Zhang, Y. Wu, and H. Ogai (Sept. 2020). Spatial Attention for Autonomous Decision-making in Highway Scene. In *59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. Chiang Mai, Thailand, pp. 1435–1440
- Jincheng Mei, Yangchen Pan, Martha White, Amir-massoud Farahmand, and Hengshuai Yao (2020). *Beyond Prioritized Replay: Sampling States in Model-Based RL via Simulated Priorities*. preprint

Additionally, we use it in most of our own publications:

- Edouard Leurent, Yann Blanco, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2018). Approximate Robust Control of Uncertain Dynamical Systems. In *Machine Learning for Intelligent Transportation Systems Workshop at the Thirty-second Conference on Neural Information Processing Systems (NeurIPS 2018)*. Montreal, Canada

- Edouard Leurent, Denis Efimov, Tarek Raissi, and Wilfrid Perruquetti (Dec. 2019). Interval Prediction for Continuous-Time Systems with Parametric Uncertainties. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. Nice, France, pp. 7049–7054
- Edouard Leurent and Jean Mercat (Dec. 2019). Social Attention for Autonomous Decision-Making in Dense Traffic. In *Machine Learning for Autonomous Driving Workshop at the Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019)*. Montreal, Canada
- Nicolas Carrara, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Odalric-Ambrym Maillard, and Olivier Pietquin (Dec. 2019). Budgeted Reinforcement Learning in Continuous State Space. In *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 9299–9309
- Edouard Leurent and Odalric-Ambrym Maillard (Sept. 2020b). Practical Open-Loop Optimistic Planning. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet. Würzburg, Germany: Springer International Publishing, pp. 69–85
- Edouard Leurent, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2020b). Robust-Adaptive Interval Predictive Control for Linear Uncertain Systems. In *2020 IEEE 59th Conference on Decision and Control (CDC)*. Jeju Island, Republic of Korea
- Edouard Leurent, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2020a). Robust-Adaptive Control of Linear Systems: beyond Quadratic Costs. In *Advances in Neural Information Processing Systems 33*. Virtual

Finally, the HIGHWAY-ENV library is featured in the documentation and examples of two of the most famous libraries in the RL ecosystem:

- OpenAI Gym, a standard interface for comparing RL algorithms. HIGHWAY-ENV is mentioned in the list of environments.
- Stable Baselines, a set of improved implementations of RL algorithms based on OpenAI Baselines. HIGHWAY-ENV is used as an example in the documentation.

A.2.2 Education

Since the publication of HIGHWAY-ENV, many masters students have contacted me throughout the years, and reported using the simulator for their final project of in various courses, occasionally

upon suggestion by their professor. I did not keep track of all these exchanges, but as a trace of this activity, see:

- the list of issues opened in HIGHWAY-ENV.
- the list of issues opened in RL-AGENTS.

A.2.3 Industry

Several engineers, mainly from the Automotive industry, have demonstrated interest in HIGHWAY-ENV, either by reaching out to me directly or by developing their own project on top of the library. To name a few,

- Simon Chauvin, Machine Learning Engineer at Autonomous Driving at ESR Labs GmbH, who contacted me to reproduce my results.
- Clément Huber, Product Owner in the Path Planning team at NAVYA Group, and Lucas Boyer, intern in that team, contacted me on several occasions to discuss the internship of Lucas based on HIGHWAY-ENV and OPENAI/BASELINES.
- Munir Jojo-Verge, Motion Planning and Decision Making Manager at Amazon Robotics, who wrote to me regarding his extension of HIGHWAY-ENV to continuous actions, see his fork which lead to the addition of the `parking-v0` environment.
- Craig Quiter, Founder of Deepdrive at Voyage, who mentioned that his recent work was inspired by HIGHWAY-ENV and with whom I had the pleasure to discuss.
- Pinaki Gupta, Behaviour Planning architect at Lucid Motors, see his fork that features variants of the `two-way-v0` and `parking-v0` environments augmented with additional vehicles and multiple goals.
- Guillaume Alleon, Head of AI research at Airbus, see his fork.
- Boris Yangel, Principal Software Engineer at Yandex, see his project.

Appendix B

Complements on Chapter 5

B.1 Proofs

B.1.1 Proof of Proposition 5.4

Proof. Thanks to the introduction of the augmented spaces $\overline{\mathcal{S}}, \overline{\mathcal{A}}$ and dynamics $\overline{P}$, this proof is the same as that in classical MOMDPs.

$$\begin{aligned}
\overline{V}^{\overline{\pi}}(\overline{s}) &\triangleq \mathbb{E}\left[\overline{G}^{\overline{\pi}} \;\middle|\; \overline{s_0} = \overline{s}\right] \\
&= \sum_{\overline{a} \in \overline{\mathcal{A}}} \mathbb{P}\left(\overline{a}_0 = \overline{a} \mid \overline{s_0} = \overline{s}\right) \mathbb{E}\left[\overline{G}^{\overline{\pi}} \;\middle|\; \overline{s_0} = \overline{s}, \overline{a}_0 = \overline{a}\right] \\
&= \sum_{\overline{a} \in \overline{\mathcal{A}}} \overline{\pi}(\overline{a}|\overline{s}) \overline{Q}^{\overline{\pi}}(\overline{s}, \overline{a}).
\end{aligned}$$

$$\begin{aligned}
\overline{Q}^{\overline{\pi}}(\overline{s}, \overline{a}) &\triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \overline{R}(\overline{s}_t, \overline{a}_t) \;\middle|\; \overline{s_0} = \overline{s}, \overline{a_0} = \overline{a}\right] \\
&= \overline{R}(\overline{s}, \overline{a}) + \sum_{\overline{s}' \in \overline{\mathcal{S}}} \mathbb{P}\left(\overline{s}_1 = \overline{s}' \mid \overline{s_0} = \overline{s}, \overline{a_0} = \overline{a}\right) \cdot \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t \overline{R}(\overline{s}_t, \overline{a}_t) \;\middle|\; \overline{s_1} = \overline{s}'\right] \\
&= \overline{R}(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \overline{P}\left(\overline{s}' \mid \overline{s}, \overline{a}\right) \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \overline{R}(\overline{s}_t, \overline{a}_t) \;\middle|\; \overline{s_0} = \overline{s}'\right] \\
&= \overline{R}(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} \overline{P}\left(\overline{s}' \mid \overline{s}, \overline{a}\right) \overline{V}^{\overline{\pi}}(\overline{s}').
\end{aligned}$$

Complements on Chapter 5

Contraction of $\overline{\mathcal{T}}^{\overline{\pi}}$. Let $\overline{\pi} \in \overline{\Pi}, \overline{Q}_1, \overline{Q}_2 \in (\mathbb{R}^2)^{\overline{\mathcal{SA}}}$.

$$\forall \overline{s} \in \overline{\mathcal{S}}, \overline{a} \in \overline{\mathcal{A}}, \quad \left|\overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_1(\overline{s},\overline{a}) - \overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_2(\overline{s},\overline{a})\right| = \left|\gamma \mathop{\mathbb{E}}_{\substack{\overline{s}'\sim\overline{P}(\overline{s}'|\overline{s},\overline{a}) \\ \overline{a}'\sim\overline{\pi}(\overline{a}'|\overline{s}')}} \overline{Q}_1(\overline{s}',\overline{a}') - \overline{Q}_2(\overline{s}',\overline{a}')\right|$$
$$\leq \gamma \left\|\overline{Q}_1 - \overline{Q}_2\right\|_\infty.$$

Hence, $\left\|\overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_1 - \overline{\mathcal{T}}^{\overline{\pi}}\overline{Q}_2\right\|_\infty \leq \gamma \left\|\overline{Q}_1 - \overline{Q}_2\right\|_\infty$

According to the Banach fixed point theorem (Banach, 1922), $\overline{\mathcal{T}}^{\overline{\pi}}$ admits a unique fixed point. It can be easily verified that $\overline{Q}^{\overline{\pi}}$ is indeed this fixed point by combining the two Bellman Expectation equations (5.6).

□

B.1.2 Proof of Theorem 5.6

Proof. Let $\overline{s}, \overline{a} \in \overline{\mathcal{A}} \times \overline{\mathcal{S}}$. For this proof, we consider potentially non-stationary policies $\overline{\pi} = (\rho, \overline{\pi}')$, with $\rho \in \mathcal{M}(\overline{\mathcal{A}})$, $\overline{\pi}' \in \mathcal{M}(\overline{\mathcal{A}})^{\mathbb{N}}$. The results will apply to the particular case of stationary optimal policies, when they exist.

$$\overline{Q}_r^\star(\overline{s},\overline{a}) = \max_{\rho,\overline{\pi}'} Q_r^{\rho,\overline{\pi}'}(\overline{s}',\overline{a}') \tag{B.1}$$
$$= \max_{\rho,\overline{\pi}'} R(s,a) + \gamma \sum_{\overline{s}'\in\overline{\mathcal{S}}} \overline{P}(\overline{s}'|\overline{s},\overline{a}) V_r^{\rho,\overline{\pi}'}(\overline{s}') \tag{B.2}$$
$$= R(s,a) + \gamma \sum_{\overline{s}'\in\overline{\mathcal{S}}} \overline{P}(\overline{s}'|\overline{s},\overline{a}) \max_{\rho,\overline{\pi}'} \sum_{\overline{a}'\in\overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') Q_r^{\overline{\pi}'}(\overline{s}',\overline{a}') \tag{B.3}$$
$$= R(s,a) + \gamma \sum_{\overline{s}'\in\overline{\mathcal{S}}} \overline{P}(\overline{s}'|\overline{s},\overline{a}) \max_{\rho} \sum_{\overline{a}'\in\overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') \max_{\overline{\pi}'\in\overline{\Pi}_a(\overline{s}')} Q_r^{\overline{\pi}'}(\overline{s}',\overline{a}') \tag{B.4}$$
$$= R(s,a) + \gamma \sum_{\overline{s}'\in\overline{\mathcal{S}}} \overline{P}(\overline{s}'|\overline{s},\overline{a}) \max_{\rho} \mathop{\mathbb{E}}_{\overline{a}'\sim\rho} Q_r^\star(\overline{s}',\overline{a}') \tag{B.5}$$

where $\overline{\pi} = (\rho, \overline{\pi}') \in \overline{\Pi}_a(\overline{s})$ and $\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')$.

This follows from:

(B.1). Definition of $\overline{Q}^\star$.

(B.2). Bellman Expectation expansion from Proposition 5.4.

(B.3). Marginalisation on $\overline{a}'$.

(B.4). • Trivially $\max_{\overline{\pi}'\in\overline{\Pi}_a(\overline{s}')} \sum_{\overline{a}'\in\overline{\mathcal{A}}} \cdot \leq \sum_{\overline{a}'\in\overline{\mathcal{A}}} \max_{\overline{\pi}'\in\overline{\Pi}_a(\overline{s})} \cdot$.

- Let $\overline{\overline{\pi}} \in \arg\max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')} Q_r^{\overline{\pi}'}(\overline{s}', \overline{a}')$, then:

$$\sum_{\overline{a}' \in \overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') \max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')} Q_r^{\overline{\pi}'}(\overline{s}', \overline{a}') = \sum_{\overline{a}' \in \overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') Q_r^{\overline{\pi}'}(\overline{s}', \overline{a}') \leq \max_{\overline{\pi}' \in \overline{\Pi}_a(\overline{s}')} \sum_{\overline{a}' \in \overline{\mathcal{A}}} \rho(\overline{a}'|\overline{s}') Q_r^{\overline{\pi}'}(\overline{s}', \overline{a}').$$

(B.5). Definition of $\overline{Q}^\star$.

Moreover, the condition $\overline{\pi} = (\rho, \overline{\pi}') \in \overline{\Pi}_a(\overline{s})$ gives

$$\mathop{\mathbb{E}}_{\overline{a}' \sim \rho} Q_c^\star(\overline{s}, \overline{a}) = \mathop{\mathbb{E}}_{\overline{a}' \sim \rho} Q_c^{\overline{\pi}'}(\overline{s}, \overline{a}) = V_c^{\overline{\pi}}(\overline{s}) \leq \beta.$$

Consequently, $\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^\star)$ belongs to the $\arg\max$ of (B.5), and in particular:

$$Q_r^\star(\overline{s}, \overline{a}) = r(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}' \in \overline{\mathcal{S}}} P(\overline{s}'|\overline{s}, \overline{a}) \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{s}', \overline{Q}^\star)} Q_r^\star(\overline{s}', \overline{a}').$$

The same reasoning can be made for $Q_c^\star$ by replacing $\max$ operators by $\min$, and $\overline{\Pi}_a$ by $\overline{\Pi}_r$. □

B.1.3 Proof of Proposition 5.8

Proof. Notice from the definitions of $\overline{\mathcal{T}}^\star$ and $\overline{\mathcal{T}}^{\overline{\pi}}$ in (5.12) and (5.7) that $\overline{\mathcal{T}}^\star$ and $\overline{\mathcal{T}}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^\star)}$ coincide on $\overline{Q}^\star$. Moreover, since $\overline{Q}^\star = \overline{\mathcal{T}}^\star \overline{Q}^\star$ by Theorem 5.6, we have: $\overline{\mathcal{T}}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^\star)} \overline{Q}^\star = \overline{\mathcal{T}}^\star \overline{Q}^\star = \overline{Q}^\star$. Hence, $\overline{Q}^\star$ is a fixed point of $\overline{\mathcal{T}}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^\star)}$, and by Proposition 5.4 it must be equal to $\overline{Q}^{\overline{\pi}_{\text{greedy}}(\cdot; \overline{Q}^\star)}$

To show the same result for $\overline{V}^\star$, notice that

$$\overline{V}^{\overline{\pi}_{\text{greedy}}(\overline{Q}^\star)}(\overline{s}) = \mathop{\mathbb{E}}_{\overline{a} \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^\star)} \overline{Q}^{\overline{\pi}_{\text{greedy}}(\overline{Q}^\star)}(\overline{s}, \overline{a}) = \mathop{\mathbb{E}}_{\overline{a} \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^\star)} \overline{Q}^\star(\overline{s}, \overline{a}).$$

By applying the definitions of $\overline{Q}^\star$ and $\overline{\pi}_{\text{greedy}}$, we recover the definition of $\overline{V}^\star$. □

B.1.4 Proof of Theorem 5.9

Proof. In the trivial case $|\mathcal{A}| = 1$, there exits only one policy $\overline{\pi}$ and $\overline{\mathcal{T}} = \overline{\mathcal{T}}^{\overline{\pi}}$, which is a contraction by Proposition 5.4.

In the general case $|\mathcal{A}| \geq 2$, we can build the following counter-example.

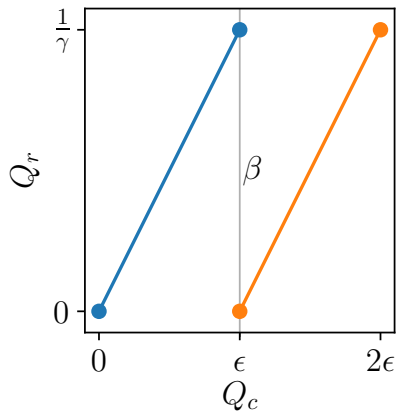


Figure B.1 – Representation of $\overline{Q}^1_\varepsilon$ (blue) and $\overline{Q}^2_\varepsilon$ (yellow)

Let $(\mathcal{S}, \mathcal{A}, P, R_r, R_c)$ be a BMDP. For any $\varepsilon > 0$, we define $\overline{Q}^1_\varepsilon$ and $\overline{Q}^2_\varepsilon$ as

$$\overline{Q}^1_\varepsilon(\overline{s}, \overline{a}) = \begin{cases} (0, 0), & \text{if } a = a_0 \\ \left(\frac{1}{\gamma}, \varepsilon\right), & \text{if } a \neq a_0 \end{cases}$$

$$\overline{Q}^2_\varepsilon(\overline{s}, \overline{a}) = \begin{cases} (0, \varepsilon), & \text{if } a = a_0 \\ \left(\frac{1}{\gamma}, 2\varepsilon\right), & \text{if } a \neq a_0 \end{cases}$$

Then, $\|\overline{Q}_1 - \overline{Q}_2\|_\infty = \varepsilon$. $\overline{Q}^1_\varepsilon$ and $\overline{Q}^2_\varepsilon$ are represented in Figure B.1.

But for $\overline{a} = (a, \beta_a)$ with $\beta_a = \varepsilon$, we have

$$\begin{aligned} \|\overline{\mathcal{T}}\overline{Q}^1_\varepsilon(\overline{s}, \overline{a}) - \overline{\mathcal{T}}\overline{Q}^2_\varepsilon(\overline{s}, \overline{a})\|_\infty &= \gamma \left\| \mathop{\mathbb{E}}_{\overline{s}' \sim \overline{P}(\overline{s}'|\overline{s},\overline{a})} \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^1_\varepsilon)} \overline{Q}^1_\varepsilon(\overline{s}', \overline{a}') - \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^2_\varepsilon)} \overline{Q}^2_\varepsilon(\overline{s}', \overline{a}') \right\|_\infty \\ &= \gamma \left\| \mathop{\mathbb{E}}_{\overline{s}' \sim \overline{P}(\overline{s}'|\overline{s},\overline{a})} \left(\frac{1}{\gamma}, \varepsilon\right) - (0, \varepsilon) \right\|_\infty \\ &= \gamma \frac{1}{\gamma} = 1 \end{aligned}$$

Hence,

$$\|\overline{\mathcal{T}}\overline{Q}^1_\varepsilon - \overline{\mathcal{T}}\overline{Q}^2_\varepsilon\|_\infty \geq 1 = \frac{1}{\varepsilon}\|\overline{Q}_1 - \overline{Q}_2\|_\infty$$

In particular, there does not exist $L > 0$ such that

$$\forall \overline{Q}_1, \overline{Q}_2 \in (\mathbb{R}^2)^{\overline{\mathcal{SA}}}, \|\overline{\mathcal{T}}\overline{Q}^1 - \overline{\mathcal{T}}\overline{Q}^2\|_\infty \leq L\|\overline{Q}^1 - \overline{Q}^2\|_\infty$$

In other words, $\overline{\mathcal{T}}$ is not a contraction for $\|\cdot\|_\infty$. □

B.1.5 Proof of Theorem 5.10

> **Remark B.1.** *This proof makes use of insights detailed in the proof of Proposition 5.11 (Section B.1.6), which we recommend the reader to consult first.*

Proof. We now study the contractivity of $\overline{\mathcal{T}}^\star$ when restricted to the functions of $\mathcal{L}_\gamma$ defined as follows:

$$\mathcal{L}_\gamma = \left\{ \begin{array}{l} \overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{SA}}} \text{ s.t. } \exists L < \frac{1}{\gamma} - 1 : \forall \overline{s} \in \overline{\mathcal{S}}, \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \\ |Q_r(\overline{s}, \overline{a}_1) - Q_r(\overline{s}, \overline{a}_2)| \leq L |Q_c(\overline{s}, \overline{a}_1) - Q_c(\overline{s}, \overline{a}_2)| \end{array} \right\}. \tag{B.6}$$

That is, for all state $\overline{s}$, the set $\overline{Q}(\overline{s}, \overline{\mathcal{A}})$ plot in the (Q_c, Q_r) plane must be the *graph* of a L-Lipschitz function, with $L < 1/\gamma - 1$.

We impose such structure for the following reason: the counter-example presented above prevented contraction because it was a pathological case in which the slope of $\overline{Q}$ can be arbitrary large. As a consequence, when solving $Q_r^\star$ such that $Q_c^\star = \beta$, a vertical slice of a $\|\cdot\|_\infty$ ball around $\overline{Q}_1$ (which must contain $\overline{Q}_2$) can be arbitrary large as well.

We denote $\text{Ball}(\overline{Q}, R)$ the ball of centre $\overline{Q}$ and radius R for the $\|\cdot\|_\infty$-norm:

$$\text{Ball}(\overline{Q}, R) = \{\overline{Q}' \in (R^2)^{\overline{\mathcal{SA}}} : \|\overline{Q} - \overline{Q}'\|_\infty \leq R\}.$$

We give the three main steps required to show that $\overline{\mathcal{T}}^\star$ restricted to $\mathcal{L}_\gamma$ is a contraction. Given $\overline{Q}^1, \overline{Q}^2 \in \mathcal{L}_\gamma$, show that:

1. $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1, R) \implies \mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R), \forall \overline{s} \in \overline{\mathcal{S}}$, where $\mathcal{F}$ is the top frontier of the convex hull of undominated points, as defined in Section B.1.6.
2. $\overline{Q} \in \mathcal{L}_\gamma \implies \mathcal{F}$ is the graph of a L-Lipschitz function, $\forall \overline{s} \in \overline{\mathcal{S}}$.
3. taking the slice $Q_c = \beta$ of a ball $\text{Ball}(\mathcal{F}, R)$ with $\mathcal{F}$ L-Lipschitz results in an interval on Q_r of range at most $(L+1)R$

These three steps will allow us to control $Q_r^{2\star} - Q_r^{1\star}$ as a function of $R = \|\overline{Q}^2 - \overline{Q}^1\|_\infty$.

Step 1 We want to show that if $\overline{Q}^1$ and $\overline{Q}^2$ are close, then $\mathcal{F}^1$ are $\mathcal{F}^2$ are close as well in the following sense:

$$\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R) \iff d(\mathcal{F}^1, \mathcal{F}^2) \leq R \iff \max_{q^2 \in \mathcal{F}^2} \min_{q^1 \in \mathcal{F}^1} \|q^2 - q^1\|_\infty \leq R. \tag{B.7}$$

Assume $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1, R)$, we show by contradiction that $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R)$. Indeed, assume there exists $q^1 \in \mathcal{F}^1$ such that $\mathcal{F}^2 \cap \text{Ball}(q^1, R) = \emptyset$. Denote q^2 the unique point of $\mathcal{F}^2$ such that $q^2_c = q^1_c$. By construction of q^1, we know that $\|q^1 - q^2\|_\infty > R$. There are two possible cases:

- $q^2_r > q^1_r$: this also directly implies that $q^2_r > q^1_r + R$. But $q^2 \in \mathcal{F}^2$, so there exist $q^2_1, q^2_2 \in Q^2, \lambda \in \mathbb{R}$ such that $q^2 = (1-\lambda)q^2_1 + \lambda q^2_2$. But since $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1, R)$, there also exist $q^1_1, q^1_2 \in \overline{Q}^1$ such that $\|q^1_1 - q^2_1\|_\infty \leq R$ and $\|q^1_2 - q^2_2\|_\infty \leq R$, and in particular $q^1_{1r} \geq q^2_{1r} - R$ and $q^1_{2r} \geq q^2_{2r} - R$. But then, the point $q^{1'} = (1-\mu)q^1_1 + \mu q^1_2$ with $\mu = (q^2_c - q^1_{1c})/(q^2_{2c} - q^1_{1c})$ verifies $q^{1'}_c = q^1_c$ and $q^{1'}_r \geq q^2_r - R > q^1_r$ which contradicts the definition of $q_1 \in \mathcal{F}^1$ as defined in (B.12).
- $q^2_r < q^1_r$: then the same reasoning can be applied by simply swapping the indexes 1 and 2.

We have shown that $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R)$. This is illustrated in Figure B.2: given a function $\overline{Q}^1$, we show the locus $\text{Ball}(\overline{Q}_1, R)$ of $\overline{Q}^2$. We then draw $\mathcal{F}^1$ the top frontier of the convex hull of $\overline{Q}^1$ and alongside the locus of all possible $\mathcal{F}^2$, which belong to a ball $\text{Ball}(\mathcal{F}^1, R)$.

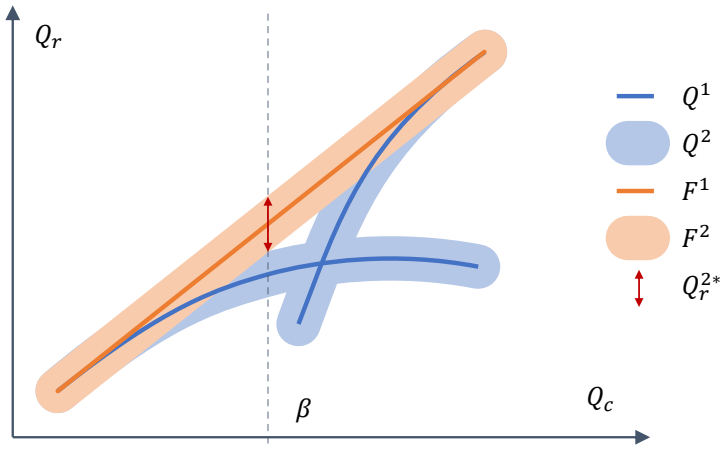


Figure B.2 – We represent the range of possible solutions $Q^{2\star}_r$ for any $\overline{Q}^2 \in \text{Ball}(\overline{Q}^1)$, given $\overline{Q}_1 \in \mathcal{L}_\lambda$

Step 2 We want to show that if $\overline{Q} \in \mathcal{L}_\gamma$, $\mathcal{F}$ is the graph of an L-Lipschitz function:

$$\forall q^1, q^2 \in \mathcal{F}, |q^2_r - q^1_r| \leq |q^2_c - q^1_c|. \tag{B.8}$$

Let $\overline{Q} \in \mathcal{L}_\gamma$ and $\overline{s} \in \overline{\mathcal{S}}$, $\mathcal{F}$ the corresponding top frontier of convex hull. For all $q^1, q^2 \in \mathcal{F}, \exists \lambda, \mu \in [0,1], q^{11}, q^{12}, q^{21}, q^{22} \in \overline{Q}(\overline{s}, \overline{\mathcal{A}})$ such that $q^1 = (1-\lambda)q^{11} + \lambda q^{12}$ and $q^2 = (1-\mu)q^{21} + \mu q^{22}$. Without loss of generality, we can assume $q^{11}_c \leq q^{12}_c$ and $q^{21}_c \leq q^{22}_c$. We also consider the worst case in terms of maximum q_r deviation: $q^{12}_c \leq q^{21}_c$. Then the maximum increment $q^2_r - q^1_r$

is:

$$\begin{aligned}\|q_r^2 - q_r^1\| &\leq \|q_r^{12} - q_r^1\| + \|q_r^{21} - q_r^{12}\| + \|q_r^2 - q_r^{21}\| \\ &= (1-\lambda)\|q_r^{12} - q_r^{11}\| + \|q_r^{21} - q_r^{12}\| + \mu\|q_r^{22} - q_r^{21}\| \\ &\leq (1-\lambda)L\|q_c^{12} - q_c^{11}\| + L\|q_c^{21} - q_c^{12}\| + \mu L\|q_c^{22} - q_c^{21}\| \\ &= L\|q_c^{12} - q_c^1\| + L\|q_c^{21} - q_c^{12}\| + L\|q_c^2 - q_c^{21}\| \\ &= L\|q_c^2 - q_c^1\|.\end{aligned}$$

This can also be seen in Figure B.2: the maximum slope of the $\mathcal{F}^1$ is lower than the maximum slope between two points of $\overline{Q}^1$.

Step 3 Let $\mathcal{F}_1$ be a L-Lipschitz set as defined in (B.8), and consider a ball $\text{Ball}(\mathcal{F}_1, R)$ around it as defined in (B.7).

We want to bound the optimal reward value $Q_r^{2\star}$ under constraint $Q_c^{2\star} = \beta$ (regular case in Section B.1.6 where the constraint is saturated), for any $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}_1, R)$. This quantity is represented as a red double-ended arrow in Figure B.2.

Because we are only interested in what happens locally at $Q_c = \beta$, we can zoom in on Figure B.2 and only consider a thin ε-section around β. In the limit $\varepsilon \to 0$, this section becomes the tangent to $\mathcal{F}^1$ at $Q_c^1 = \beta$. It is represented in Figure B.3, from which we derive a geometrical proof:

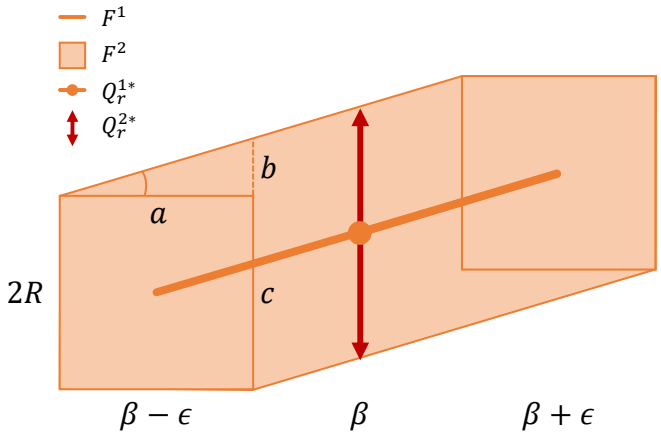


Figure B.3 – We represent a section $[\beta - \varepsilon, \beta + \varepsilon]$ of $\mathcal{F}^1$ and $\text{Ball}(\mathcal{F}^1, R)$. We want to bound the range of $Q_r^{2\star}$.

$$\begin{aligned}\Delta Q_r^{2\star} &= b + c \\ &\leq La + c && (\mathcal{F}^1 \ L\text{-Lipschitz})\end{aligned}$$

$$= 2LR + 2R = 2R(L+1).$$

Hence,

$$|Q_r^{2\star} - Q_r^{1\star}| \leq \frac{\Delta Q_r^{2\star}}{2} = R(L+1)$$

and $Q_c^{1\star} = Q_c^{2\star} = \beta$. Consequently, $\|\overline{Q}^{2\star} - \overline{Q}^{1\star}\|_\infty \leq (L+1)R$.

Finally, consider the edge case in Section B.1.6: the constraint is not active, and the optimal value is simply $\arg\max_{q \in \mathcal{F}} q^r$. In particular, since we showed that $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, R)$, and since $\overline{Q}^{2\star} \in \mathcal{F}^2$, there exist $q^1 \in \mathcal{F}^1 : \|\overline{Q}^{2\star} - q^1\|_\infty \leq R$ and in particular ${\overline{Q}^{1\star}}_r \geq q_r^1 \geq {\overline{Q}^{2\star}}_r - R$. Reciprocally, by the same reasoning, $Q_r^{2\star} \geq Q_r^{1\star} - R$. Hence, we have that $|Q_r^{2\star} - Q_r^{1\star}| \leq R \leq R(L+1)$.

Wrapping it up We have shown that for any $\overline{Q}^1, \overline{Q}^2 \in \mathcal{L}_\gamma$, and all $\overline{s} \in \overline{\mathcal{S}}$, $\mathcal{F}^2 \in \text{Ball}(\mathcal{F}^1, \|\overline{Q}^2 - \overline{Q}^1\|_\infty)$ and $\mathcal{F}^1$ is the graph of a L-Lipschitz function with $L < 1/\gamma - 1$. Moreover, the solutions of $\overline{\pi}_{\text{greedy}}(\overline{Q}^1)$ and $\overline{\pi}_{\text{greedy}}(\overline{Q}^2)$ at $\overline{s}$ are such that $\|\overline{Q}^{2\star} - \overline{Q}^{1\star}\|_\infty \leq (L+1)\|\overline{Q}^2 - \overline{Q}^1\|_\infty$.

Hence, for all $\overline{a}$,

$$\begin{aligned}
&\|\overline{\mathcal{T}}^\star \overline{Q}^1(\overline{s},\overline{a}) - \overline{\mathcal{T}}^\star \overline{Q}^2(\overline{s},\overline{a})\|_\infty \\
&\qquad = \gamma \left\| \mathop{\mathbb{E}}_{\overline{s}' \sim \overline{P}(\overline{s}'|\overline{s},\overline{a})} \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^1)} \overline{Q}^1(\overline{s}',\overline{a}') - \mathop{\mathbb{E}}_{\overline{a}' \sim \overline{\pi}_{\text{greedy}}(\overline{Q}^2)} \overline{Q}^2(\overline{s}',\overline{a}') \right\|_\infty \\
&\qquad = \gamma \left\| \overline{Q}^{2\star} - \overline{Q}^{1\star} \right\|_\infty \\
&\qquad \leq \gamma(L+1)\|\overline{Q}^2 - \overline{Q}^1\|_\infty.
\end{aligned}$$

Taking the sup on $\overline{\mathcal{SA}}$,

$$\|\overline{\mathcal{T}}^\star \overline{Q}^1 - \overline{\mathcal{T}}^\star \overline{Q}^2\|_\infty \leq \gamma(L+1)\|\overline{Q}^1 - \overline{Q}^2\|_\infty$$

with $\gamma(L+1) < 1$. As a conclusion, $\overline{\mathcal{T}}^\star$ is a $\gamma(L+1)$-contraction on $\mathcal{L}_\gamma$. □

B.1.6 Proof of Proposition 5.11

Definition B.2. *Let A be a set, and f a function defined on A. We define*

- *the convex hull of A:* $\mathcal{C}(A) = \{\sum_{i=1}^p \lambda_i a_i : a_i \in A, \lambda_i \in \mathbb{R}^+, \sum_{i=1}^p \lambda_i = 1, p \in \mathbb{N}\}$;
- *the convex edges of A:* $\mathcal{C}^2(A) = \{\lambda a_1 + (1-\lambda)a_2 : a_1, a_2 \in A, \lambda \in [0,1]\}$;
- *Dirac distributions of A:* $\delta(A) = \{\delta(a - a_0) : a_0 \in A\}$;

- *the image of A by f:* $f(A) = \{f(a) : a \in A\}$.

Proof. Let $\overline{s} = (s, \beta) \in \overline{\mathcal{S}}$ and $\overline{Q} \in (\mathbb{R}^2)^{\overline{\mathcal{S}\mathcal{A}}}$. We recall the definition of $\overline{\pi}_{\text{greedy}}$:

$$\overline{\pi}_{\text{greedy}}(\overline{a}|\overline{s}; \overline{Q}) \in \arg\min_{\rho \in \overline{\Pi}_r^{\overline{Q}}} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_c(\overline{s}, \overline{a}) \tag{5.13a}$$

$$\text{where} \quad \overline{\Pi}_r^{\overline{Q}} = \arg\max_{\rho \in \mathcal{M}(\overline{\mathcal{A}})} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_r(\overline{s}, \overline{a}) \tag{5.13b}$$

$$\text{s.t.} \mathop{\mathbb{E}}_{\overline{a}\sim\rho} Q_c(\overline{s}, \overline{a}) \leq \beta \tag{5.13c}$$

Note that any policy in the $\arg\min$ in (5.13a) is suitable to compute $\overline{\mathcal{T}}^\star$. We first reduce the set of candidate optimal policies. Consider the problem described in (5.13b),(5.13c): it can be seen as a single-step CMDP problem with reward $R = Q_r$ and cost $C = Q_c$. By (Theorem 4.4 Beutler and K. W. Ross, 1985), we know that the solutions are mixtures of two deterministic policies. Hence, we can replace $\mathcal{M}(\mathcal{A})$ by $\mathcal{C}^2(\delta(\overline{\mathcal{A}}))$ in (5.13b).

Moreover, remark that:

$$\begin{aligned}
&\{\mathop{\mathbb{E}}_{\overline{a}\sim\rho} \overline{Q}(\overline{s}, \overline{a}) : \rho \in \mathcal{C}^2(\delta(\overline{\mathcal{A}}))\} \\
&\qquad = \{\mathop{\mathbb{E}}_{\overline{a}\sim\rho} \overline{Q}(\overline{s}, \overline{a}) : \rho = (1-\lambda)\delta(\overline{a} - \overline{a}_1) + \lambda\delta(\overline{a} - \overline{a}_2), \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \lambda \in [0, 1]\} \\
&\qquad = \{(1-\lambda)\overline{Q}(\overline{s}, \overline{a}_1) + \lambda\overline{Q}(\overline{s}, \overline{a}_2), \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \lambda \in [0, 1]\} \\
&\qquad = \mathcal{C}^2(\overline{Q}(\overline{s}, \overline{\mathcal{A}}))\}.
\end{aligned}$$

Hence, the problem (5.13b), (5.13c) has become:

$$\tilde{\overline{\Pi}}^{Q_r} = \arg\max_{(q_r, q_c) \in \mathcal{C}^2(\overline{Q}(\overline{s}, \overline{\mathcal{A}}))} q_r \quad \text{s.t.} \quad q_c \leq \beta$$

and the solution of $\overline{\pi}_{\text{greedy}}$ is $q^\star = \arg\min_{q \in \tilde{\overline{\Pi}}^{Q_r}} q_c$.

The original problem in the space of actions $\overline{\mathcal{A}}$ is now expressed in the space of values $\overline{Q}(\overline{s}, \overline{\mathcal{A}})$ (which is why we use $=$ instead of $\in$ before $\arg\min$ here).

We further restrict the search space of $q^\star$ following two observations:

1. $q^\star$ belongs to the *undominated* points $\mathcal{C}^2(\overline{Q}^-)$:

$$\overline{Q}^+ = \{(q_c, q_r) : q_c > q_c^\pm = \min_{q^+} q_c^+ \text{ s.t. } q^+ \in \arg\max_{q \in \overline{Q}(\overline{s}, \overline{\mathcal{A}})} q_r\} \tag{B.10}$$

$$\overline{Q}^- = \overline{Q}(\overline{s}, \overline{\mathcal{A}}) \setminus \overline{Q}^+. \tag{B.11}$$

Denote $q^\star = (1-\lambda)q^1 + \lambda q^2$, with $q^1, q^2 \in \overline{Q}(\overline{s}, \overline{\mathcal{A}})$. There are three possible cases:

(a) $q^1, q^2 \notin \overline{Q}^-$. Then $q_c^\star = (1-\lambda)q_c^1 + \lambda q_c^2 > q_c^\pm$. But then $q_c^\pm < q_c^\star \leq \beta$ so $q^\pm \in \tilde{\overline{\Pi}}^{Q_r}$ with a strictly lower q_c than $q^\star$, which contradicts the $\arg\min$.

(b) $q^1 \in \overline{Q}^-, q^2 \notin \overline{Q}^-$. But then consider the mixture $q^\top = (1-\lambda)q^1 + \lambda q^\pm$. Since $q_r^\pm \geq q_r^2$ and $q_r^\pm < q_r^2$, we also have $q_r^\top \geq q_r^\star$ and $q_c^\top < q_c^\star$, which also contradicts the $\arg\min$.

(c) $q^1, q^2 \in \overline{Q}^-$ is the only remaining possibility.

2. $q^\star$ belongs to the *top frontier* $\mathcal{F}$:

$$\mathcal{F}_{\overline{Q}} = \{q \in \mathcal{C}^2(\overline{Q}^-) : \nexists q' \in \mathcal{C}^2(\overline{Q}^-) : q_c = q'_c \text{ and } q_r < q'_r\}. \tag{B.12}$$

Trivially, otherwise q' would be a better candidate than $q^\star$.

Let us characterise this frontier $\mathcal{F}$. It is both:

1. the *graph of a non-decreasing function*: $\forall q^1, q^2 \in \mathcal{F}$ such that $q_c^1 \leq q_c^2$ then $q_r^1 \leq q_r^2$.
By contradiction, if we had $q_r^1 > q_r^2$, we could define $q^\top = (1-\lambda)q^1 + \lambda q^\pm$ where $q^\pm$ is the dominant point as defined in (B.10). By choosing $\lambda = (q_c^2 - q_c^1)/(q_c^\pm - q_c^1)$ such that $q_c^\top = q_c^2$, then since $q_r^\pm \geq q_r^1 > q_r 2$ we also have $q_r^\top > q_r^2$ which contradicts $q^2 \in \mathcal{F}$.

2. the *graph of a concave function*: $\forall q^1, q^2, q^3 \in \mathcal{F}$ such that $q_c^1 \leq q_c^2 \leq q_c^3$ with λ such that $q_c^2 = (1-\lambda)q_c^1 + \lambda q_c^3$, then $q_r^2 \geq (1-\lambda)q_r^1 + \lambda q_r^3$.
Trivially, otherwise the point $q^\top = (1-\lambda)q^1 + \lambda q^3$ would verify $q_c^\top = q_c^2$ and $q_r^\top > q_r^2$, which would contradict $q^2 \in \mathcal{F}$.

We denote $\mathcal{F}_{\overline{Q}} = \mathcal{F} \cap \overline{Q}$. Clearly, $q^\star \in \mathcal{C}^2(\mathcal{F}_{\overline{Q}})$: let $q^1, q^2 \in \overline{Q}^-$ such that $q^\star = (1-\lambda)q^1 + \lambda q^2$. First, $q^1, q^2 \in \overline{Q}^- \subset \mathcal{C}^2(\overline{Q}^-)$. Then, by contradiction, if there existed $q^{1'}$ or $q^{2'}$ with equal q_c and strictly higher q_r, again we could build an admissible mixture $q^\top = (1-\lambda)q^{1'} + \lambda q^{2'}$ strictly better than $q^\star$.

$q^\star$ can be written as $q^\star = (1-\lambda)q^1 + \lambda q^2$ with $q^1, q^2 \in \mathcal{F}_{\overline{Q}}$ and, without loss of generality, $q_c^1 \leq q_c^2$.

Regular case There exists $q^0 \in \mathcal{F}_{\overline{Q}}$ such that $q_c^0 \geq \beta$. Then q^1 and q^2 must flank the budget: $q_c^1 \leq \beta \leq q_c^2$. Indeed, by contradiction, if $q_c^2 \geq q_c^1 > \beta$ then $q_c^\star > \beta$ which contradicts $\overline{\Pi}_r^{\overline{Q}}$. Conversely, if $q_c^1 \leq q_c^2 < \beta$ then $q^\star < \beta \leq q_c^0$, which would make $q^\star$ a worse candidate than $q^\top = (1-\lambda)q^\star + \lambda q^0$ when λ is chosen such that $q_c^\top = \beta$, and contradict $\overline{\Pi}_r^{\overline{Q}}$ again.

Because $\mathcal{F}$ is the graph of a non-decreasing function, λ should be as high as possible, as long as the budget $q^\star \leq \beta$ is respected. We reach the highest $q_r^\star$ when $q_c^\star = \beta$, that is: $\lambda = (\beta - q_c^1)/(q_c^2 - q_c^1)$.

It remains to show that q^1 and q^2 are two successive points in $\mathcal{F}_{\overline{Q}}$: $\nexists q \in \mathcal{F}_{\overline{Q}} \setminus \{q^1, q^2\} : q_c^1 \leq q_c \leq q_c^2$. Otherwise, as $\mathcal{F}$ is the graph of a concave function, we would have $q_r \geq (1-\mu)q_r^1 + \mu q_r^2$. q_r cannot be strictly greater than $(1-\mu)q_r^1 + \mu q_r^2$ which would contradict $q^\star$, but it can still be equal, which means the tree points q, q^1, q^2 are aligned. In fact, every points aligned with q^1 and q^2 can also be used to construct mixtures resulting in $q^\star$, but among these solutions we can still choose q^1 and q^2 as the two points in $\mathcal{F}_{\overline{Q}}$ closest to $q^\star$.

Edge case $\forall q \in \mathcal{F}_{\overline{Q}}, q_c < \beta$. Then $q^\star = \arg\max_{q \in \mathcal{F}} q_r = q^{\pm} = \arg\max_{q \in \overline{Q}^-} q_r$. □

Appendix C

Complements on Chapter 6

Outline We provide proofs for every claimed result in Section C.1. We look into the time and memory complexity and propose efficient implementations of `OLOP` and `KL-OLOP` in Section C.2.1, and of `GBOP-D` and `GBOP` in Section C.2.2.

C.1 Proofs

C.1.1 Proof of Lemma 6.5

Proof. The proof is identical to that of Lemma 4 in (Sébastien Bubeck and Rémi Munos, 2010).

Since $\arg\max_{a\in\mathcal{A}} T_a(M)$, and $\sum_{a\in\mathcal{A}} T_a(M) = M$, we have $T_{a(n)}(M) \geq M/K$, and thus:

$$\frac{M}{K}(V - V(a(n))) \leq (V - V(a(n)))T_{a(n)}(M) \leq \sum_{m=1}^{M} V - V\left(a^m\right)$$

Hence, we have, $r_n \leq \frac{K}{M}\sum_{m=1}^{M} V - V\left(a^m\right)$. Now remark that, for any sequence of actions $a \in \mathcal{A}^L$, we have either:

- $a_{1:H} \in \mathcal{I}_H$; which implies $V - V(a) \leq \frac{2\gamma^{H+1}}{1-\gamma}$
- or there exists $1 \leq h \leq H$ such that $a_{1:h} \in \mathcal{J}_h$; which implies $V - V(a) \leq V - V\left(a_{1:h-1}\right) + \frac{\gamma^h}{1-\gamma} \leq \frac{3\gamma^h}{1-\gamma}$.

Thus we can write:

$$\begin{aligned}\sum_{m=1}^{M}\left(V - V\left(a^m\right)\right) &= \sum_{m=1}^{M}\left(V - V\left(a^m\right)\right)\left(\mathbb{1}\left\{a^m \in \mathcal{I}_H\right\} + \mathbb{1}\left\{\exists 1 \leq h \leq H : a^m_{1:h} \in \mathcal{J}_h\right\}\right)\\ &\leq \frac{2\gamma^{H+1}}{1-\gamma}M + 3\sum_{h=1}^{H}\sum_{a\in\mathcal{J}_h}\frac{\gamma^h}{1-\gamma}T_a(M)\end{aligned}$$

□

C.1.2 Proof of Lemma 6.9

Proof. The event $\tau^a_{h,h'} = 1$ implies $a^{m+1} \in aA^*$ and (6.9). This implies by Lemma 6.6 that either (UCB violation), (LCB violation) or (Large CI) is satisfied. Now by Lemma 6.8 this implies that either (UCB violation) is true or (LCB violation) is true or (6.8) is false. We now prove that if (6.10) is not satisfied then (6.9) is true, which clearly ends the proof. This follows from: For any $0 \leq t \leq h'$:

$$\begin{aligned} T_{a_{1:t}}(m) &= \sum_{b \in a_{1:t}A^{h-t}} T_b(m) \geq \sum_{b \in \mathcal{P}^{a_{1:t}}_{h,h'}} T_b(m) \\ &\geq \left(\gamma^{2(t-h')}\right)\left(2f(m)(h+1)^2\gamma^{2(h'-h-1)}\right) \\ &= 2f(m)(h+1)^2\gamma^{2(t-h-1)}\,. \end{aligned}$$

□

C.1.3 Proof of Lemma 6.10

Proof. The proof is identical to that of Lemma 9 in (Sébastien Bubeck and Rémi Munos, 2010).

Let $h' \geq 1$ and $0 \leq s \leq h'$. We introduce the following random variables:

$$m^a_s = \min\left(M, \min\left\{m \geq 0 : \left|\mathcal{P}^a_{h,h'}(m)\right| \geq \gamma^{2(s-h')}\right\}\right).$$

We will prove recursively that,

$$\left|\mathcal{P}^{\emptyset}_{h,h'}(m)\right| \leq \sum_{t=0}^{s} \gamma^{2(t-h')}\left|\mathcal{I}_t\right| + \sum_{a \in \mathcal{I}_s}\left|\mathcal{P}^a_{h,h'} \setminus \cup_{t=0}^{s}\mathcal{P}^{a_{1:t}}_{h,h'}\left(m^{a_{1:t}}_t\right)\right| \tag{C.1}$$

The result is true for $s = 0$ since $\mathcal{I}_0 = \{\emptyset\}$ and by definition of $m^{\emptyset}_0$,

$$\left|\mathcal{P}^{\emptyset}_{h,h'}(m)\right| \leq \gamma^{-2h'} + \left|\mathcal{P}^{\emptyset}_{h,h'}(m) \setminus \mathcal{P}^{\emptyset}_{h,h'}\left(m^{\emptyset}_0\right)\right|$$

Now let us assume that the result is true for $s < h'$. We have:

$$\begin{aligned} \sum_{a \in \mathcal{I}_s}\left|\mathcal{P}^a_{h,h'}(m) \setminus \cup^{a_{1:t}}_{h,h'}\left(m^{a_{1:t}}_t\right)\right| &= \sum_{a \in \mathcal{I}_{s+1}}\left|\mathcal{P}^a_{h,h'}(m) \setminus \cup_{t=0}^{s}\mathcal{P}^{a_{1:t}}_{h,h'}\left(m^{a_{1:t}}_t\right)\right| \\ &\leq \sum_{a \in \mathcal{I}_{s+1}} \gamma^{2(s+1-h')} + \left|\mathcal{P}^a_{h,h'}(m) \setminus \cup_{t=0}^{s+1}\mathcal{P}^{a_{1:t}}_{h,h'}\left(m^{a_{1:t}}_t\right)\right| \end{aligned}$$

$$= \gamma^{2(s+1-h')} \left|\mathcal{I}_{s+1}\right| + \sum_{a \in \mathcal{I}_{s+1}} \left|\mathcal{P}^a_{h,h'}(m) \setminus \cup_{t=0}^{s+1} \mathcal{P}^{a_{1:t}}_{h,h'}\left(m_t^{a_{1:t}}\right)\right|$$

which ends the proof of (C.1). Thus we proved (by taking $s = h'$ and $m = M$):

$$\begin{aligned}\left|\mathcal{P}^{\emptyset}_{h,h'}(M)\right| &\leq \sum_{t=0}^{h'} \gamma^{2(t-h')} \left|\mathcal{I}_t\right| + \sum_{a \in \mathcal{I}_{h'}} \left|\mathcal{P}^a_{h,h'}(M) \setminus \cup_{t=0}^{s+1} \mathcal{P}^{a_{1:t}}_{h,h'}\left(m_t^{a_{1:t}}\right)\right. \\ &= \sum_{t=0}^{h'} \gamma^{2(t-h')} \left|\mathcal{I}_t\right| + \sum_{a \in \mathcal{J}_h} \left|\mathcal{P}^a_{h,h'}(M) \setminus \cup^{a_{1:t}}_{h,h'}\left(m_t^{a_{1:t}}\right)\right|\end{aligned}$$

Now, for any $a \in \mathcal{J}_h$, let $\tilde{m} = \max_{0 \leq t \leq h'} m_t^{a_{1:t}}$. Note that for $m \geq \tilde{m}$, equation (6.10) is not satisfied. Thus we have

$$\begin{aligned}\left|\mathcal{P}^a_{h,h'} \setminus \cup^{s+1}_{h,h'} \mathcal{P}^{a_{1:t}}_{h,h'}\left(m_t^{a_{1:t}}\right)\right| &= \sum_{m=\tilde{m}}^{M-1} \tau^a_{h,h'}(m+1) = \sum_{m=0}^{M-1} \tau^a_{h,h'}(m+1) \mathbb{1}\{\text{(6.10) is not satisfied }\} \\ &\leq \sum_{m=0}^{M-1} \tau^a_{h,h'}(m+1) \mathbb{1}\{\text{(UCB violation) or (LCB violation)}\}\end{aligned}$$

where the last inequality results from Lemma 6.9. Hence, we proved:

$$\left|\mathcal{P}^{\emptyset}_{h,h'}\right| \leq \sum_{t=0}^{h'} \gamma^{2(t-h')} \left|\mathcal{I}_t\right| + \sum_{m=0}^{M-1} \sum_{a \in \mathcal{J}_h} \mathbb{1}\{\text{(UCB violation) or (LCB violation)}\}$$

Taking the expectation and applying Lemma 6.7 yield the claimed bound for $h' \geq 1$.

Now for $h' = 0$ we need a modified version of Lemma 6.9. Indeed in this case one can directly prove that $\tau^a_{h,0}(m+1) = 1$ implies that either equation (UCB violation) or (LCB violation) is satisfied (this follows from the fact that $\tau^a_{h,0}(m+1) = 1$ always imply that (6.8) is true for $h' = 0$). Thus we obtain:

$$\left|\mathcal{P}^{\emptyset}_{h,h'}\right| = \sum_{m=0}^{M-1} \sum_{a \in \mathcal{J}_h} \tau^a_{h,0}(m+1) \leq \sum_{m=0}^{M-1} \sum_{a \in \mathcal{J}_h} \mathbb{1}\{\text{(UCB violation) or (LCB violation)}\}$$

Taking the expectation and applying Lemma 6.7 yield the claimed bound for $h' = 0$ and ends the proof.

□

C.1.4 Proof of Lemma 6.11

Proof. The proof is identical to that of Lemma 10 in (Sébastien Bubeck and Rémi Munos, 2010):

$$\begin{aligned}\sum_{a\in\mathcal{J}_h} T_a(M) &= \sum_{a\in\mathcal{J}_h\setminus\mathcal{P}^{\emptyset}_{h,h-1}} T_a(M) + \sum_{h'=1}^{h-1}\sum_{a\in\mathcal{P}^{\emptyset}_{h,h'}\setminus\mathcal{P}^{\emptyset}_{h,h'-1}} T_a(M) + \sum_{a\in\mathcal{P}^{\emptyset}_{h,0}} T_a(M)\\
&\leq 2f(m)(h+1)^2\gamma^{2(h-2-h)}\left|\mathcal{J}_h\right|\\
&\quad + \sum_{h'=1}^{h-1} 2f(m)(h+1)^2\gamma^{2(h'-2-h)}\log M\left|\mathcal{P}^{\emptyset}_{h,h'}\right| + M\left|\mathcal{P}^{\emptyset}_{h,0}\right|\\
&= \tilde{\mathcal{O}}\left(\left(\kappa'\right)^h + \gamma^{-2h}\sum_{h'=1}^{h-1}\gamma^{2h'}\left|\mathcal{P}^{\emptyset}_{h,h'}\right| + M\left|\mathcal{P}^{\emptyset}_{h,0}\right|\right)\end{aligned}$$

Taking the expectation and applying the bound of Lemma 6.10 give the claimed bound.

□

C.1.5 Proof of Lemma 6.15

Proof. The tightening property is directly obtained by definition of monotonicity. Let us show the preservation of monotonicity. Let U a monotonic upper-bound, $a \in \mathcal{A}^h$. Then, for any $b \in \mathcal{A}$:

$$U(ab) \geq B(U)(ab) \implies r(ab) + \gamma U(ab) \geq r(ab) + \gamma B(U)(ab).$$

Thus, my taking the $\max$ on b, $B(U)(a) \geq B^2(U)(a)$. The same can be obtained for a lower-bound L.

The finite time convergence can be obtained by recursion from the leaves to the root, by noticing that if the value of a set of siblings aA is invariant by B, then the value of their parent a is invariant by B^2. □

C.1.6 Proof of Lemma 6.16

Proof. The proof of tightening and monotonicity preservation is the same as that of Lemma 6.15. The contraction property is standard for the Bellman Operator, see *e.g.* Puterman M., Markov Decision Processes: Discrete Stochastic Dynamic Programming (2005). □

C.1.7 Proof of Lemma 6.18

Proof. By definition, for $a \in \mathcal{A}^h$,

$$\begin{aligned} V(a) &= \sup_{b \in a\mathcal{A}^\infty} \sum_{t=1}^{\infty} \gamma^t \mu(b_{1:t}) \\ &= \sum_{t=1}^{h} \gamma^t \mu(a_{1:t}) \sup_{b \in a\mathcal{A}^\infty} \sum_{t=h+1}^{\infty} \gamma^t \mu(b_{1:t}) \\ &= G(s_1, a)\gamma^h \sup_{b \in \mathcal{A}^\infty} \sum_{t=1}^{\infty} \gamma^t \mu(b_{1:t} \text{ starting from } s(a)) \\ &= G(s_1, a)\gamma^h V(s(a)) \end{aligned}$$

□

C.1.8 Proof of Theorem 6.20

We recall the main steps of the proof of Hren and Rémi Munos (2008).

1. The recommendation a_n has a maximal depth d_n in the tree, and its gap $r_n = V^\star - V(a_{n,1})$ is bounded by $r_n \leq \frac{\gamma^{d_n}}{1-\gamma}$. We need to relate d_n to n.
2. Each expanded node belongs to $\mathcal{T}^\infty = \bigcup_{h \geq 0} \mathcal{T}_h^\infty$, where

   $$\mathcal{T}_h^\infty = \left\{ a \in \mathcal{A}^h : V^\star - V(a) \leq \frac{\gamma^h}{1-\gamma} \right\}.$$

   Introduce the difficulty measure κ such that $|\mathcal{T}_h^\infty| = \mathcal{O}(\kappa^h)$ (the smallest).
3. In the worst case, expanded nodes fully fill the depths of $\mathcal{T}^\infty$ up to d_n: $n = \sum_{d=1}^{d_n} n_d \leq C \sum_{d=1}^{d_n} \kappa^d = \begin{cases} \mathcal{O}(d_n) & \text{if } \kappa = 1 \\ \mathcal{O}(\kappa^{d_n}) & \text{else.} \end{cases}$

   Hence $r_n = \begin{cases} \mathcal{O}(\gamma^n) & \text{if } \kappa = 1 \\ \mathcal{O}(\gamma^{\frac{\log n}{\log \kappa}}) = \mathcal{O}(n^{-\frac{\log 1/\gamma}{\log \kappa}}) & \text{else.} \end{cases}$

C.1.9 Proof of Lemma 6.22

Proof. Let $L_2 \leq L_1 \leq V \leq U_1 \leq U_2$, then $\mathcal{T}_h^\infty(L_1, U_1) \subset \mathcal{T}_h^\infty(L_2, U_2)$, which implies

$$|\mathcal{T}_h^\infty(L_1, U_1)|^{1/h} \leq |\mathcal{T}_h^\infty(L_2, U_2)|^{1/h}$$

and the claimed result in the limit $h \to \infty$. □

C.1.10 Proof of Theorem 6.23

In this proof, we temporarily assume that $U = B(U)$ and $L = B(L)$. We follow the same steps as in the proof of the regret of OPD.

Remark C.1. *It no longer holds that a_n must be of maximal depth d_n. This is due to the fact the exploration bonus $\gamma^h U(a)$ is not depth-wise constant: consider two nodes a, b at the same depth with $R(a) > R(b)$. In OPD, both get the same bonus $\gamma^h/(1-\gamma)$, and the node a is expanded first. But with the local bonus, b could be expanded in priority rather than a, if its own bonus is sufficiently higher than that of a, precisely if $R(a) + \gamma^h U(a) < R(b) + \gamma^h U(b)$. For instance, $U(a) = 0$ when a is known to be a terminal state while b can lead to future rewards. If after expanding and exploring the subtree of b we find out that $V(b) = 0$, we still return the recommendation a, which is of non-maximal depth.*

The regret bound still holds, however. First, notice that:

Lemma C.2 (Expansion)**.** *Whenever a node a of depth h is expanded by the optimistic algorithm, its first action a_1 enjoys a simple regret $V(a^\star) - V(a_1) \leq \gamma^h(U(a) - L(a))$.*

Proof. Let t be the time of expansion of a, it holds that $\overline{U}_t(b) \leq \overline{U}_t(a)$ for all $b \in \partial\mathcal{T}_t$, in particular those in a branch starting by an optimal action $a^\star$. Since $U = B(U)$ and $L = B(L)$, we also have $\overline{U}_t(a^\star) = \max_{b \in a^\star A^*} \overline{U}_t(b) \leq \overline{U}_t(a)$, and $\overline{L}_t(a_1) = \max b \in a_1 A^* \overline{L}_t(b) \geq \overline{L}_t(a)$. Thus, $V(a^\star) - V(a_1) \leq \overline{U}_t(a^\star) - \overline{L}_t(a_1) \leq \overline{U}_t(a) - \overline{L}_t(a) = \gamma^h(U(a) - L(a))$. □

Lemma C.3 (Recommendation)**.** *The recommended action a_n has a simple regret $r_n \leq \frac{\gamma^{d_n}}{1-\gamma}$, where d_n is the maximal depth of $\mathcal{T}_n$.*

Proof. Let i a node of maximal depth d_n, and consider the recommended node a_n at time n, of depth d. In particular, $\overline{L}_n(a_n) \geq \overline{L}_n(i)$, and since $(\overline{L}_t)_t$ is non-decreasing we also have $\overline{L}_n(i) \geq \overline{L}_t(i)$. At the time t when i is expanded, we have $\overline{U}_t(a_n) \leq \overline{U}_t(i)$, and since $(\overline{U}_t)_t$ is non-increasing we also have $\overline{U}_n(a_n) \leq \overline{U}_t(a_n)$. We can conclude with Lemma C.2 applied to a_n: $r_n \leq \gamma^d(U(a_n) - L(a_n) = \overline{U}_n(a_n) - \overline{L}_n(a_n) \leq \overline{U}_t(a_n) - \overline{L}_n(i) \leq \overline{U}_t(i) - \overline{L}_t(i) = \gamma^{d_n}(U(i) - L(i)$, which yields the claimed bound since $U(i) - L(i) \leq V_{\max} - 0$. □

Lemma C.4 (Near-optimal nodes)**.** *Every node expanded by* (6.16) *is in*

$$\mathcal{T}^{\infty}(L,U) = \bigcup_{h\geq 0} \mathcal{T}_h^{\infty}(L,U).$$

Proof. Let a be a node of depth h expanded at round n, then $\overline{U}_n(a) \geq \overline{U}_n(b)$ for all $b \in \partial\mathcal{T}_n$. Thus, since $U = B(U)$, we have $\overline{U}(a) = \overline{B(U)}(\emptyset) = B(U)(s_1) \geq V(s_1) = V^\star$. Thus, $V^\star - V(a) \leq \overline{U}(a) - \overline{L}(a) = \gamma^h(U(a) - L(a))$. □

Finally, we can move on to the proof of Theorem 6.23. Let n_d be the number of expanded nodes of depth d, by Lemma C.4 we have $n_d \leq |\mathcal{T}_d^{\infty}(L,U)| \leq C\kappa(L,U)^d$. Thus,

$$n = \sum_{d=1}^{d_n} n_d \leq C \sum_{d=0}^{d_n} \kappa(L,U)^d = C\frac{\kappa(L,U)^{d_n+1} - 1}{\kappa(L,U) - 1}$$

Hence, $d_n \geq C' \frac{\log n}{\log \kappa(L,U)}$, which along with Lemma C.2 gives the claimed bound.

Note that if L, U are monotonic bounds that do not verify $L = B(L)$ and $U = B(U)$, then planning with $B(L), B(U)$ instead will yield the proved bound with a branching factor $\kappa(B(L), B(U))$, and since $L \leq B(L) \leq V \leq B(U) \leq U$ we have $\kappa(B(L), B(U)) \leq \kappa(L,U)$, which still gives

$$r_n = \mathcal{O}\left(n^{-\frac{\log 1/\gamma}{\log \kappa(L,U)}}\right);$$

C.1.11 Proof of Lemma 6.24

Proof. We first show that if U is equivalent to $\mathcal{U}$, meaning that for any sequence $a \in T(\mathcal{G}_n)$ we have $U(a) = \mathcal{U}(s(a))$, then $B_n(U)$ is equivalent to $\mathcal{B}_n(\mathcal{U})$.

By definition of $T(\mathcal{G}_n)$, any sequence of action $a \in T(\mathcal{G}_n)$ corresponds to a path $s_1, a_1, \ldots, s_h, a_h, s_{h+1}$ in $\mathcal{G}_n$. If $a \in \partial T(\mathcal{G}_n)$, then necessarily $s(a) \in \partial\mathcal{G}_n$, and both are unchanged by B_n and $\mathcal{B}_n$ respectively. Conversely, if $a \in \mathring{T}(\mathcal{G}_n)$, then $s(a) \in \mathring{\mathcal{G}}_n$ by construction. Thus,

$$\begin{aligned}
B_n(U)(a) &= \max_{b\in\mathcal{A}} r(s(a),b) + \gamma U(ab) \\
&= \max_{b\in\mathcal{A}} r(s(a),b) + \gamma \mathcal{U}(s(ab)) && \text{(by assumption)} \\
&= \max_{b\in\mathcal{A}} r(s(a),b) + \gamma \mathcal{U}(P(s(a),b))
\end{aligned}$$

$$= \mathcal{B}_n(\mathcal{U}_n)(s(a)).$$

By induction, for any $k > 0$ $B_n^k(U)$ is equivalent to $\mathcal{B}_n^k(\mathcal{U})$, and at the limit $k \to \infty$ it comes that U_n is equivalent to $\mathcal{U}_n$. The same result can be shown similarly for L_n and $\mathcal{L}_n$. □

C.1.12 Proof of Lemma 6.25

We start by showing a preliminary lemma.

Lemma C.5 (Bounds of sequence values)**.** *The bounds $(\overline{L}_n, \overline{U}_n)$ on the value of sequences of actions verify are respectively non-decreasing and non-increasing with respect to n, and verify: for all $a \in \mathcal{A}^*$, $\overline{U_n}(a) = \max_{a' \in aA^\infty} \overline{U}(a')$.*

Proof. The second property can be easily shown by induction using the fact that U_n and L_n are fixed-points of B_n by definition. Applying this equation at each depth h gives the result. From this observation, we can deduce that $\overline{L}_n$ is increasing with n. Indeed, since when $T(\mathcal{G}_n)$ is expanded with additional nodes compared to $T(\mathcal{G}_{n-1})$, the leaves a of $T(\mathcal{G}_{n-1})$ with previous value $L_{n-1}(a) = 0$ are updated to $L_n(a) = \max_b r(s(a), b) \geq 0 = L_{n-1}(a)$, and this increase at the leaves is then propagated through $\max_{a' \in aA^\infty}$ to any internal node a. Thus, L_n is non-decreasing and likewise, U_n is non-increasing with respect to n. The same is obtained directly of the bounds on sequence values $(\overline{L}_n, \overline{U}_n)$. □

Which enables us to proceed to the proof of Lemma 6.25.

Proof. Let t be the time of expansion of a, it holds that $\overline{U}_t(b) \leq \overline{U}_t(a)$ for all $b \in T(\mathcal{G}_n)$. In particular for b in a branch starting by an optimal action $a^\star$ $\overline{U}_t(a) \geq \max_{b \in a^\star A^*} \overline{U}_t(b) = \overline{U}_t(a^\star)$. Thus, $V(a^\star) - V(a) \leq \overline{U}_t(a^\star) - \overline{L}_t(a) \leq \overline{U}_t(a) - \overline{L}_t(a) = \gamma^h(U_t(a) - L_t(a))$. □

C.1.13 Proof of Lemma 6.26

Proof. Let i an expanded node of maximal depth $d_n \in \mathbb{R} \cup \{\infty\}$, and consider the recommended node a_n at time n, of depth $d \in \mathbb{R} \cup \{\infty\}$. In particular, $\overline{L}_n(a_n) \geq \overline{L}_n(i)$, and since $(\overline{L}_t)_t$ is non-decreasing we also have $\overline{L}_n(i) \geq \overline{L}_t(i)$. At the time t when i is expanded, we have $\overline{U}_t(a_n) \leq \overline{U}_t(i)$, and since $(\overline{U}_t)_t$ is non-increasing we also have $\overline{U}_n(a_n) \leq \overline{U}_t(a_n)$. We can conclude with Lemma 6.25 applied to a_n: $r_n \leq V^\star - V(a_n) \leq \gamma^d(U(a_n) - L(a_n) = \overline{U}_n(a_n) - \overline{L}_n(a_n) \leq \overline{U}_t(a_n) - \overline{L}_n(i) \leq \overline{U}_t(i) - \overline{L}_t(i) = \gamma^{d_n}(U_t(i) - L_t(i)$, which yields the claimed bound since $U(i) - L(i) \leq V_{\max} - 0$. □

C.1.14 Proof of Theorem 6.27

Proof. Let $\kappa' > \kappa_\infty$. Since $\kappa(L_n, U_n) \to \kappa_\infty$, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $\kappa(L_n, U_n) \leq \kappa'$. By Lemma 6.25, at each iteration n the expanded node must belong to $\mathcal{T}^\infty(L_n, U_n)$. Let $n \geq n_0$, and define $d_0 = \min\{d \in \mathbb{N} : \exists t \in [n_0, n], b_t \in \mathcal{A}^d\}$. By definition, for all $d \geq d_0$, any expanded node of depth d was expanded at a time $t \geq n_0$, and thus $b_t \in \mathcal{T}_t^\infty \subset \mathcal{T}_{n_0}^\infty$. We denote n_d the number of expanded nodes of depth d. If $d_n = \infty$, then $r_n = 0$ and the bound holds. Else, we obtain

$$n = \sum_{d=0}^{d_0-1} n_d + \sum_{d=d_0}^{d_n} n_d \leq C_0 + C_1 \sum_{d=d_0}^{d_n} (\kappa')^d \leq C_0 + C_1'(\kappa')^{d_n}$$

And since $r_n \leq \frac{\gamma^{d_n}}{1-\gamma}$ by Lemma 6.26, we obtain the claimed bound.

Moreover, given a history of observed transitions up to iteration n, the bounds U_n, L_n obtained from (6.20) on the unrolled tree $T(\mathcal{G}_n)$ are tighter than those of (6.14) since $T_n \subset T(\mathcal{G}_n)$, which implies by Lemma 6.22 that $\kappa(L_n, U_n) \leq \kappa$. We obtain $\kappa_\infty \leq \kappa$ at the limit. □

C.1.15 Proof of Proposition 6.28

The Figure C.1 shows the planning tree corresponding to the MDP $\mathcal{M}$. Whenever the action a_1 is taken (in green) the resulting subtree is represented by a leaf node $s^\star$ of value $V^\star = \frac{r^\star}{1-\gamma}$. When, in contrast, we take a sequence of actions among $a_2 \dots a_K$ (in orange), we stay in the state s^+ and denote V_h the corresponding value at depth h.

Lemma C.6. *Any sequence of actions in $A \setminus a_1$ is in $\mathcal{T}^\infty$.*

Proof. Any such sequence of actions yields the sequence of rewards $r^-, r^+, \dots, r^+$. and end up in the state s^+ with value at least $V^\star$ (obtained by further taking a_1 indefinitely). Thus its value V_h verifies,

$$\begin{aligned}
V_h &\geq \sum_{t=0}^{h-1} \gamma^t r_t + \gamma^h V^\star \\
&= r^- - r^+ + \sum_{t=0}^{h-1} \gamma^t r^+ + \gamma^h V^\star \\
&= (-\frac{\gamma}{1-\gamma} - 1)S + \frac{1-\gamma^h}{1-\gamma}(r^\star + S) + \gamma^h V^\star \\
&= V^\star - S\frac{\gamma^h}{1-\gamma} \geq V^\star - \frac{\gamma^h}{1-\gamma}
\end{aligned}$$

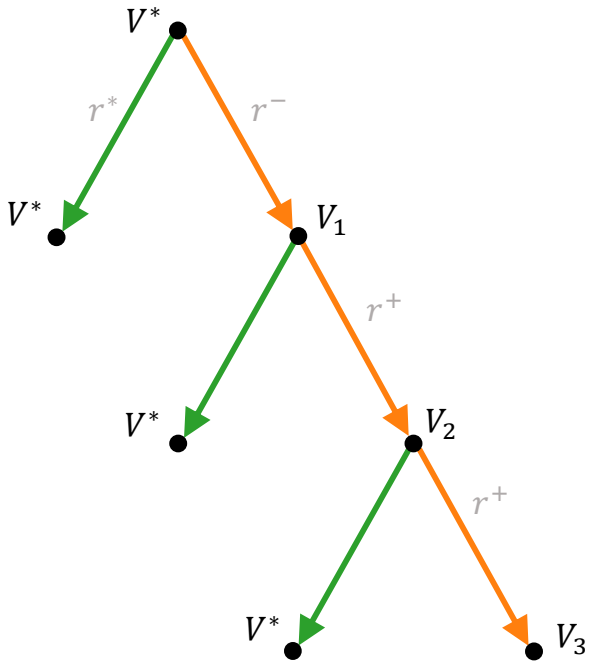


Figure C.1 – Planning tree of the MDP $\mathcal{M}$ of Figure 6.8

.

□

We can directly conclude that $\kappa \geq \limsup |\{a_2, \dots, a_K\}^h|^{1/h} = K - 1$.

Now, consider the nodes expanded by GBOP-D. The first expansion is that of the root, which discovers $s^\star$ and s^+. In the absence of information on these two state, the bound $V_{\max}$ is used and the first action a_1 gets a higher $\overline{U}$ that any other action $a_2, \dots, a_K$ since $r^\star \geq r^-$. Hence, at the second iteration, the node a_1 gets expanded. At this point, the self-loop of the state $s^\star$ is discovered, which means that form now on the bounds verify $L_n(a_1) = V^\star = U_n(a_1)$ for $n \geq 2$, which means that $L_n(a_1\mathcal{A}^*) - U_n(a_1\mathcal{A}^*) = 0$. The nodes $a_2, \dots, a_K$ can be expanded at most once before the entire MDP is discovered and $L_n = V = U_n$ over the entire tree, which means that $\mathcal{T}_n^\infty$ is the set of optimal nodes, *i.e.* the nodes in the only optimal sequence $a_1^\star$. Hence, $\kappa_\infty = 1$.

C.2 Time and memory complexities

C.2.1 KL-OLOP

After having considered the sample efficiency of OLOP and KL-OLOP in Theorem 6.4, we now study their time and memory complexities. We will only mention the case of KL-OLOP for ease of presentation, but all results easily extend to OLOP.

The Algorithm 6.1 requires, at each episode, to compute and store in memory of the reward upper-bounds and U-values of all nodes in the tree $\mathcal{T} = \sum_{h=0}^{L} \mathcal{A}^h$. Hence, its time and memory complexities $C(\texttt{KL-OLOP})$ are

$$C(\texttt{KL-OLOP}) = \mathcal{O}(M|\mathcal{T}|) = \mathcal{O}(MK^L).$$

The curse of dimensionality brought by the branching factor K and horizon L makes it intractable in practice to actually run KL-OLOP in its original form even for small problems. However, most of this computation and memory usage is wasted, as with reasonable sample budgets n the vast majority of the tree $\mathcal{T}$ will not be actually explored and hence does not hold any valuable information.

We propose in Algorithm C.1 a lazy version of KL-OLOP which only stores and processes the explored subtree, as shown in Figure C.2, while preserving the inner workings of the original algorithm.

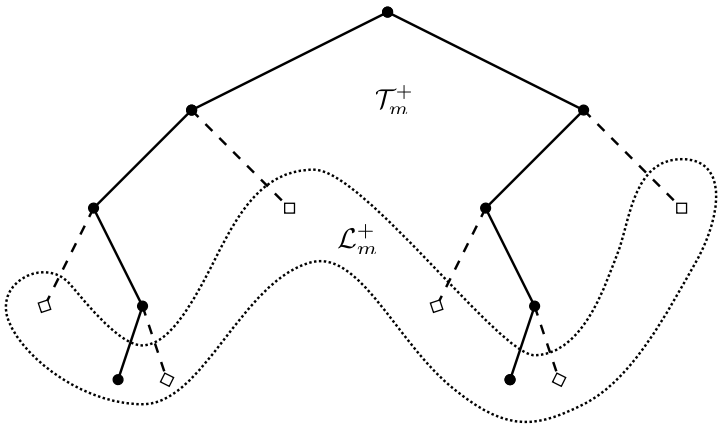


Figure C.2 – A representation of the tree $\mathcal{T}_m^+$, with $K = 2$ actions and after episode $m = 2$, when two sequences have been sampled. They are represented with solid lines and dots •, and they constitute the explored subtree $\mathcal{T}_m$. When extending $\mathcal{T}_m$ with the missing children of each node, represented with dashed lines and diamonds ⋄, we obtain the full extended subtree $\mathcal{T}_m^+$. The set of its leaves is denoted $\mathcal{L}_m^+$ and shown as a dotted set.

Algorithm C.1: Lazy Open Loop Optimistic Planning

Let M be the largest integer such that $M \log M/(2 \log 1/\gamma) \leq n$
Let $L = \log M/(2 \log 1/\gamma)$
Let $\mathcal{T}_0^+ = \mathcal{L}_0^+ = \{\emptyset\}$
for *each episode* $m = 1, \cdots, M$ **do**
  Compute $U_a(m-1)$ from (6.4) for all $a \in \mathcal{T}_{m-1}^+$
  Compute $B_a(m-1)$ from (6.5) for all $a \in \mathcal{L}_{m-1}^+$
  Sample a sequence with highest B-value: $a \in \arg\max_{a \in \mathcal{L}_{m-1}^+} B_a(m-1)$
  Choose an arbitrary continuation $a^m \in \mathcal{A}\mathcal{A}^{L-|a|}$ `// e.g. uniformly`
  Let $\mathcal{T}_m^+ = \mathcal{T}_{m-1}^+$ and $\mathcal{L}_m^+ = \mathcal{L}_{m-1}^+$
  for $t = 1, \cdots, L$ **do**
    if $a_{1:t}^m \notin \mathcal{T}_m^+$ **then**
      Add $a_{1:t-1}^m A$ to $\mathcal{T}_m^+$ and $\mathcal{L}_m^+$
      Remove $a_{1:t-1}^m$ from $\mathcal{L}_m^+$
return the most played sequence $a(n) \in \arg\max_{a \in \mathcal{L}_m^+} N_a(m)$

Proposition C.7 (Time and memory complexity)**.** *Algorithm C.1 has time and memory complexities of*

$$C(\texttt{Lazy KL-OLOP}) = O(KLM^2)$$

The corresponding complexity gain compared to the original Algorithm 6.1 is:

$$\frac{C(\texttt{Lazy KL-OLOP})}{C(\texttt{KL-OLOP})} = \frac{n}{K^{L-1}}$$

which highlights that only a subtree corresponding to the sample budget n is processed instead of the search whole tree $\mathcal{T}$.

Proof. At episode $m = 1, \cdots, M$, we compute and store in memory the reward upper-bounds and U-values of all nodes in the subtree $\mathcal{T}_m^+$. Moreover, the tree $\mathcal{T}_m^+$ is constructed iteratively by adding K nodes at most L times at each episode from 0 to m. Hence, $|\mathcal{T}_m^+| = O(mKL)$. This yields directly $C(\texttt{Lazy KL-OLOP}) = \sum_{m=1}^{M} O(mKL) = O(M^2KL)$. □

Proposition C.8 (Consistency)**.** *The set of sequences returned by Algorithm C.1 is the same as the one returned by Algorithm 6.1. In particular, Algorithm C.1 enjoys the same regret bounds as in Theorem 6.4.*

Proof. To prove consistency of Algorithm C.1, we need to show that the sequences of actions a^m sampled at every episode are chosen arbitrarily from the same sets as in Algorithm C.1. Namely,

$$\left\{ b \in \mathcal{A}\mathcal{A}^{L-|a|} : a \in \underset{a \in \mathcal{L}^+_{m-1}}{\arg\max}\, B_a(m-1) \right\} = \underset{a \in \mathcal{A}^L}{\arg\max}\, B_a(m-1)$$

To that end, we first introduce some useful notations:

Let $\mathcal{T}_m$ be the set of visited nodes after episode m:

$$\mathcal{T}_m \triangleq \{a \in \mathcal{A}^* : N_a(m) > 0\}$$

We also define its extension $\mathcal{T}^+_m$ of visited nodes and their children:

$$\mathcal{T}^+_m \triangleq \mathcal{T}_m + \mathcal{T}_m A$$

Now for $a \in \mathcal{A}^*$, $\pi_m(a)$ (resp. $\pi^+_m(a)$) refers to its longest prefix within $\mathcal{T}_m$ (resp. $\mathcal{T}^+_m$):

$$\pi_m(a) \triangleq \underset{b \in \mathcal{T}_m}{\arg\max}\{|b| : a \in b\mathcal{A}^*\}$$
$$\pi^+_m(a) \triangleq \underset{b \in \mathcal{T}^+_m}{\arg\max}\{|b| : a \in b\mathcal{A}^*\}$$

Finally, $\mathcal{L}_m$ and $\mathcal{L}^+_m$ are the image of $\mathcal{A}^L$ by π_m and π^+_m, respectively.

$$\mathcal{L}_m \triangleq \pi_m(\mathcal{A}^L)$$
$$\mathcal{L}^+_m \triangleq \pi^+_m(\mathcal{A}^L)\}$$

Remark C.9 (About children extensions)**.** *We could frame Algorithm C.1 in terms of $\mathcal{T}_m$ and $\mathcal{L}_m$, for which mathematical proofs are more straight-forward. However, the iterative construction of $\mathcal{L}_m$ is tricky and it would require inverting π_m on $\mathcal{L}_m$ which is non-trivial. On the contrary, introducing their extensions $\mathcal{T}^+_m$ and $\mathcal{L}^+_m$ slightly complicates the proof, but greatly simplifies the construction of $\mathcal{L}^+_m$ and the computation of ${\pi^+_m}^{-1}$ on $\mathcal{L}^+_m$, which is why we use these sets in practice.*

Lemma C.10 (Sets construction). *$\mathcal{T}_m^+$ and $\mathcal{L}_m^+$ are indeed the sets computed in Algorithm C.1.*

Proof. Note that for each episode $1 \leq m \leq M-1$, we have:

$$\mathcal{T}_{m+1} = \mathcal{T}_m + \sum_{t=0}^{L} a_{1:t}^{m+1} \tag{C.2}$$

Indeed, the nodes visited at least once at time $m+1$ where either already visited once at time m (*e.g.* in $\mathcal{T}_m$) or have been visited for the first time during episode $m+1$, which means they are a prefix of a^{m+1}. The reverse is clearly true as well.

This enables to write:

$$\begin{aligned}
\mathcal{T}_{m+1}^+ &= \mathcal{T}_{m+1} + \mathcal{T}_{m+1}A && \text{by definition} \\
&= \mathcal{T}_m + \sum_{t=0}^{L} a_{1:t}^{m+1} + (\mathcal{T}_m + \sum_{t=0}^{L} a_{1:t}^{m+1})A && \text{by (C.2)} \\
&= (\mathcal{T}_m + \mathcal{T}_m A) + \sum_{t=0}^{L} a_{1:t}^{m+1} + \sum_{t=0}^{L} a_{1:t}^{m+1}A \\
&= \mathcal{T}_m^+ + a_{1:0}^{m+1} + \sum_{t=0}^{L} a_{1:t}^{m+1}A && \text{as } \sum_{t=1}^{L} a_{1:t}^{m+1} \subset \sum_{t=0}^{L} a_{1:t}^{m+1}A \\
&= \mathcal{T}_m^+ + \sum_{t=0}^{L} a_{1:t}^{m+1}A && \text{as } a_{1:0}^{m+1} = \emptyset \in \mathcal{T}_0 \subset \mathcal{T}_m \subset \mathcal{T}_m^+
\end{aligned}$$

This recursion is the one implemented in Algorithm C.1: at each episode m, we add to $\mathcal{T}_m^+$ the children of the nodes along the sampled action sequence a^m.

Finally, we highlight that $\mathcal{L}_m^+ = \pi^+(\mathcal{A}^L)$ is the set of leaves of $\mathcal{T}_m^+$. Indeed, nodes of $\mathcal{L}_m^+$ belong to $\mathcal{T}_m^+$, but they cannot have a child in $\mathcal{T}_m^+$ as it would contradict the definition of $\mathcal{L}_m^+$. Conversely, any leaf a of $\mathcal{T}_m^+$ can be continued arbitrarily to a sequence b of $\mathcal{A}^L$, which $a = \pi_m^+(b) \in \pi^+(\mathcal{A}^L) = \mathcal{L}_m^+$.

Thus, when updating $\mathcal{T}_{m-1}^+$, the set of its leaves is updated accordingly: when the children of a leaf $a_{1:t-1}^m$ are added to $\mathcal{T}_m^+$, they become new leaves in place of their parent. Hence, they are added to $\mathcal{L}_m^+$ while $a_{1:t-1}^m$ is removed from it. □

Lemma C.11 (U-values conservation). *For all $a \in \mathcal{A}^*$,*

$$U_a(m) = U_{\pi_m(a)}(m) = U_{\pi_m^+(a)}(m)$$

Proof. Let $a \in \mathcal{A}^*$, denote $h = |a|$ and $h' = |\pi_m(a)|$.

By definition of $\pi_m(a)$, $0 \leq h' \leq h$, and

- for $1 \leq t \leq h'$, we have $a_{1:t} = \pi_m(a)_{1:t}$;
- for $h'+1 \leq t \leq h$, we have $a_{1:t} \notin \mathcal{T}_m$, hence $T_{a_{1:t}}(m) = 0$ and $U^\mu_{a_{1:t}}(m) = 1$.

Then,

$$\begin{aligned}
U_a(m) &= \sum_{t=1}^{h} \gamma^t U^\mu_{a_{1:t}}(m) + \frac{\gamma^{h+1}}{1-\gamma} \\
&= \sum_{t=1}^{h'} \gamma^t U^\mu_{a_{1:t}}(m) + \sum_{t=h'+1}^{h} \gamma^t \underbrace{U^\mu_{a_{1:t}}(m)}_{1} + \frac{\gamma^{h+1}}{1-\gamma} \\
&= \sum_{t=1}^{h'} \gamma^t U^\mu_{\pi_m(a)_{1:t}}(m) + \frac{\gamma^{h'+1}}{1-\gamma} \\
&= U_{\pi_m(a)}(m)
\end{aligned}$$

Now, consider $\pi_m^+(a) \in \mathcal{T}_m^+$. By definition, it belongs either to $\mathcal{T}_m$ or $\mathcal{T}_m A$.

- If $\pi_m^+(a) \in \mathcal{T}_m$, then $\pi_m^+(a) = \pi_m(a)$ and $U_{\pi_m^+(a)}(m) = U_{\pi_m(a)}(m)$.
- Else, $\pi_m^+(a) \in \mathcal{T}_m A$ and $p(\pi_m^+(a)) = \pi_m(a)$.

  As $\pi_m^+(a) \notin \mathcal{T}_m$, we have $T_{\pi_m^+(a)}(m) = 0$ and $U^\mu_{\pi_m^+(a)}(m) = 1$. This yields:

$$U_{\pi_m^+(a)}(m) = \sum_{t=1}^{h'} \gamma^t U^\mu_{\pi_m^+(a)_{1:t}}(m) + \gamma^{h'+1} \underbrace{U^\mu_{\pi_m^+(a)}(m)}_{1} + \frac{\gamma^{h'+2}}{1-\gamma} = U_{\pi_m(a)}(m)$$

We showed that $U_{\pi_m^+(a)}(m) = U_{\pi_m(a)}(m)$, which concludes the proof. □

Lemma C.12 (Inverse projection)**.** *For all $a \in \mathcal{L}_m^+$ of length $h \leq L$,*

$$\pi_m^{+-1}(a) = a\mathcal{A}^{L-h}$$

This allows to easily pick a sequence inside $\pi_m^{+-1}(a)$: just continue the sequence a with a default action of A (e.g. *the first*) *until it reaches length L.*

Proof. Let $a \in \mathcal{L}_m^+$.

By definition of π_m^+, any sequence in $\pi_m^{+-1}(a)$ is a suffix of a of length L, so we clearly have the direct inclusion $\pi_m^{+-1}(a) \subset a\mathcal{A}^{L-h}$.

Now for the other side: let $b \in \mathcal{A}\mathcal{A}^{L-h}$, *i.e.* $a = b_{1:h}$. We need to show that $\pi_m^+(b) = a$. As $a \in \mathcal{L}_m^+$, there exists $c \in \mathcal{A}^L$ such that $\pi_m^+(c) = a$.

- If h = L, then $b = a$, so $b \in \mathcal{L}_m^+ \subset \mathcal{T}_m^+$, and hence $\pi_m^+(b) = b = a$.
- If h < L, we can show by contradiction that $a \notin \mathcal{T}_m$. Indeed, if $a \in \mathcal{T}_m$, then $c_{1:h+1}$ is the child of a node of $\mathcal{T}_m$ and hence belongs to $\mathcal{T}_m^+$. But then, $c_{1:h+1}$ is a prefix of c in $\mathcal{T}_m^+$ with greater length than a, which contradicts the definition of $a = \pi_m^+(c)$.

  Now, because $a \notin \mathcal{T}_m$, it is also true for all suffixes of a, and in particular for $b_{1:t}$ with $h \leq t \leq L$. Indeed, we have $a_{1:t}^s = b_{1:t} \implies a_{1:h}^s = b_{1:h} = a$, so:

  $$T_{b_{1:t}}(m) = \sum_{s=1}^{m} \mathbb{1}\{a_{1:t}^s = b_{1:t}\} \leq \sum_{s=1}^{m} \mathbb{1}\{a_{1:h}^s = a\} = N_a(m) = 0$$

  Hence, $b_{1:t} \notin \mathcal{T}_m$ for all $h \leq t \leq L$, so in particular $b_{1:t} \notin \mathcal{T}_m^+$ for all $h+1 \leq t \leq L$. Since $b_{1:h} = a \in \mathcal{T}_m^+$, a is indeed the longest prefix of b in $\mathcal{T}_m^+$, that is: $\pi_m^+(b) = a$.

We have shown the other side of the inclusion: $a\mathcal{A}^{L-h} \subset \pi_m^{+-1}(a)$, which entails that the two sets are in fact equal. □

We can now conclude our proof of Proposition C.8: at episode m, KL-OLOP samples a sequence of action a^m within the set $\arg\max_{a \in \mathcal{A}^L} U_a(m)$. However, we have:

$$\arg\max_{c \in \mathcal{A}^L} U_c(m) = \arg\max_{c \in \mathcal{A}^L} U_{\pi_m^+(c)}(m) \qquad \text{by Lemma C.11}$$

$$
\begin{aligned}
&= \pi_m^{+-1}\left(\underset{a\in\pi_m^+(\mathcal{A}^L)}{\arg\max}\, U_a(m)\right)\\
&= \left\{b \in \pi_m^{+-1}(a) : a \in \underset{a\in\mathcal{L}_m^+}{\arg\max}\, U_a(m)\right\}\\
&= \left\{b \in \mathcal{A}\mathcal{A}^{L-|a|} : a \in \underset{a\in\mathcal{L}_m^+}{\arg\max}\, U_a(m)\right\} \qquad \text{by Lemma C.12}
\end{aligned}
$$

Thus, at each episode the sequence of actions a^m sampled by Algorithm C.1 could have been sampled by Algorithm 6.1 as well.

In particular, if the arbitrary rule used to pick a sequence from a set is the same for the two algorithms, then the sampled sequences a^m will be identical, will have the same visit count $T_{a^m}(m)$, and in the end the returned action $a(n)$ will be the same. □

C.2.2 GBOP

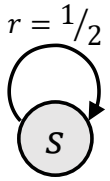
In this section, we provide more details about the implementation of GBOP-D and GBOP. First, we discuss how two procedures can be approximated so that they terminate in finite time, and study the impact of this approximation on the regret guarantees. Second, we propose a lazy implementation of the bounds computation through $\mathcal{B}_n^\infty$ that only considers a subset of nodes to update.

Termination

Bounds computation The bounds computation step $\mathcal{B}_n^\infty$ (line 1 of GBOP-D) can converge in infinite time whenever $\mathcal{G}_n$ contains a loop, as shown in Figure C.3. We consider the effect of stopping early after a fixed number of iterations $k(\varepsilon, \gamma)$.

Proposition C.13 (Time complexity of bounds computation)**.** *An ε-approximation of $(\mathcal{L}_n, \mathcal{U}_n)$ can be computed by applying $\mathcal{B}_n$ for a finite number $k(\varepsilon, \gamma)$ of iterations, with*

$$k(\varepsilon, \gamma) = \log_\gamma \frac{1}{\varepsilon(1-\gamma)}.$$



k	0	1	$\cdots$	k
$\mathcal{U} = \mathcal{B}^k(V_{\max})(s)$	$V_{\max}$	$\frac{1}{2} + \gamma V_{\max}$		$\frac{1}{2}(1-\gamma^k)V_{\max} + \gamma^k V_{\max}$
$\mathcal{L} = \mathcal{B}^k(0)(s)$	0	$\frac{1}{2}$		$\frac{1}{2}(1-\gamma^k)V_{\max}$

Figure C.3 – Top: a simple looping MDP with $|S| = |A| = 1$ after having observed a single transition $(n = 1)$. **Bottom**: the sequence of bounds $\mathcal{B}_1^k(0)$ and $\mathcal{B}_1^k(V_{\max})$. They converge geometrically to their limit $\mathcal{U}_1 = \mathcal{L}_1 = V = \frac{1}{2}V_{\max}$, thus in infinite time.

Proof. $\mathcal{B}_n$ is a γ-contraction by Lemma 6.16, and $\mathcal{U}_n$ (resp $\mathcal{L}_n$) is at a distance (in $\|\dot{\|}_\infty$) at most $V_{\max}$ of the initial value bound $V_{\max}$ (resp 0). Thus, the k^{th} application of $\mathcal{B}_n$ decreases this error by a factor γ^k, which gives the result. □

The impact of using an ε-approximation of $(\mathcal{L}_n, \mathcal{U}_n)$ during planning is the following:

Proposition C.14 (Effect of early stopping)**.** *Denote the approximate bounds $(\hat{\mathcal{L}}_n, \hat{\mathcal{U}}_n)$ obtained by applying $\mathcal{B}_n^{k(\varepsilon,\gamma)}$ instead of $\mathcal{B}_n^\infty$, and likewise $(\hat{L}_n, \hat{U}_n)$ in their tree version obtained by applying $B_n^{k(\varepsilon,\gamma)}$ instead of B_n^∞. Then, running GBOP-D with $\hat{\mathcal{L}}_n, \hat{\mathcal{U}}_n$ gives the following regret:*

$$r_n = \tilde{\mathcal{O}}\left(n^{-\log \frac{1}{\gamma} / \log \hat{\kappa}_\infty}\right),$$

with

$$\kappa_\infty \leq \hat{\kappa}_\infty \triangleq \lim_{n\to\infty} \kappa(\hat{L}_n, \hat{U}_n) \leq \kappa.$$

Moreover, the approximation gap $\kappa_\infty - \hat{\kappa}_\infty$ is non-increasing with respect to ε.

It is difficult to control more explicitly the gap between κ_∞ and $\hat{\kappa}_\infty$, which might be discontinuous with ε.

Proof. Note that $\hat{L}_n$ and $\hat{U}_n$ are valid monotonic bounds on V, verifying

$$0 \leq \hat{L}_n \leq L_n \leq V \leq U_n \leq \hat{U}_n \leq V_{\max}.$$

Thus, Lemma C.5 holds with the difference that we only have an inequality

$$\overline{U_n}(a) \geq \max_{a' \in a\mathcal{A}^\infty} \overline{U}(a')$$

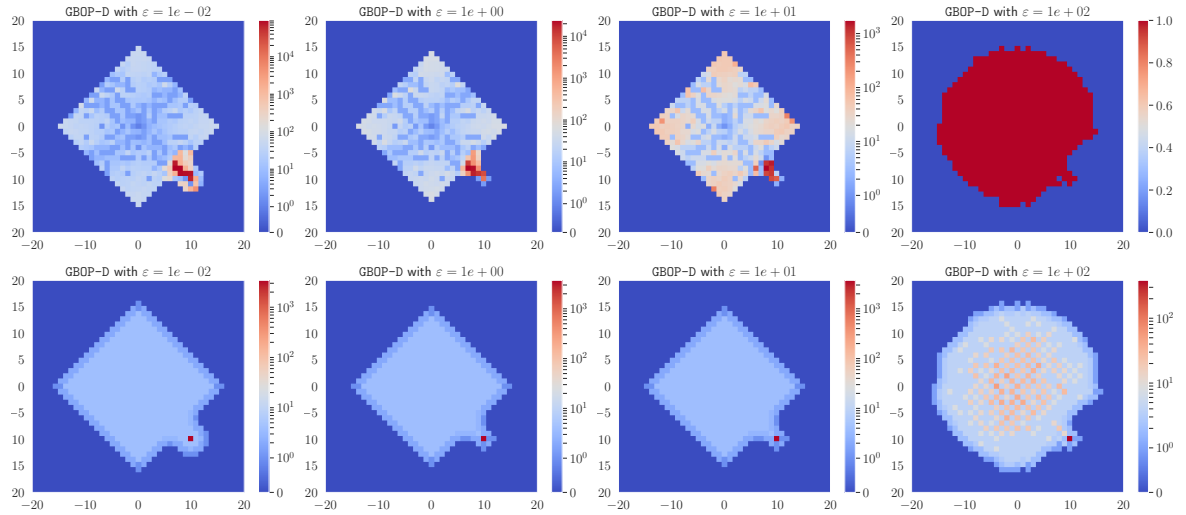


Figure C.4 – Top row: Map showing the number of updates triggered by $\mathcal{B}$ at each expanded state for various values of ε. **Bottom row**: Map showing the corresponding state occupations. The rightmost column corresponds to $\varepsilon \geq V_{\max} = 20$, *i.e.* every approximation stops after a single iteration.

rather than an equality, by monotonicity but non-invariance by B_n. However, this was the actual inequality used in Lemma 6.25, which still holds by replacing L_n, U_n by their approximation $\hat{L}_n, \hat{U}_n$. Likewise, Lemma 6.26 holds. The proof of Theorem 6.27, can be written with the modification that expanded nodes belong to $T_h^\infty(\hat{L}_n, \hat{U}_n))$, which gives the claimed bound.

As ε decreases, $k(\varepsilon, \gamma)$ increases, which means by Lemma 6.15 that $(\hat{\mathcal{L}}_n, \hat{\mathcal{U}}_n)$ get tighter and $\hat{\boldsymbol{\kappa}}_\infty$ shrinks by Lemma 6.22. It reaches its minimum $\boldsymbol{\kappa}_\infty$ when $\varepsilon = 0$. □

Thus, we observe that there is a *tradeoff* between the time complexity $k(\varepsilon)$ and the sample complexity $\hat{\boldsymbol{\kappa}}_\infty$: decreasing one increases the other. We illustrate this trade-off empirically in Figure C.4.

Note that OPD uses d_n iterations of B_n, which corresponds to a tuning of ε with n: $\varepsilon_n = \frac{\gamma^{d_n}}{1-\gamma} = \mathcal{O}(n^{-\frac{\log 1/\gamma}{\log \kappa}})$.

Sampling rule The sampling rule of GBOP-D (line 2 of GBOP-D) can yield an infinite sequence b_n. We propose to stop the sampling after a fixed depth d_n^+.

Proposition C.15 (Time complexity of sampling)**.** *Consider the variant of GBOP-D where we stop the sampling rule when reaching a fixed depth d_n^+ chosen polynomial with n:*

$$d_n^+ = \lceil \alpha n^\beta \rceil, \text{ with } \alpha, \beta > 0$$

> *Then, the regret bound of Theorem 6.27 (or that of Proposition C.14 when using early stopping in the bounds computation) still holds.*

Note that this is not too constraining compared to OPD, for which the sampling rule complexity d_n is upper-bounded by n. Hence, by choosing $\alpha = \beta = 1$, GBOP-D preserve the same complexity as OPD in the worst case.

Proof. Let $\kappa' > \boldsymbol{\kappa}_\infty$ (or $\kappa' > \hat{\boldsymbol{\kappa}}_\infty$ under approximate bounds). In the proof of Theorem 6.27, it is shown that the maximum depth d_n of an expanded node is at least $d_n^- \triangleq \log_{\kappa'} \frac{n - C_0}{C_1'}$, which allows to conclude with Lemma 6.26 that $r_n = \mathcal{O}(\gamma^{d_n}) = \mathcal{O}(\gamma^{d_n^-})$. By choosing d_n^+ polynomial, we have that d_n^+ is greater than d_n^- for n sufficiently high. Thus, by stopping the sampling after reaching a depth d_n^+, we have that $r_n \leq \gamma^{\min\{d_n, d_n^+\}}/(1-\gamma) = \mathcal{O}(\gamma^{d_n^-}) = \mathcal{O}(n^{\frac{-\log 1/\gamma}{\log \kappa'}})$ □

Efficient implementation of $\mathcal{B}_n^\infty$

The bounds $\mathcal{L}_n$ and $\mathcal{U}_n$ are computed by fixed-point iteration of $\mathcal{B}_n$ from the trivial bounds $(0, V_{\max})$. The naive implementation of $\mathcal{B}_n$ requires to iterate over the whole set of state-action pairs in $\mathcal{G}_n$. Two ideas can be used to increase the efficiency of both steps:

(i) Instead of starting the iteration with the trivial bounds, the previous estimate $\mathcal{L}_{n-1}, \mathcal{U}_{n-1}$ can be used instead at iteration n. Since these bounds are closer to their limit ($0 \leq \mathcal{L}_{n-1} \leq \mathcal{L}_n$ and $\mathcal{U}_n \leq \mathcal{U}_{n-1} \leq V_{\max}$), the fixed-point iteration will converge quicker.

(ii) In particular, since $\mathcal{L}_{n-1}$ and $\mathcal{U}_{n-1}$ are invariant by $\mathcal{B}_n$, the the only nodes modified by a supplementary application of $\mathcal{B}_n$ are the parents of only updated node: the expanded state s_n. Once its value is updated by $\mathcal{B}_n$, the same reasoning can be applied for the next iteration of $\mathcal{B}_n$: only its predecessors can be updated. Thus, we can keep track of a set q of states that can be updated, for every application of $\mathcal{B}_n$.

These idea are formalised in Algorithm C.2. Note that the criterion $\|\mathcal{B}_n^{k+1} - \mathcal{B}_n^k\| \leq \frac{1-\gamma}{\gamma}\varepsilon$ is used to detect that the limit $\mathcal{B}_n^\infty$ is approximated with accuracy ε, and stems from $\mathcal{B}_n$ being a γ-contraction:

Proof. $\|\mathcal{B}_n^k - \mathcal{B}_n^\infty\| \leq \gamma\|\mathcal{B}_n^{k+1} - \mathcal{B}_n^\infty\| \leq \gamma\|\mathcal{B}_n^{k+1} - \mathcal{B}_n^k\| + \gamma\|\mathcal{B}_n^k - \mathcal{B}_n^\infty\|$, with $\|\mathcal{B}_n^{k+1} - \mathcal{B}_n^k\| \leq \frac{1-\gamma}{\gamma}\varepsilon$, thus $\|\mathcal{B}_n^k - \mathcal{B}_n^\infty\| \leq \varepsilon$. □

Algorithm C.2: A queue-based implementation of $\mathcal{B}_n^\infty$.

Input: Initial bound $\mathcal{U}_{n-1}$, expanded node s_n, accuracy ε
Output: An ε-approximation of $\mathcal{U}_n$
$\mathcal{U}_n \leftarrow \mathcal{U}_{n-1}$
$q \leftarrow [s_n]$
while *q is not empty* **do**
    $s' \leftarrow$ Pop the first node from the queue q
    $\mathcal{U}' \leftarrow \mathcal{B}_n(\mathcal{U}_n)(s')$ ▷ `Node backup`
    if $\mathcal{U}' - \mathcal{U}_n > \frac{1-\gamma}{\gamma}\varepsilon$ **then** ▷ `Stopping rule`
        Push the predecessors s of s' to the queue q ▷ `Propagation rule`
    $\mathcal{U}_n(s') \leftarrow \mathcal{U}'$
return $\mathcal{U}_n$

Appendix D

Complements on Chapter 7

D.1 Proofs

D.1.1 Proof of Proposition 7.5

Proof. We differentiate $J(\theta) = \sum_{n=1}^{N} \|y_n - \Phi_n\theta\|^2_{\Sigma_p^{-1}} + \lambda\|\theta\|^2$ as in (7.8) with respect to θ:

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_{n=1}^{N} \nabla_\theta (y_n - \Phi_n\theta)^\intercal \Sigma_p^{-1}(y_n - \Phi_n\theta) + \nabla_\theta \lambda\|\theta\|^2 \\ &= -2\sum_{n=1}^{N} y_n^\intercal \Sigma_p^{-1}\Phi_n + 2\sum_{n=1}^{N} \theta^\intercal(\Phi_n^\intercal\Sigma^{-1}\Phi_n) + 2\lambda\theta^\intercal\end{aligned}$$

Hence,

$$\nabla_\theta J(\theta) = 0 \iff \left(\sum_{n=1}^{N} \Phi_n^\intercal\Sigma_p^{-1}\Phi_n + I_d\right)\theta = \sum_{n=1}^{N} y_n^\intercal\Sigma_p^{-1}\Phi_n$$

□

D.1.2 Proof of Theorem 7.8

We start by showing a preliminary proposition:

Proposition D.1 (Matrix version of Theorem 1 of Abbasi-Yadkori, Pál, and Szepesvári, 2011)**.** *Let $\{F_n\}_{n=0}$ be a filtration. Let $\{\eta_n\}_{n=1}^{\infty}$ be a $\mathbb{R}^p$-valued stochastic process such that η_n is F_n-measurable and $\mathbb{E}[\eta_n \mid F_{n-1}]$ is Σ_p-sub-Gaussian.*

Let $\{\Phi_n\}_{n=1}^{\infty}$ be an $\mathbb{R}^{p\times d}$-valued stochastic process such that Φ_n is F_n-measurable. Assume that G is a $d \times d$ positive definite matrix. For any $n \geq 0$, define

$$\overline{G}_n = G + \sum_{s=1}^{n} \Phi_s^\intercal \Sigma_p^{-1} \Phi_s \in \mathbb{R}^{d\times d} \quad S_n = \sum_{s=1}^{n} \Phi_s^\intercal \Sigma_p^{-1} \eta_s \in \mathbb{R}^d.$$

Then, for any $\delta > 0$, with probability at least $1-\delta$, for all $n \geq 0$,

$$\|S_n\|_{\overline{G}_n^{-1}} \leq \sqrt{2\log\left(\frac{\det\left(\overline{G}_n\right)^{1/2}}{\delta \det(G)^{1/2}}\right)}.$$

Proof. Let

$$G_t = \sum_{s=1}^{t} \Phi_s^\intercal \Sigma_p^{-1} \Phi_s \in \mathbb{R}^{d\times d}$$

And for any $z \in \mathbb{R}^d$,

$$M_t^z = \exp\left(\langle z, S_t\rangle - \frac{1}{2}\|z\|_{G_t}\right)$$

$$D_t^z = \exp\left(\langle \Phi_t z, \eta_t\rangle_{\Sigma_p^{-1}} - \frac{1}{2}\|\Phi_t z\|_{\Sigma_p^{-1}}\right)$$

Then,

$$\begin{aligned} M_t^z &= \exp\left(\sum_{s=1}^{t} z^\intercal \Phi_s^\intercal \Sigma_p^{-1} \eta_s - \frac{1}{2}(\Phi_s z)^\intercal \Sigma_p^{-1}(\Phi_s z)\right) \\ &= \prod_{s=1}^{t} D_s^z \end{aligned}$$

and using the sub-Gaussianity of η_t

$$\begin{aligned} \mathbb{E}[D_t^z \mid F_{t-1}] = &\exp\left(-\frac{1}{2}\|\Phi_t z\|_{\Sigma_p^{-1}}\right) \\ &\mathbb{E}\left[\exp\left(\langle \Phi_t z, \eta_t\rangle_{\Sigma_p^{-1}}\right) \,\middle|\, F_{t-1}\right] \\ \leq &\exp\left(-\frac{1}{2}\|\Phi_t z\|_{\Sigma_p^{-1}}\right) \end{aligned}$$

$$\exp\left((z^\intercal \Phi_t^\intercal \Sigma_p^{-1})\Sigma_p(\Sigma_p^{-1}\Phi_t z)\right)$$
$$= 1$$

$$\mathbb{E}\left[M_t^z \mid F_{t-1}\right] = \left(\prod_{s=1}^{t-1} D_s^z\right)\mathbb{E}\left[D_t^z \mid F_{t-1}\right] \leq M_{t-1}^z$$

Showing that $(M_t^z)_{t=1}^\infty$ is indeed a supermartingale and in fact $\mathbb{E}[M_t^z] \leq 1$. It then follows by Doob's upcrossing lemma for supermartingale that $M_\infty^z = \lim_{t\to\infty} M_t^z$ is almost surely well-defined, and so is M_τ^z for any random stopping time τ.

Next, we consider the stopped martingale $M_{\min(\tau,t)}^z$. Since $(M_t^z)_{t=1}^\infty$ is a non-negative supermartingale and τ is a random stopping time, we deduce by Doob's decomposition that

$$\begin{aligned}\mathbb{E}[M_{\min(\tau,t)}^z] &= \mathbb{E}[M_0^z] + \mathbb{E}[\sum_{s=0}^{t-1}(M_{s+1}^z - M_s^z)\mathbb{I}\{\tau > s\}] \\ &\leq 1 + \mathbb{E}[\sum_{s=0}^{t-1}\mathbb{E}[M_{s+1}^z - M_s^z|F_s]\mathbb{I}\{\tau > s\}] \\ &\leq 1\end{aligned}$$

Finally, an application of Fatou's lemma show that

$$\mathbb{E}[M_\tau^z] = \mathbb{E}[\liminf_{t\to\infty} M_{\min(\tau,t)}^z] \leq \liminf_{t\to\infty}\mathbb{E}[M_{\min(\tau,t)}^z] \leq 1.$$

This results allows to apply a result from (Peña, Lai, and Shao, 2008).

Lemma D.2 (Theorem 14.7 of Peña, Lai, and Shao, 2008)**.** *If Z is a random vector and B is a symmetric positive definite matrix such that*

$$\forall\gamma \in \mathbb{R}^d, \log\mathbb{E}\exp\left(\gamma^\intercal Z - \frac{1}{2}\gamma^\intercal B\gamma\right) \leq 0,$$

then for any positive definite non-random matrix C, it holds

$$\mathbb{E}\left[\sqrt{\frac{\det(C)}{\det(B+C)}}\exp\left(\frac{1}{2}\|Z\|^2_{(B+C)^{-1}}\right)\right] \leq 1.$$

In particular, by Markov inequality, for all $\delta \in (0,1)$,

$$\mathbb{P}\left(\|Z\|_{(B+C)^{-1}} \geq \sqrt{2\log\left(\frac{\det\left((B+C)^{1/2}\right)}{\delta\det(C)^{1/2}}\right)}\right) \leq \delta.$$

Here, by using $Z = \sum_{s=1}^{t} \Phi_s \Sigma_p^{-1} \eta_s$, $B = G_t$, $C = G$,

$$\mathbb{P}\left(\|S_t\|_{(G_t+G)^{-1}} \geq \sqrt{2 \log\left(\frac{\det(G_t+G)^{1/2}}{\delta \det(G)^{1/2}} \right)} \right) \leq \delta$$

□

Having shown this preliminary result, we move on to the proof of Theorem 7.8.

Proof. For all $x \in \mathbb{R}^d$, (7.9) gives

$$\begin{aligned} x^\intercal \theta_{N,\lambda} - x^\intercal \theta &= x^\intercal G_{N,\lambda}^{-1} \sum_{n=1}^{N} \Phi_n^\intercal \Sigma_p^{-1} \eta_n - \lambda x^\intercal G_{N,\lambda}^{-1} \theta \\ &= \langle x, \sum_{n=1}^{N} \Phi_n^\intercal \Sigma_p^{-1} \eta_n \rangle_{G_{N,\lambda}^{-1}} - \lambda \langle x, \theta \rangle_{G_{N,\lambda}^{-1}} \end{aligned}$$

Using the Cauchy-Schwartz inequality, we get

$$|x^\intercal \theta_{N,\lambda} - x^\intercal \theta| \leq \|x\|_{G_{N,\lambda}^{-1}} \left(\left\| \sum_{n=1}^{N} \Phi_n^\intercal \Sigma_p^{-1} \eta_n \right\|_{G_{N,\lambda}^{-1}} + \lambda \|\theta\|_{G_{N,\lambda}^{-1}} \right)$$

In particular, for $x = G_{N,\lambda}(\theta_{N,\lambda} - \theta)$, we get after simplifying with $\|\theta_{N,\lambda} - \theta\|_{G_{N,\lambda}}$,

$$\|\theta_{N,\lambda} - \theta\|_{G_{N,\lambda}} \leq \left\| \sum_{n=1}^{N} \Phi_n^\intercal \Sigma_p^{-1} \eta_n \right\|_{G_{N,\lambda}^{-1}} + \lambda \|\theta\|_{G_{N,\lambda}^{-1}}$$

By applying Proposition D.1 with $G = \lambda I_d$, we obtain that with probability at least $1 - \delta$,

$$\|\theta_{N,\lambda} - \theta\|_{G_{N,\lambda}} \leq \sqrt{2 \log\left(\frac{\det(G_{N,\lambda})^{1/2}}{\delta \det(\lambda I_d)^{1/2}} \right)} + \lambda \|\theta\|_{G_{N,\lambda}^{-1}}$$

And since $\|\theta\|^2_{G_{N,\lambda}^{-1}} \leq 1/\lambda_{\min}(G_{N,\lambda}) \|\theta\|_2^2 \leq 1/\lambda \|\theta\|_2^2$ and $\|\theta\|_2^2 \leq d\|\theta\|_\infty^2 \leq dS^2$,

$$\|\theta_{N,\lambda} - \theta\|_{G_{N,\lambda}} \leq \sqrt{2 \log\left(\frac{\det(G_{N,\lambda})^{1/2}}{\delta \det(\lambda I_d)^{1/2}} \right)} + (\lambda d)^{1/2} S$$

□

D.1.3 Proof of Theorem 7.28

Proof. The predictor designed in Section 7.3 verifies the inclusion property (7.2). Thus, for sequence of controls $\mathbf{u}$, any dynamics $A(\theta) \in C_{[N],\delta}$, and perturbations $\underline{\omega} \leq \omega \leq \overline{\omega}$, the corresponding state at time t_n is bounded by $\underline{x}_n \leq x_n \leq \overline{x}_n$, which implies that $R(x_n) \geq \min_{x \in [\underline{x}_n(\mathbf{u}), \overline{x}_n(\mathbf{u})]} R(x) = \underline{R}_n(\mathbf{u})$.

Thus, by taking the min over $C_{[N],\delta}$ and $[\underline{\omega}, \overline{\omega}]$, we also have for any sequence of controls $\mathbf{u}$,

$$\begin{aligned} V^r(\mathbf{u}) &= \min_{\substack{A(\theta) \in C_{[N],\delta} \\ \underline{\omega} \leq \omega \leq \overline{\omega}}} \sum_{n=N+1}^{\infty} \gamma^n R(x_n) \\ &\geq \sum_{n=N+1}^{\infty} \gamma^n \underline{R}_n(\mathbf{u}) \\ &= \hat{V}^r(\mathbf{u}) \end{aligned}$$

□

D.1.4 Proof of Theorem 7.30

We first bound the model estimation error.

Lemma D.3.

$$\|A(\theta) - A(\theta_{N,\lambda})\|_F = \mathcal{O}\left(\sqrt{\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})}}\right)$$

Proof. We have

$$\|\theta - \theta_{N,\lambda}\|^2_{G_{N,\lambda}} \geq \lambda_{\min}(G_{N,\lambda}) \|\theta - \theta_{N,\lambda}\|_2^2$$

And (7.12) gives

$$\|\theta - \theta_{N,\lambda}\|^2_{G_{N,\lambda}} = \mathcal{O}(\beta_N(\delta)^2)$$

Moreover, $A(\theta)$ belongs to a linear image of this L^2-ball. By writing a the j^{th} column of a matrix M as M_j, and its coefficient i, j as $M_{i,j}$,

$$\begin{aligned} ((A(\theta) - A(\theta_{N,\lambda}))^\intercal (A(\theta) - A(\theta_{N,\lambda})))_{i,j} &= (\theta - \theta_{N,\lambda})^\intercal \phi_i^\intercal \phi_j (\theta - \theta_{N,\lambda}) \\ &\leq \lambda_{\max}(\phi_i^\intercal \phi_j) \|\theta - \theta_{N,\lambda}\|_2^2 \end{aligned}$$

Thus, $\|A(\theta) - A(\theta_{N,\lambda})\|_F^2 = \mathrm{Tr}\left[(A(\theta) - A(\theta_{N,\lambda}))^\intercal (A(\theta) - A(\theta_{N,\lambda}))\right] = \mathcal{O}\left(\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})}\right)$ □

Then, we propagate this estimation error through the state prediction.

Lemma D.4. *If there exist $P > 0, Q_0 \in \mathbb{R}^{p\times p}$, $\rho > 0$ such that*

$$\begin{bmatrix} A_N^\intercal P + P A_N + Q_0 & P|D| \\ |D|^\intercal P & -\rho I_r \end{bmatrix} < 0,$$

then for all $t > t_N$,

$$\|\overline{x}(t) - \underline{x}(t)\| \leq \left(C_0 + \mathcal{O}\left(\frac{\beta_N(\delta)}{\sqrt{\lambda_{\min}(G_{N,\lambda})}}\right)\right) C_\omega(t),$$

where

$$C_0 = \sqrt{\frac{2\rho\lambda_{\max}(P)}{\lambda_{\min}(P)\lambda_{\min}(Q_0)}},$$

and

$$C_\omega(t) = \sup_{\tau\in[0,t]} \|\overline{\omega}(\tau) - \underline{\omega}(\tau)\|_2.$$

Proof. Let $e = \overline{x} - \underline{x}$. (7.20) gives the dynamics

$$\dot{e} = A_N e + |\Delta A|(\overline{x}^+ + \underline{x}^-) + |D|(\overline{\omega} - \underline{\omega})$$

where recall that $|M| = M^+ + M^-$ for any matrix $M \in \mathbb{R}^{p\times p}$.

We define the Lyapunov function $V = e^\intercal P e$, which is non-negative definite provided that $P > 0$, and compute its derivative

$$\dot{V} = X^\intercal \begin{bmatrix} A_N^\intercal P + P A_N + Q & P|D| & P|\Delta A| \\ |D|^\intercal P & -\rho I_r & 0 \\ |\Delta A|^\intercal P & 0 & -\alpha I_p \end{bmatrix} X$$
$$- e^\intercal Q e + \alpha|\underline{x}^+ + \overline{x}^-|^2 + \rho|\overline{\omega} - \underline{\omega}|^2$$

with $X = \begin{bmatrix} e & \overline{\omega} - \underline{\omega} & \underline{x}^+ + \overline{x}^- \end{bmatrix}^\intercal$, for any $Q \in \mathbb{R}^{p\times p}$, $\rho, \alpha \in \mathbb{R}$.

Moreover, it holds that $-\underline{x}^+ - \overline{x}^- \leq e \leq \overline{x}^+ + \underline{x}^-$, which implies $|\underline{x}^+ + \overline{x}^-| \leq 2|e|$. Hence,

$$\dot{V} \leq X^\intercal \underbrace{\left[\begin{array}{cc|c} A_N^\intercal P + PA_N + Q + 4\alpha I_p & P|D| & P|\Delta A| \\ |D|^\intercal P & -\rho I_r & 0 \\ \hline |\Delta A|^\intercal P & 0 & -\alpha I_p \end{array}\right]}_{\Upsilon} X$$
$$- e^\intercal Q e + \rho \|\overline{\omega} - \underline{\omega}\|_2^2$$

Thus, if we had $\Upsilon \leq 0$, $Q > 0$, $\rho > 0$, then we would have

$$\dot{V} \leq -\mu V + \rho \|\overline{\omega} - \underline{\omega}\|_2^2$$

with $\mu = \frac{\lambda_{\min}(Q)}{\lambda_{\max}(P)}$. Since $V(t_N) = 0$, this further implies that for all $t > t_N$,

$$V(t) \leq \frac{\rho}{\mu} C_\omega^2(t) \tag{D.1}$$

We now examine the condition $\Upsilon \leq 0$. We resort to its Schur complement: given $\alpha > 0$, $\Upsilon \leq 0$ if and only if $R \geq S$, where $S = \alpha^{-1} \begin{bmatrix} |\Delta A|^\intercal P & 0 \end{bmatrix}^\intercal \begin{bmatrix} |\Delta A|^\intercal P & 0 \end{bmatrix}$ and R is the top-left block of $-\Upsilon$:

$$R = \begin{bmatrix} -A_N^\intercal P - PA_N - Q - 4\alpha I_p & -P|D| \\ -|D|^\intercal P & \rho I_r \end{bmatrix}$$

Choose $Q = \frac{1}{2} Q_0 - 4\alpha I_p$. Assume that P is fixed and satisfies the conditions of the lemma. We have

$$\begin{aligned} \lambda_{\max}(S) &\leq \alpha^{-1} \lambda_{\max}(P)^2 \lambda_{\max}(|\Delta A|^\top |\Delta A|) \\ &\leq \alpha^{-1} \lambda_{\max}(P)^2 \|\Delta A\|_F^2 \end{aligned}$$

Thus, by taking $\alpha = \frac{2\lambda_{\max}(P)^2 \|\Delta A\|_F^2}{\lambda_{\min}(Q_0)} = \mathcal{O}(\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})})$, we can obtain that $S \leq \begin{bmatrix} \frac{1}{2} Q_0 & 0 \\ 0 & 0 \end{bmatrix}$. Thus,

$$R - S \geq \begin{bmatrix} -A_N^\intercal P - PA_N - Q_0 & -P|D| \\ -|D|^\intercal P & \rho I_r \end{bmatrix} > 0$$

as it is assumed in the conditions of the lemma. Hence, under such a choice of α and Q, we recover $\Upsilon \leq 0$. (D.1) follows with $\mu = \frac{\lambda_{\min}(Q)}{\lambda_{\max}(P)} = \frac{\frac{1}{2}\lambda_{\min}(Q_0) - 4\alpha}{\lambda_{\max}(P)}$. Finally, we obtain

$$\begin{aligned} \|e(t)\|_2^2 &\leq \lambda_{\min}(P)^{-1} V(t) \\ &\leq \frac{2\rho \lambda_{\max}(P)/\lambda_{\min}(P)}{\lambda_{\min}(Q_0) - 8\alpha} C_\omega^2(t) \end{aligned}$$

Developing at the first order in α gives

$$\begin{aligned}\|e(t)\|_2 &\leq C_0 \left(1 + \frac{4\alpha}{\lambda_{\min}(Q_0)} + \mathcal{O}(\alpha^2)\right) C_\omega(t) \\ &\leq \left(C_0 + \mathcal{O}\left(\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})}\right)\right) C_\omega(t)\end{aligned}$$

□

Finally, we propagate the state prediction error bound to the pessimistic rewards and surrogate objective to get our final result.

Proof. For any sequence of controls $\mathbf{u}$, dynamical parameters $\theta \in C_{N,\delta}$ and disturbances $\underline{\omega} \leq \omega \leq \overline{\omega}$, we clearly have

$$V(\mathbf{u})^r \leq V(\mathbf{u}) = \mathop{\mathbb{E}}_{\omega} \sum_n \gamma^n R(x_n)$$

Moreover, by the inclusion property (7.2), we have that $\underline{x}_n \leq x_n \leq \overline{x}_n$, which implies that $R(x_n) \leq \max_{x \in [\underline{x}_n(\mathbf{u}), \overline{x}_n(\mathbf{u})]} R(x)$. Assuming R is L-lipschitz,

$$\begin{aligned}V(\mathbf{u}) - \hat{V}^r(\mathbf{u}) &\leq \sum_{n=N+1}^{\infty} \gamma^n \max_{x \in [\underline{x}_n(\mathbf{u}), \overline{x}_n(\mathbf{u})]} (\max - \min) R(x) \\ &\leq \sum_{n=N+1}^{\infty} \gamma^n L \left\|\underline{x}_n(\mathbf{u}) - \overline{x}_n(\mathbf{u})\right\|_2 \\ &\leq L \left(C_0 + \mathcal{O}\left(\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})}\right)\right) \sum_{n>N} \gamma^n C_\omega(t_n) \\ &= \Delta_\omega + \mathcal{O}\left(\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})}\right)\end{aligned}$$

with $\Delta_\omega = LC_0 \sum_{n>N} \gamma^n C_\omega(t_n)$, which is finite by Assumption 7.6.

Finally, we use the result of Lemma 7.29 to account for planning with a finite budget, and relate $\hat{V}^r(a^\star)$ to $\hat{V}^r(a_K)$. □

D.1.5 Proof of Corollary 7.31

Proof. By (7.9) and (7.32), we have

$$\lambda_{\min}(G_{N,\lambda}) \geq (N - n_0)\underline{\phi}^2 + \sum_{n<n_0} \Phi_n^\intercal \Sigma_p^{-1} \Phi_n$$

and by (7.12),

$$\begin{aligned}\beta_N(\delta) &= \sqrt{2\log\left(\frac{\det(G_{N,\lambda})^{1/2}}{\delta\det(\lambda I_d)^{1/2}}\right)} + (\lambda d)^{1/2}S \\ &\leq \sqrt{\log\left(N^{d/2}\overline{\phi}^d/(\delta\lambda^{d/2})\right)} + \mathcal{O}(1)\end{aligned}$$

Thus,

$$\frac{\beta_N(\delta)^2}{\lambda_{\min}(G_{N,\lambda})} = \mathcal{O}\left(\frac{\log(N^{d/2}/\delta)}{N}\right)$$

Stability condition 2. By Lemma D.3 and the above, the sequence $(A_N)_N$ converges to $A(\theta)$ in Frobenius norm. Thus,

$$M_n \triangleq \begin{bmatrix} A_N^\intercal P + PA_N + Q_0 & P|D| \\ |D|^\intercal P & -\rho I_r \end{bmatrix} \text{ also converges to } M \triangleq \begin{bmatrix} A(\theta)^\intercal P + PA(\theta) + Q_0 & P|D| \\ |D|^\intercal P & -\rho I_r \end{bmatrix},$$

which is assumed to be negative definite.

Moreover, the two functions that map a matrix to its characteristic polynomial and a polynomial to its roots, are both continuous. Thus, by continuity, the largest eigenvalue of M_n converges to that of M, which is strictly negative. Hence, there exists some $N_0 \in \mathbb{N}$ such that for all $N > N_0$, M_N is negative definite, as required in the condition 2. of Theorem 7.30. □

D.1.6 Proof of Proposition 7.35

We start by showing the following lemma:

Lemma D.5 (Robust values ordering)**.** *In addition to the robust U-value defined in (7.34), that we extend to inner nodes*

$$U_a^r(k) \triangleq \begin{cases} \min_{m\in[M]} \sum_{n=0}^{h-1} \gamma^n R_n^m + \frac{\gamma^h}{1-\gamma} & \text{if } a \text{ is a leaf;} \\ \max_{b\in\mathcal{A}} U_{ab}^r(k) & \text{else.} \end{cases}, \tag{D.2}$$

we also define the robust value of a sequence of actions a

$$V_a^r \triangleq \max_{\mathbf{u}\in a\mathcal{A}^\infty} \min_{m\in[M]} \sum_{n=h(a)+1}^{\infty} \gamma^n R_n^m \tag{D.3}$$

and the robust U-values of a sequence of action a

$$L_a^r(K) \triangleq \begin{cases} \min_{m\in[M]} \sum_{n=0}^{h-1} \gamma^n R_n^m & \text{if } a \text{ is a leaf;} \\ \max_{b\in\mathcal{A}} L_{ab}^r(n) & \text{else.} \end{cases} \tag{D.4}$$

Then, the robust values, L-values and U-values exhibit similar properties as the optimal values, L-values and U-values, that is: for all $0 < k < K$ and $a \in \mathcal{T}$,

$$L_a^r(k) \leq L_a^r(K) \leq V_a^r \leq U_a^r(K) \leq U_a^r(k) \tag{D.5}$$

Proof. By definition, when starting with sequence a, the value $L_a^m(k)$ represents the minimum admissible reward, while $U_a^m(k)$ corresponds to the best admissible reward achievable with respect to the the possible continuations of a. Thus, for all $a \in \mathcal{A}^\star$, $L_a^m(k)$ and $L_a^r(k)$ are non-decreasing functions of k and $U_a^m(k)$ and $U_a^r(k)$ are a non-increasing functions of k, while V_a^m and V_a^r do not depend on k.

Moreover, since the reward function R is assumed be bounded in $[0, 1]$, the sum of discounted rewards from a node of depth d is at most $\gamma^d + \gamma^{d+1} + \cdots = \frac{\gamma^d}{1-\gamma}$. As a consequence, for all $k \geq 0$, $a \in \mathcal{L}_k$ of depth d, and any sequence of rewards $(R_n)_{n\in\mathbb{N}}$ obtained from following a path in $a\mathcal{A}^\infty$ with any dynamics $m \in [M]$:

$$U_a^m(k) = \sum_{n=0}^{d-1} \gamma^n R_n^m \leq \sum_{n=0}^{\infty} \gamma^n R_n^m \leq \sum_{n=0}^{d-1} \gamma^n R_n^m + \frac{\gamma^d}{1-\gamma} = B_a^m(k)$$

Hence,

$$\min_{m\in[M]} U_a^m(k) \leq \min_{m\in[M]} \sum_{n=0}^{\infty} \gamma^n R_n \leq \min_{m\in[M]} B_a^m(k) \tag{D.6}$$

And as the left-hand and right-hand sides of (D.6) are independent of the particular path that was followed in $a\mathcal{A}^\infty$, it also holds for the robust path

$$\min_{m\in[M]} U_i^m(k) \leq \max_{a'\in a\mathcal{A}^\infty} \min_{m\in[M]} \sum_{t=0}^{\infty} \gamma^n R_n^m \leq \min_{m\in[M]} B_i^m(k)$$

that is,

$$L_a^r(k) \leq V_a^r \leq U_a^r(k) \tag{D.7}$$

Finally, (D.7) is extended to the rest of $\mathcal{T}_k$ by recursive application of (D.3), (D.4) and (D.2). □

We now turn to the proof of the theorem.

Proof. Hren and Rémi Munos (2008) first show in Theorem 2 that the simple regret r_K of their optimistic planner is bounded by $\frac{\gamma^{d_K}}{1-\gamma}$ where d_K is the depth of $\mathcal{T}_K$. This properties relies on the fact that the returned action belongs to the deepest explored branch, which we can show likewise by contradiction using Lemma D.5. This yields directly that the returned action $a = i_0$ where i is some node of maximal depth d_K expanded at round $k \leq K$, which by selection rule verifies $U_a^r(k) = U_i^r(k) = \max_{x\in\mathcal{A}} U_x^r(k)$ and

$$\begin{aligned} V^r - V_a^r &= V_{a^\star}^r - V_a^r \\ &\leq U_{a^\star}^r(k) - V_a^r \\ &\leq U_a^r(k) - L_a^r(k) \\ &= U_i^r(k) - L_i^r(k) \\ &= \frac{\gamma^{d_K}}{1-\gamma}. \end{aligned}$$

Secondly, they bound the depth d_K of $\mathcal{T}_K$ with respect to K. To that end, they show that the expanded nodes always belong to the sub-tree $\mathcal{T}_\infty$ of all the nodes of depth d that are $\frac{\gamma^d}{1-\gamma}$-optimal. Indeed, if a node i of depth d is expanded at round k, then $U_i^r(k) \geq U_j^r(k)$ for all $j \in \mathcal{L}_k$ by selection rule, thus the max-backups of (7.34) up to the root yield $U_i^r(k) = U_\emptyset^r(k)$. Moreover, by Lemma D.5 we have that $U_\emptyset^r(k) \geq V_\emptyset^r = V^r$ and so $V_i^r \geq L_i^r(k) = U_i^r(k) - \frac{\gamma^d}{1-\gamma} \geq V^r - \frac{\gamma^d}{1-\gamma}$, thus $i \in \mathcal{T}_\infty$.

Then from the definition of κ applied to nodes in $\mathcal{T}_\infty$, there exists d_0 and c such that the number n_d of nodes of depth $d \geq d_0$ in $\mathcal{T}_\infty$ is bounded by $c\kappa^d$. As a consequence,

$$K = \sum_{d=0}^{d_K} n_d = n_0 + \sum_{d=d_0+1}^{d_K} n_d \leq n_0 + c\sum_{d=d_0+1}^{d_K} \kappa^d.$$

- If $\kappa > 1$, then $K \leq n_0 + c\kappa^{d_0+1}\frac{\kappa^{d_K-d_0}-1}{\kappa-1}$ and thus $d_K \geq d_0 + \log_\kappa \frac{(K-n_0)(\kappa-1)}{c\kappa^{d_0+1}}$.
  We conclude that $r_K \leq \frac{\gamma^{d_K}}{1-\gamma} = \frac{1}{1-\gamma}\left(\frac{(K-n_0)(\kappa-1)}{c\kappa^{d_0+1}}\right)^{\frac{\log\gamma}{\log\kappa}} = \mathcal{O}\left(K^{-\frac{\log 1/\gamma}{\log\kappa}}\right)$.
- If $\kappa = 1$, then $K \leq n_0 + c(d_K - d_0)$, hence we have $r_K = O\left(\gamma^{Kc}\right)$.

□

D.2 A tighter enclosing polytope

Lemma D.6 (Confidence polytope)**.** *We can enclose the confidence ellipsoid obtained in* (7.12) *within a polytope*

$$\mathcal{P} = \left\{ A_0 + \sum_{i=1}^{2^d} \lambda_i \Delta A_i : \lambda \in [0,1]^{2^d}, \sum_{i=1}^{2^d} \lambda_i = 1 \right\}. \tag{D.8}$$

with

$$\begin{aligned} &h_k \text{ is the } k^{th} \text{ element of } \{-1,1\}^d \text{ for } k \in [2^d], \\ &G_{N,\lambda} = PDP^{-1}, \quad \Delta\theta_k = \beta_N(\delta)^{1/2} P^{-1} D^{-1/2} h_k, \\ &A_0 = A + \theta_{N,\lambda}^\intercal \phi, \quad \Delta A_k = \Delta\theta_k^\intercal \phi. \end{aligned}$$

Proof. The ellipsoid in (7.12) is described by

$$\begin{aligned} \theta \in \mathcal{C}_{[N],\delta} &\implies (\theta - \theta_{N,\lambda})^\intercal G_{N,\lambda} (\theta - \theta_{N,\lambda}) \leq \beta_N(\delta) \\ &\implies (\theta' - \theta'_{N,\lambda})^\intercal D (\theta' - \theta'_{N,\lambda}) \leq \beta_N(\delta) \\ &\implies \sum_{i=1}^{d} D_{i,i} (\theta'_i - \theta'_{N,\lambda,i})^2 \leq \beta_N(\delta) \\ &\implies \forall i, |\theta'_i - \theta'_{N,\lambda,i}| \leq \beta_N(\delta)^{1/2} D_{i,i}^{-1/2} \end{aligned}$$

This describes a $\mathbb{R}^d$ box containing $\theta' = P\theta$, whose k^{th} vertex is represented by

$$\theta'_{N,\lambda} + \beta_N(\delta)^{1/2} D^{-1/2} h_k.$$

We obtain the corresponding box on θ by transforming each vertex of the box with P^{-1}. □

List of Figures

List of Figures

List of Figures

List of Algorithms

List of Tables

List of References

Abbasi-Yadkori, Yasin, Dávid Pál, and Csaba Szepesvári (2011). Improved Algorithms for Linear Stochastic Bandits. In *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Curran Associates, Inc., pp. 2312–2320.

Abbasi-Yadkori, Yasin and Csaba Szepesvári (June 2011). Regret Bounds for the Adaptive Control of Linear Quadratic Systems. In *Proceedings of the 24th Annual Conference on Learning Theory*. Ed. by Sham M. Kakade and Ulrike von Luxburg. Vol. 19. Proceedings of Machine Learning Research. Budapest, Hungary: PMLR, pp. 1–26.

Abe, Naoki, Prem Melville, Cezar Pendus, Chandan K. Reddy, David L. Jensen, Vince P. Thomas, et al. (2010). Optimizing Debt Collections Using Constrained Reinforcement Learning. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. Washington, DC, USA: Association for Computing Machinery, pp. 75–84.

Abeille, Marc and Alessandro Lazaric (July 2018). Improved Regret Bounds for Thompson Sampling in Linear Quadratic Control Problems. In *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 1–9.

Abraham, C. and J. Thédié (1960). Le prix d'une vie humaine dans les décisions économiques. *Revue Française de Recherche Opérationnelle* 16, pp. 157–168.

Achiam, Joshua, David Held, Aviv Tamar, and Pieter Abbeel (Aug. 2017). Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 22–31.

Adetola, Veronica and Martin Guay (2008). Adaptive Model Predictive Control for Constrained Nonlinear Systems. *IFAC Proceedings Volumes* 41.2. 17th IFAC World Congress, pp. 1946–1951.

Adrot, O. and J.-M. Flaus (Dec. 2003). Trajectory computation of dynamic uncertain systems. In *42nd IEEE International Conference on Decision and Control*. Maui, HI, USA, pp. 1291–1296.

Ait Rami, M., C.H. Cheng, and C. de Prada (Dec. 2008). Tight Robust interval observers: an LP approach. In *Proc. of 47th IEEE Conference on Decision and Control*. Cancun, Mexico, pp. 2967–2972.

Altché, Florent and Arnaud de La Fortelle (2016). Analysis of optimal solutions to robot coordination problems to improve autonomous intersection management policies. In *2016 IEEE Intelligent Vehicles Symposium, IV 2016, June 19-22*. Gotenburg, Sweden: IEEE, pp. 86–91.

Altché, Florent, Xiangjun Qian, and Arnaud de La Fortelle (2017). An Algorithm for Supervised Driving of Cooperative Semi-Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 18.12, pp. 3527–3539.

Altché, Florent, Xiangjun Qian, and Arnaud de La Fortelle (2016). Time-optimal coordination of mobile robots along specified paths. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, October 9-14, 2016*. Daejeon, South Korea: IEEE, pp. 5020–5026.

Altman, Eitan (1999). *Constrained Markov Decision Processes*. Chapman and Hall/CRC.

Amos, Brandon, Ivan Jimenez, Jacob Sacks, Byron Boots, and J. Zico Kolter (2018). Differentiable MPC for End-to-end Planning and Control. In *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 8289–8300.

Artuñedo, A., J. Villagra, and J. Godoy (2019). Real-Time Motion Planning Approach for Automated Driving in Urban Environments. *IEEE Access* 7, pp. 180039–180053.

Artuñedo, A., J. Villagra, J. Godoy, and M. D. d. Castillo (2020). Motion Planning Approach Considering Localization Uncertainty. *IEEE Transactions on Vehicular Technology* 69.6, pp. 5983–5994.

Artzner, Philippe, Freddy Delbaen, Jean-Marc Eber, and David Heath (1999). Coherent Measures of Risk. *Mathematical Finance* 9.3, pp. 203–228.

Åström, Karl Johan (1965). Optimal Control of Markov Processes with Incomplete State Information I. eng. *Journal of Mathematical Analysis and Applications* 10.1, pp. 174–205.

Aswani, Anil, Humberto Gonzalez, S. Shankar Sastry, and Claire Tomlin (2013). Provably safe and robust learning-based model predictive control. *Automatica* 49.5, pp. 1216–1226.

Auer, Peter, Thomas Jaksch, and Ronald Ortner (Dec. 2009). Near-optimal Regret Bounds for Reinforcement Learning. In *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Vancouver, B.C., Canada: Curran Associates, Inc., pp. 89–96.

Awad, Edmond, Sohan Dsouza, Richard Kim, Jonathan Schulz, Joseph Henrich, Azim Shariff, et al. (Nov. 2018). The Moral Machine experiment. *Nature* 563.7729, pp. 59–64.

Awan, M.A. (2014). *Compensation of Low Performance Steering System Using Torque Vectoring*. Cranfield University.

Azar, Mohammad Gheshlaghi, Ian Osband, and Rémi Munos (Aug. 2017). Minimax Regret Bounds for Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 263–272.

Bacha, Andrew, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, et al. (2008). Odin: Team VictorTango's entry in the DARPA Urban Challenge. *Journal of Field Robotics* 25.8, pp. 467–492.

Bacon, Pierre-Luc, Jean Harb, and Doina Precup (Feb. 2017). The Option-Critic Architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, USA: AAAI Press, pp. 1726–1734.

Bagnell, James, David Bradley, David Silver, Boris Sofman, and Anthony Stentz (2010). Learning for autonomous navigation. *IEEE Robotics and Automation Magazine* 17.2, pp. 74–84.

Bai, Haoyu, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee (2015). Intention-aware online POMDP planning for autonomous driving in a crowd. In *IEEE International Conference on Robotics and Automation, ICRA 2015, 26-30 May, 2015*. Seattle, WA, USA: IEEE, pp. 454–460.

Baker, Christopher R. and John M. Dolan (2008). Traffic interaction in the urban challenge: Putting boss on its best behavior. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008*. Acropolis Convention Center, Nice, France: IEEE, pp. 1752–1758.

Ballesteros, Joaquin, Luis Merino, Miguel Angel Trujillo, Antidio Viguria, and Anibal Ollero (May 2013). Improving the efficiency of online POMDPs by using belief similarity measures. In *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany, pp. 1792–1798.

Banach, Stefan (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae* 3.1, pp. 133–181.

Bandyopadhyay, Tirthankar, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus (2013). Intention-Aware Motion Planning. In *Algorithmic Foundations of Robotics X*. Ed. by Emilio Frazzoli, Tomas Lozano-Perez, Nicholas Roy, and Daniela Rus. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 475–491.

Bansal, Mayank, Alex Krizhevsky, and Abhijit Ogale (2018). *ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst*. preprint.

Banzhaf, H. Spencer (Nov. 2014). Retrospectives: The Cold-War Origins of the Value of Statistical Life. *Journal of Economic Perspectives* 28.4, pp. 213–26.

Barbier, Mathieu, Christian Laugier, Olivier Simonin, and Javier Ibañez-Guzmán (Nov. 2018). Probabilistic Decision-Making at Road Intersections: Formulation and Quantitative Evaluation. In *ICARCV 2018 - 15 th International Conference on Control, Automation, Robotics and Vision*. Singapour, Singapore, pp. 1–8.

Basar, T. and P. Bernhard (1996). H-infinity Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach. *IEEE Transactions on Automatic Control* 41.9, pp. 1397–.

Bellman, Richard (2010). *Dynamic Programming*. USA: Princeton University Press.

Ben-Tal, Aharon, Laurent El Ghaoui, and Arkadi Nemirovski (2009). *Robust optimization*. Vol. 28. Princeton University Press.

Berg, Jur van den, Pieter Abbee, and Ken Goldberg (2011). LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research* 30.7, pp. 895–913.

Berg, Jur van den, Sachin Patil, and Ron Alterovitz (2017). Motion Planning Under Uncertainty Using Differential Dynamic Programming in Belief Space. In *Robotics Research : The 15th International Symposium ISRR*. Cham: Springer International Publishing, pp. 473–490.

Berkenkamp, Felix and Angela P. Schoellig (July 2015). Safe and robust learning control with Gaussian processes. In *2015 European Control Conference, ECC 2015, 15-17 July 2015*. Linz, Austria, pp. 2496–2501.

Berkenkamp, Felix, Angela P. Schoellig, and Andreas Krause (May 2016). Safe controller optimization for quadrotors with Gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA), 16-21 May 2016*. Stockholm, Sweden, pp. 491–496.

Bernard, Olivier and Jean-Luc Gouzé (2004). Closed loop observers bundle for uncertain biotechnological models. *Journal of Process Control* 14.7. Dynamics, Monitoring, Control and Optimization of Biological Systems, pp. 765–774.

Bertsimas, Dimitris, David B Brown, and Constantine Caramanis (2011). Theory and applications of robust optimization. *SIAM review* 53.3, pp. 464–501.

Beutler, Frederick J. and Keith W. Ross (1985). Optimal policies for controlled Markov chains with a constraint. *Journal of Mathematical Analysis and Applications* 112.1, pp. 236–252.

Bhattacharyya, Raunak P., Derek J. Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J. Kochenderfer (2018). Multi-Agent Imitation Learning for Driving Simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, October 1-5, 2018*. Madrid, Spain: IEEE, pp. 1534–1539.

Bohren, Jonathan, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, et al. (2008). Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge. *Journal of Field Robotics* 25.9, pp. 598–614.

Bojarski, Mariusz, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, et al. (2016). *End to End Learning for Self-Driving Cars*. preprint.

Bolajraf, M., M. Ait Rami, and U. Helmke (2011). Robust Positive Interval Observers For Uncertain Positive Systems. *IFAC Proceedings Volumes* 44.1. 18th IFAC World Congress, pp. 14330–14334.

Bonnefon, J.-F., A. Shariff, and I. Rahwan (June 2016). The social dilemma of autonomous vehicles. *Science* 352.6293, pp. 1573–1576.

Bouraine, S., Th Fraichard, O. Azouaoui, and Hassen Salhi (June 2014). Passively safe partial motion planning for mobile robots with limited field-of-views in unknown dynamic environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA), 31 May–7 June*. Hong Kong, China, pp. 3576–3582.

Bouraine, Sara, Thierry Fraichard, and Hassen Salhi (Apr. 2012). Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. *Autonomous Robots* 32.3, pp. 267–283.

Boutilier, Craig and Tyler Lu (June 2016). Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI-16), June 25th to 29th*. New York, pp. 52–61.

Bouton, M., A. Cosgun, and M. J. Kochenderfer (June 2017). Belief state planning for autonomously navigating urban intersections. In *2017 IEEE Intelligent Vehicles Symposium (IV), 11-14 June 2017*. Los Angeles, CA, USA, pp. 825–830.

Bouton, M., A. Nakhaei, K. Fujimura, and M. J. Kochenderfer (May 2018). Scalable Decision Making with Sensor Occlusions for Autonomous Driving. In *2018 IEEE International Conference on Robotics and Automation (ICRA), 21-25 May 2018*. Brisbane, QLD, Australia, pp. 2076–2081.

Bouton, Maxime, Jesper Karlsson, Alireza Nakhaei, Kikuo Fujimura, Mykel J Kochenderfer, and Jana Tumova (July 2019). Reinforcement learning with probabilistic guarantees for autonomous driving. In *Workshop on Safety Risk and Uncertainty in Reinforcement Learning, Conference on Uncertainty in Artificial Intelligence (UAI)*. Tel Aviv, Israel.

Bouton, Maxime, Alireza Nakhaei, Kikuo Fujimura, and Mykel J. Kochenderfer (June 2019). Safe Reinforcement Learning with Scene Decomposition for Navigating Complex Urban Environments. In *2019 IEEE Intelligent Vehicles Symposium (IV), 9-12 June*. Paris, France, pp. 1469–1476.

Brechtel, Sebastian, Tobias Gindele, and Rüdiger Dillmann (June 2013). Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation. In *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR, pp. 370–378.

Brechtel, Sebastian, Tobias Gindele, and Rudiger Dillmann (Oct. 2014). Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC), 8-11 Oct*. Qingdao, China, pp. 392–399.

Bry, Adam and Nicholas Roy (May 2011). Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation, 9-13 May*. Shanghai, China, pp. 723–730.

Bubeck, Sébastien and Rémi Munos (June 2010). Open Loop Optimistic Planning. In *The 23rd Conference on Learning Theory (COLT 2010), June 27–29*. Haifa, Israel, pp. 477–489.

Bubeck, Sébastien, Gilles Stoltz, Csaba Szepesvári, and Rémi Munos (2009). Online Optimization in X-Armed Bandits. In *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Curran Associates, Inc., pp. 201–208.

Buehler, Martin, Karl Iagnemma, and Sanjiv Singh (2009). *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. 1st. Springer Publishing Company, Incorporated.

Buşoniu, Lucian, Alexander Daniels, Remi Munos, and Robert Babuska (Apr. 2013). Optimistic planning for continuous-action deterministic systems. *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 16-19 April*, pp. 69–76.

Buşoniu, Lucian and Remi Munos (Apr. 2012). Optimistic planning for Markov decision processes. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Neil D. Lawrence and Mark Girolami. Vol. 22. Proceedings of Machine Learning Research. La Palma, Canary Islands: PMLR, pp. 182–189.

Buşoniu, Lucian, Előd Páll, and Rémi Munos (2018). Continuous-action planning for discounted infinite-horizon nonlinear optimal control with Lipschitz values. *Automatica* 92, pp. 100–108.

Cappé, Olivier, Aurélien Garivier, Odalric-Ambrym Maillard, Rémi Munos, and Gilles Stoltz (2013). Kullback-Leibler Upper Confidence Bounds for Optimal Sequential Allocation. *The Annals of Statistics* 41.3, pp. 1516–1541.

Carrara, Nicolas, Romain Laroche, Jean-Léon Bouraoui, Tanguy Urvoy, and Olivier Pietquin (2018). Safe transfer learning for dialogue applications. In *International Conference on Statistical Language and Speech Processing (SLSP)*.

List of References

Carrara, Nicolas, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Odalric-Ambrym Maillard, and Olivier Pietquin (Dec. 2019). Budgeted Reinforcement Learning in Continuous State Space. In *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 9299–9309.

Chandramohan, Senthilkumar, Matthieu Geist, and Olivier Pietquin (Sept. 2010). Optimizing spoken dialogue management with fitted value iteration. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, September 26-30, 2010*. Ed. by Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura. Makuhari, Chiba, Japan: ISCA, pp. 86–89.

Charpentier, Arthur and Béatrice Cherrier (June 2019). La valeur de la vie humaine. *Risques* 118, pp. 107–111.

Chaslot, Guillaume, Mark Winands, H. Herik, Jos Uiterwijk, and Bruno Bouzy (Nov. 2008). Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation* 04.3, pp. 343–357.

Chebotarev, S., D. Efimov, T. Raïssi, and A. Zolghadri (2015). Interval Observers for Continuous-Time LPV Systems with L_1/L_2 Performance. *Automatica* 58.8, pp. 82–89.

Chen, Y. F., M. Everett, M. Liu, and J. P. How (Sept. 2017). Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 24-28 Sept*. Vancouver, BC, Canada, pp. 1343–1350.

Chevalier-Boisvert, Maxime, Lucas Willems, and Suman Pal (2018). *Minimalistic Gridworld Environment for OpenAI Gym*. https://github.com/maximecb/gym-minigrid.

Chow, Yinlam, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone (Jan. 2017). Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *J. Mach. Learn. Res.* 18.1, pp. 6070–6120.

Chow, Yinlam, Aviv Tamar, Shie Mannor, and Marco Pavone (Dec. 2015). Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In *Advances in Neural Information Processing Systems 28, Dec 6–10*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Montreal, Canada: Curran Associates, Inc., pp. 1522–1530.

Codevilla, Felipe, Matthias Miiller, Antonio Lopez, Vladlen Koltun, and Alexey Dosovitskiy (May 2018). End-to-End Driving Via Conditional Imitation Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA), 21-25 May*. Brisbane, QLD, Australia, pp. 4693–4700.

Combastel, C. (2013). Stable Interval Observers in BBC for Linear Systems With Time-Varying Input Bounds. *IEEE Transactions on Automatic Control* 58.2, pp. 481–487.

Coquelin, Pierre-Arnaud and Rémi Munos (July 2007). Bandit Algorithms for Tree Search. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, July*. UAI'07. Vancouver, BC, Canada: AUAI Press, pp. 67–74.

Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, et al. (June 2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27-30 June*. Las Vegas, NV, USA, pp. 3213–3223.

Coulom, Rémi (June 2007a). Computing Elo Ratings of Move Patterns in the Game of Go. In *Computer Games Workshop*. Ed. by H. Jaap van den Herik, Mark Winands, Jos Uiterwijk, and Maarten Schadd. Amsterdam, Netherlands.

— (2007b). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Computers and Games*. Ed. by H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 72–83.

Dann, Christoph, Lihong Li, Wei Wei, and Emma Brunskill (June 2019). Policy Certificates: Towards Accountable Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 1507–1516.

Dashkovskiy, S.N., D.V. Efimov, and E.D. Sontag (2011). Input to state stability and allied system properties. *Automation and Remote Control* 72.8, pp. 1579–1614.

De Freitas, Julian, Sam E Anthony, George Alvarez, and Andrea Censi (Jan. 2019). *Doubting Driverless Dilemmas*. preprint.

Dean, Sarah, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu (2018). Regret Bounds for Robust Adaptive Control of the Linear Quadratic Regulator. In *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 4188–4197.

— (Aug. 2019). On the Sample Complexity of the Linear Quadratic Regulator. *Foundations of Computational Mathematics*.

Delage, Erick and Shie Mannor (Jan. 2010). Percentile Optimization for Markov Decision Processes with Parameter Uncertainty. *Oper. Res.* 58.1, pp. 203–213.

Delos, Vincent and Denis Teissandier (Jan. 2015). Minkowski Sum of Polytopes Defined by Their Vertices. *Journal of Applied Mathematics and Physics (JAMP)* 3.1, pp. 62–67.

Dijkstra, E. W. (Dec. 1959). A note on two problems in connexion with graphs. *Numerische Mathematik* 1.1, pp. 269–271.

Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun (Nov. 2017). CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by Sergey Levine, Vincent Vanhoucke, and Ken Goldberg. Vol. 78. Proceedings of Machine Learning Research. Mountain View, California: PMLR, pp. 1–16.

Doyle, J. (1978). Guaranteed margins for LQG regulators. *IEEE Transactions on Automatic Control* 23.4, pp. 756–757.

Drèze, Jacques (1962). L'utilité sociale d'une vie humaine. *Revue Française de Recherche Opérationnelle* 23, pp. 93–118.

Du, Yanzhu, David Hsu, Hanna Kurniawati, Wee Lee, Sylvie Ong, and Shao Png (May 2010). A POMDP Approach to Robot Motion Planning Under Uncertainty. In *Workshop on Solving Real-World POMDP Problems at the Conference on Automated Planning and Scheduling*. Toronto, Canada.

Editions Nationales du Permis de Conduire (2017). *Objectif Code !*

Efimov, D., L.M. Fridman, T. Raïssi, A. Zolghadri, and R. Seydou (2012). Interval Estimation for LPV Systems Applying High Order Sliding Mode Techniques. *Automatica* 48, pp. 2365–2371.

Efimov, D. and T. Raïssi (2016). Design of interval observers for uncertain dynamical systems. *Automation and Remote Control* 77.2, pp. 191–225.

Efimov, D., T. Raïssi, S. Chebotarev, and A. Zolghadri (2013a). Interval State Observer for Nonlinear Time Varying Systems. *Automatica* 49.1, pp. 200–205.

— (2013b). Interval State Observer for Nonlinear Time Varying Systems. *Automatica* 49.1, pp. 200–205.

Efimov, D., T. Raïssi, and A. Zolghadri (2013). Control of nonlinear and LPV systems: interval observer-based framework. *IEEE Trans. Automatic Control* 58.3, pp. 773–782.

Eraqi, Hesham M., Mohamed N. Moustafa, and Jens Honer (Dec. 2017). End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies. In *Machine Learning for Intelligent Transportation Systems Workshop in the 31st Conference on Neural Information Processing Systems (NIPS), December*. Montreal, Canada.

Faradonbeh, Mohamad Kazem Shirani, Ambuj Tewari, and George Michailidis (2020). Optimism-based adaptive regulation of linear-quadratic systems Systems. *IEEE Transactions on Automatic Control* early access.

Farina, L. and S. Rinaldi (2000). *Positive Linear Systems: Theory and Applications*. New York: Wiley.

Faust, Aleksandra, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, et al. (May 2018). PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *Proceedings - IEEE International Conference on Robotics and Automation*. Brisbane, QLD, Australia, pp. 5113–5120.

Feldman, Zohar and Carmel Domshlak (2014). Simple Regret Optimization in Online Planning for Markov Decision Processes. *Journal of Artifial Intelligence Research* 51, pp. 165–205.

Filippi, S., O. Cappé, and A. Garivier (Sept. 2010). Optimism in Reinforcement Learning and Kullback-Leibler Divergence. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing, 29 Sept.–1 Oct*. Allerton, IL, USA, pp. 115–122.

Fisac, Jaime F., Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin (2019). A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems. *IEEE Transactions on Automatic Control* 64.7, pp. 2737–2752.

Foot, Philippa (1967). The Problem of Abortion and the Doctrine of Double Effect. *Oxford Review* 5, pp. 5–15.

Fraichard, T. (July 1993). Dynamic trajectory planning with dynamic constraints: A 'state-time space' approach. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*. Vol. 2. Yokohama, Japan, 1393–1400 vol.2.

Fraichard, Thierry (Mar. 2014). *Will the Driver Seat Ever Be Empty?* Research Report RR-8493. INRIA.

Fridman, Lex, Jack Terwilliger, and Benedikt Jenik (Dec. 2018). DeepTraffic: Crowdsourced Hyperparameter Tuning of Deep Reinforcement Learning Systems for Multi-Agent Dense Traffic Navigation. In *Deep Reinforcement Learning Workshop at NeurIPS 2018*. Montreal, Canada.

Fukushima, Hiroaki, Tae-Hyoung Kim, and Toshiharu Sugie (2007). Adaptive model predictive control for a class of constrained linear systems based on the comparison model. *Automatica* 43.2, pp. 301–308.

Funke, Joseph, Paul Theodosis, Rami Hindiyeh, Ganymed Stanek, Krisada Kritatakirana, Chris Gerdes, et al. (June 2012). Up to the limits: Autonomous Audi TTS. In *2012 IEEE Intelligent Vehicles Symposium*. Alcala de Henares, Spain, pp. 541–547.

Galceran, Enric, Alexander G. Cunningham, Ryan M. Eustice, and Edwin Olson (Aug. 2017). Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots* 41.6, pp. 1367–1382.

García, Javier, Fern, and o Fernández (2015). A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16.42, pp. 1437–1480.

Geibel, Peter and Fritz Wysotzki (July 2005). Risk-Sensitive Reinforcement Learning Applied to Control under Constraints. *Journal of Artificial Intelligence Research* 24.1, pp. 81–108.

Ghavamzadeh, Mohammad, Marek Petrik, and Yinlam Chow (2016). Safe Policy Improvement by Minimizing Robust Baseline Regret. In *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 2298–2306.

Gindele, Tobias, Sebastian Brechtel, and Rudiger Dillmann (2015). Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine* 7.1, pp. 69–79.

Gogoll, Jan and Julian F. Müller (June 2017). Autonomous Cars: In Favor of a Mandatory Ethics Setting. *Science and Engineering Ethics* 23.3, pp. 681–700.

González, D., J. Pérez, V. Milanés, and F. Nashashibi (2016). A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 17.4, pp. 1135–1145.

Gorissen, Bram L., İhsan Yanıkoğlu, and Dick den Hertog (2015). A practical guide to robust optimization. *Omega* 53, pp. 124–137.

Grill, Jean-Bastien, Omar Darwiche Domingues, Pierre Menard, Remi Munos, and Michal Valko (2019). Planning in entropy-regularized Markov decision processes and games. In *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 12404–12413.

Grill, Jean-Bastien, Michal Valko, and Remi Munos (2016). Blazing the trails before beating the path: Sample-efficient Monte-Carlo planning. In *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 4680–4688.

Hart, Peter E., Nils J. Nilsson, and Bertram Raphael (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

Heess, Nicolas, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver (2016). *Learning and Transfer of Modulated Locomotor Controllers*. preprint.

Ho, Jonathan and Stefano Ermon (2016). Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 4565–4573.

Hoel, Carl Johan, Krister Wolff, and Leo Laine (Nov. 2018). Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, pp. 2148–2155.

Homem-de-Mello, Tito and Güzin Bayraksan (2014). Monte Carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science* 19.1, pp. 56–85.

Hostetler, Jesse, Alan Fern, and Tom Dietterich (July 2014). State Aggregation in Monte Carlo Tree Search. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. Québec City, Québec, Canada: AAAI Press, pp. 2446–2452.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT.

Hren, Jean-François and Rémi Munos (2008). Optimistic Planning of Deterministic Systems. In *Recent Advances in Reinforcement Learning*. Ed. by Sertan Girgin, Manuel Loth, Rémi Munos, Philippe Preux, and Daniil Ryabko. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 151–164.

Ibrahimi, Morteza, Adel Javanmard, and Benjamin V. Roy (2012). Efficient Reinforcement Learning for High Dimensional Linear Quadratic Systems. In *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., pp. 2636–2644.

Isele, David, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura (May 2018). Navigating Occluded Intersections with Autonomous Vehicles Using Deep Reinforcement Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD, Australia, pp. 2034–2039.

Iyengar, Garud N. (2005). Robust Dynamic Programming. *Mathematics of Operations Research* 30.2, pp. 257–280.

Janai, Joel, Fatma Güney, Aseem Behl, and Andreas Geiger (2020). Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art. *Foundations and Trends® in Computer Graphics and Vision* 12.1–3, pp. 1–308.

Jaulin, L. (2002). Nonlinear bounded-error state estimation of continuous time systems. *Automatica* 38.2, pp. 1079–1082.

Jin, Chi, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan (July 2020). Provably efficient reinforcement learning with linear function approximation. In *Proceedings of Thirty Third Conference on Learning Theory*. Ed. by Jacob Abernethy and Shivani Agarwal. Vol. 125. Proceedings of Machine Learning Research. Online: PMLR, pp. 2137–2143.

Jonsson, Anders, Emilie Kaufmann, Pierre Ménard, Omar Darwiche Domingues, Edouard Leurent, and Michal Valko (Dec. 2020). Planning in Markov Decision Processes with Gap-Dependent Sample Complexity. In *Advances in Neural Information Processing Systems 33*. Virtual.

Kakade, Sham and John Langford (July 2002). Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 267–274.

Kalman, Rudolph Emil et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering* 82.1, pp. 35–45.

Kammel, Sören, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagzent, et al. (2008). Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge. *Journal of Field Robotics* 25.9, pp. 615–639.

Kapoor, Parv, Anand Balakrishnan, and Jyotirmoy V. Deshmukh (2020). *Model-based Reinforcement Learning from Signal Temporal Logic Specifications*. preprint.

Karaman, Sertac and Emilio Frazzoli (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30.7, pp. 846–894.

Kaufmann, Emilie and Wouter M Koolen (2017). Monte-Carlo Tree Search by Best Arm Identification. In *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et al. Curran Associates, Inc., pp. 4897–4906.

Kaufmann, Emilie, Pierre Ménard, Omar Darwiche Domingues, Anders Jonsson, Edouard Leurent, and Michal Valko (2020). *Adaptive Reward-Free Exploration*. Submitted to ALT 2021, under review.

Kavraki, Lydia E., Petr Švestka, Jean Claude Latombe, and Mark H. Overmars (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12.4, pp. 566–580.

Kearns, Michael J., Yishay Mansour, and Andrew Y. Ng (2002). A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. *Machine Learning* 49.2-3, pp. 193–208.

Kendall, Alex, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, et al. (May 2019). Learning to Drive in a Day. *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8248–8254.

Kesting, Arne, Martin Treiber, and Dirk Helbing (2007). General Lane-Changing Model MOBIL for Car-Following Models. *Transportation Research Record* 1999.1, pp. 86–94.

Khalil, Hassan K. (2002). *Nonlinear Systems*. 3rd. Prentice Hall PTR.

Kieffer, Michel and Eric Walter (Dec. 2004). Guaranteed Nonlinear State Estimator for Cooperative Systems. *Numerical Algorithms* 37.1, pp. 187–198.

Kocsis, Levente and Csaba Szepesvári (2006). Bandit Based Monte-Carlo Planning. In *European Conference on Machine Learning (ECML)*. Ed. by Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 282–293.

Köhler, J., E. Andina, R. Soloperto, M. A. Müller, and F. Allgöwer (Dec. 2019). Linear robust adaptive model predictive control: Computational complexity and conservatism. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. Nice, France, pp. 1383–1388.

Kuderer, Markus, Shilpa Gulati, and Wolfram Burgard (May 2015). Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA, pp. 2641–2646.

Kuefler, A., J. Morton, T. Wheeler, and M. Kochenderfer (June 2017). Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA, pp. 204–211.

L.A., Prashanth and Mohammad Ghavamzadeh (2013). Actor-Critic Algorithms for Risk-Sensitive MDPs. In *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Lake Tahoe, NV/CA, USA: Curran Associates, Inc., pp. 252–260.

Lamiraux, F. and J. -. Lammond (2001). Smooth motion planning for car-like vehicles. *IEEE Transactions on Robotics and Automation* 17.4, pp. 498–501.

Laroche, Romain, Paul Trichelair, and Remi Tachet Des Combes (June 2019). Safe Policy Improvement with Baseline Bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 3652–3661.

Latombe, Jean-Claude (July 1991). A Fast Path Planner for a Car-like Indoor Mobile Robot. In *Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 2*. AAAI'91. Anaheim, California: AAAI Press, pp. 659–665.

Laumond, J. -., P. E. Jacobs, M. Taix, and R. M. Murray (1994). A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation* 10.5, pp. 577–593.

LaValle, S. M. and S. A. Hutchinson (1998). Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation* 14.6, pp. 912–925.

Lavalle, Steven M. (1998). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. Computer Science Department, Iowa State University.

Le, Hoang, Cameron Voloshin, and Yisong Yue (June 2019). Batch Policy Learning under Constraints. In *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 3703–3712.

Lenz, David, Tobias Kessler, and Alois Knoll (June 2016). Tactical cooperative planning for autonomous highway driving using Monte-Carlo Tree Search. In *2016 IEEE Intelligent Vehicles Symposium (IV)*. Gothenburg, Sweden, pp. 447–453.

Lenz, Ian, Ross A. Knepper, and Ashutosh Saxena (July 2015). DeepMPC: Learning Deep Latent Features for Model Predictive Control. In *Robotics: Science and Systems XI, 2015*. Ed. by Lydia E. Kavraki, David Hsu, and Jonas Buchli. Sapienza University of Rome, Rome, Italy.

Leung, Karen, Edward Schmerling, Mo Chen, John Talbot, J. Christian Gerdes, and Marco Pavone (Nov. 2020). On Infusing Reachability-Based Safety Assurance Within Probabilistic Planning Frameworks for Human-Robot Vehicle Interactions. In *Proceedings of the 2018 International Symposium on Experimental Robotics (ISER 2018)*. Ed. by Jing Xiao, Torsten Kröger, and Oussama Khatib. Buenos Aires, Argentina: Springer International Publishing, pp. 561–574.

Leurent, Edouard (2018). *An Environment for Autonomous Driving Decision-Making*. https://github.com/eleurent/highway-env. GitHub repository.

Leurent, Edouard, Yann Blanco, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2018). Approximate Robust Control of Uncertain Dynamical Systems. In *Machine Learning for Intelligent Transportation Systems Workshop at the Thirty-second Conference on Neural Information Processing Systems (NeurIPS 2018)*. Montreal, Canada.

Leurent, Edouard, Denis Efimov, and Odalric-Ambrym Maillard (Dec. 2020a). Robust-Adaptive Control of Linear Systems: beyond Quadratic Costs. In *Advances in Neural Information Processing Systems 33*. Virtual.

— (Dec. 2020b). Robust-Adaptive Interval Predictive Control for Linear Uncertain Systems. In *2020 IEEE 59th Conference on Decision and Control (CDC)*. Jeju Island, Republic of Korea.

Leurent, Edouard, Denis Efimov, Tarek Raissi, and Wilfrid Perruquetti (Dec. 2019). Interval Prediction for Continuous-Time Systems with Parametric Uncertainties. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. Nice, France, pp. 7049–7054.

Leurent, Edouard and Odalric-Ambrym Maillard (Nov. 2020a). Monte-Carlo Graph Search: the Value of Merging Similar States. In *Asian Conference on Machine Learning (ACML 2020)*. Ed. by Sinno Jialin Pan and Masashi Sugiyama. Bangkok, Thailand, pp. 577–592.

— (Sept. 2020b). Practical Open-Loop Optimistic Planning. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet. Würzburg, Germany: Springer International Publishing, pp. 69–85.

Leurent, Edouard and Jean Mercat (Dec. 2019). Social Attention for Autonomous Decision-Making in Dense Traffic. In *Machine Learning for Autonomous Driving Workshop at the Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019)*. Montreal, Canada.

Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel (2016). End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research* 17.39, pp. 1–40.

Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu (2020). *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. preprint.

Li, Lihong, Jason D. Williams, and Suhrid Balakrishnan (Sept. 2009). Reinforcement learning for dialog management using least-squares Policy iteration and fast feature selection. In *Conference of the International Speech Communication Association (InterSpeech 2009)*. Brighton, United Kingdom, pp. 2447–2451.

Li, Minne, Lisheng Wu, Jun WANG, and Haitham Bou Ammar (Dec. 2019). Multi-View Reinforcement Learning. In *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vancouver, BC, Canada: Curran Associates, Inc., pp. 1420–1431.

Liang, Xinle, Yang Liu, Tianjian Chen, Ming Liu, and Qiang Yang (2019). *Federated Transfer Reinforcement Learning for Autonomous Driving*. preprint.

Lin, Patrick (2015). Why Ethics Matters for Autonomous Cars. In *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 69–85.

Liu, Chunming, Xin Xu, and Dewen Hu (2014). Multiobjective Reinforcement Learning: A Comprehensive Overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.3, pp. 385–398.

Liu, Jingchu, Pengfei Hou, Lisen Mu, Yinan Yu, and Chang Huang (2018). *Elements of Effective Deep Reinforcement Learning towards Tactical Driving Decision Making*. preprint.

Liu, Ming-Yu, Thomas Breuel, and Jan Kautz (2017). Unsupervised Image-to-Image Translation Networks. In *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V.

Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et al. Curran Associates, Inc., pp. 700–708.

Lopez, Pablo Alvarez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, et al. (2018). Microscopic Traffic Simulation using SUMO. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Lorenzen, Matthias, Frank Allgöwer, and Mark Cannon (July 2017). Adaptive Model Predictive Control with Robust Constraint Satisfaction. *IFAC-PapersOnLine* 50.1, pp. 3313–3318.

Lu, Xiao-Yun and Sarah K. Spurgeon (Nov. 1997). Robust sliding mode control of uncertain nonlinear systems. *Systems & Control Letters* 32.2, pp. 75–90.

Lu, Xiaonan and Mark Cannon (July 2019). Robust adaptive tube model predictive control. In *2019 American Control Conference (ACC)*. Philadelphia, PA, USA, pp. 3695–3701.

Luenberger, David G. (2013). *Investment science*. Oxford University Press, Incorporated.

Ma, Xiaoteng, Li Xia, Zhengyuan Zhou, Jun Yang, and Qianchuan Zhao (June 2020). DSAC: Distributional Soft Actor Critic for Risk-Sensitive Reinforcement Learning. In *Reinforcement Learning for Real Life Workshop at ICML 2019*. Long Beach, CA, USA.

Maček, Kristijan, Dizan Vasquez, Thierry Fraichard, and Roland Siegwart (Oct. 2008). Safe Vehicle Navigation in Dynamic Urban Scenarios. In *2008 11th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Beijing, China, pp. 482–489.

Marcos, A. and J. Balas (2004). Development of linear-parameter-varying models for aircraft. *J. Guidance, Control, Dynamics* 27.2, pp. 218–228.

Markowitz, Harry M. (1959). *Portfolio Selection: Efficient Diversification of Investments*. Yale University Press.

Mausser, H. and D. Rosen (Apr. 1999). Beyond VaR: from measuring risk to managing risk. In *Proceedings of the IEEE/IAFE 1999 Conference on Computational Intelligence for Financial Engineering (CIFEr)*. New York, NY, USA, pp. 163–178.

Mavrogiannis, Angelos, Rohan Chandra, and Dinesh Manocha (2020). *B-GAP: Behavior-Guided Action Prediction for Autonomous Navigation*. preprint.

Mayne, D.Q., S.V. Raković, R. Findeisen, and F. Allgöwer (2009). Robust output feedback model predictive control of constrained linear systems: Time varying case. *Automatica* 45.9, pp. 2082–2087.

Mayne, D.Q., J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert (2000). Constrained model predictive control: Stability and optimality. *Automatica* 36.6, pp. 789–814.

Mazenc, F. and O. Bernard (2011). Interval observers for linear time-invariant systems with disturbances. *Automatica* 47.1, pp. 140–147.

Mei, Jincheng, Yangchen Pan, Martha White, Amir-massoud Farahmand, and Hengshuai Yao (2020). *Beyond Prioritized Replay: Sampling States in Model-Based RL via Simulated Priorities*. preprint.

Ménard, Pierre, Omar Darwiche Domingues, Anders Jonsson, Emilie Kaufmann, Edouard Leurent, and Michal Valko (July 2020). *Fast active learning for pure exploration in reinforcement learning*. Research Report. DeepMind.

Mercat, Jean, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil (May 2020). Multi-Head Attention for Joint Multi-Modal Vehicle Motion Forecasting. In *IEEE International Conference on Robotics and Automation*. Virtual conference. Paris, France.

Messaoud, Kaouther, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi (June 2019). Non-local Social Pooling for Vehicle Trajectory Prediction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, pp. 975–980.

Michalska, H. and D. Q. Mayne (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control* 38.11, pp. 1623–1633.

Mitsch, Stefan, Khalil Ghorbal, David Vogelbacher, and André Platzer (2017). Formal verification of obstacle avoidance and navigation of ground robots. *The International Journal of Robotics Research* 36.12, pp. 1312–1340.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, et al. (Feb. 2015). Human-level control through deep reinforcement learning. *Nature* 518.7540, pp. 529–533.

Moisan, M., O. Bernard, and J.L. Gouzé (2009). Near optimal interval observers bundle for uncertain bio-reactors. *Automatica* 45.1, pp. 291–295.

Montemerlo, Michael, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, et al. (2008). Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics* 25.9, pp. 569–597.

Moody, J. and M. Saffell (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks* 12.4, pp. 875–889.

Mueller, Matthias, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun (Oct. 2018). Driving Policy Transfer via Modularity and Abstraction. In *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto. Vol. 87. Proceedings of Machine Learning Research. Zürich, Switzerland: PMLR, pp. 1–15.

Munos, Rémi (2011). Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness. In *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Curran Associates, Inc., pp. 783–791.

— (2014). *From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning*. Vol. 7. Foundations and Trends® in Machine Learning, pp. 1–129.

N, Sriram N, Buyu Liu, Francesco Pittaluga, and Manmohan Chandraker (Aug. 2020). SMART: Simultaneous Multi-Agent Recurrent Trajectory Prediction. In *16th European Conference on Computer Vision (ECCV 2020)*. Glasgow, United Kingdom.

Nadjahi, Kimia, Romain Laroche, and Rémi Tachet des Combes (Sept. 2020). Safe Policy Improvement with Soft Baseline Bootstrapping. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet. Würzburg, Germany: Springer International Publishing, pp. 53–68.

Naghshvar, Mohammad, Ahmed K Sadek, and Auke J Wiggers (2018). *Risk-averse Behavior Planning for Autonomous Driving under Uncertainty*. preprint.

Nilim, Arnab and Laurent El Ghaoui (2005). Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research* 53.5, pp. 780–798.

Noothigattu, Ritesh, Snehalkumar (Neil) S. Gaikwad, Edmond Awad, Sohan Dsouza, Iyad Rahwan, Pradeep Ravikumar, et al. (Feb. 2018). A Voting-Based System for Ethical Decision Making. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, (*AAAI-18*). Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. New Orleans, Louisiana, USA: AAAI Press, pp. 1587–1594.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, et al. (2019). *Solving Rubik's Cube with a Robot Hand*. preprint.

Ouyang, Yi, Mukul Gagrani, and Rahul Jain (Oct. 2017). Control of unknown linear systems with Thompson sampling. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing* (*Allerton*). Monticello, IL, USA, pp. 1198–1205.

Paden, Brian, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli (2016). A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles* 1.1, pp. 33–55.

Paxton, Chris, Vasumathi Raman, Gregory D. Hager, and Marin Kobilarov (Sept. 2017). Combining neural networks and tree search for task and motion planning in challenging environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*). Vancouver, BC, Canada, pp. 6059–6066.

Peña, Victor H, Tze Leung Lai, and Qi-Man Shao (2008). *Self-normalized processes: Limit theory and Statistical Applications*. Springer Science & Business Media.

Pietquin, Olivier, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet (2011). Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing* (*TSLP*) 7.3, p. 7.

Pineau, Joelle, Geoff Gordon, and Sebastian Thrun (Aug. 2003). Point-Based Value Iteration: An Anytime Algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. IJCAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc., pp. 1025–1030.

Pivtoraiko, M. and A. Kelly (Sept. 2005). Efficient Constrained Path Planning via Search in State Lattices. In *The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space* (*i-SAIRAS 2005*). Vol. 603. ESA Special Publication. Munich, Germany, p. 33.

Polack, Philip (Oct. 2018). Consistency and stability of hierarchical planning and control systems for autonomous driving. Theses. PSL Research University.

Polack, Philip, Florent Altché, and Brigitte D'Andréa-Novel (June 2017). The Kinematic Bicycle Model : a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles? In *2017 IEEE Intelligent Vehicles Symposium* (*IV*). Los Angeles, CA, USA, pp. 812–818.

Pomerleau, Dean A. (Dec. 1989). ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems 1*. Ed. by D. S. Touretzky. Denver, Colorado, USA: Morgan-Kaufmann, pp. 305–313.

Porta, Josep M., Nikos Vlassis, Matthijs T.J. Spaan, and Pascal Poupart (Dec. 2006). Point-Based Value Iteration for Continuous POMDPs. *Journal of Machine Learning Research* 7, pp. 2329–2367.

Poupart, Pascal, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling (Jan. 2015). Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, pp. 3342–3348.

Pouyanfar, Samira, Muneeb Saleem, Nikhil George, and Shu Ching Chen (June 2019). ROADS: Randomization for obstacle avoidance and driving in simulation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Long Beach, CA, USA, pp. 1267–1276.

Prakash, Aayush, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, et al. (May 2019). Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada, pp. 7249–7255.

Qi, Charles R., Hao Su, Kaichun Mo, and Leonidas J. Guibas (July 2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, pp. 77–85.

Raïssi, T. and D. Efimov (2018). Some recent results on the design and implementation of interval observers for uncertain systems. *Automatisierungstechnik* 66.3, pp. 213–224.

Raïssi, T., D. Efimov, and A. Zolghadri (2012). Interval state estimation for a class of nonlinear systems. *IEEE Trans. Automatic Control* 57.1, pp. 260–265.

Raïssi, T., G. Videau, and A. Zolghadri (2010). Interval observers design for consistency checks of nonlinear continuous-time systems. *Automatica* 46.3, pp. 518–527.

Reeds, J. A. and L. A. Shepp (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific J. Math.* 145.2, pp. 367–393.

Rehder, Eike, Jannik Quehl, and Christoph Stiller (May 2017). Driving Like a Human: Imitation Learning for Path Planning using Convolutional Neural Networks. In *Workshop at IEEE International Conference on Robotics and Automation (ICRA)*. Singapore.

Rehder, Eike, Florian Wirth, Martin Lauer, and Christoph Stiller (May 2018). Pedestrian Prediction by Planning Using Deep Neural Networks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD, Australia, pp. 5903–5908.

Rhinehart, Nicholas, Rowan McAllister, Kris Kitani, and Sergey Levine (Oct. 2019). PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South), pp. 2821–2830.

Rhinehart, Nicholas, Rowan McAllister, and Sergey Levine (May 2020). Deep Imitative Models for Flexible Inference, Planning, and Control. In *International Conference on Learning Representations (ICLR)*. Virtual Conference, formerly Addis Ababa, Ethiopa.

Roijers, Diederik M., Peter Vamplew, Shimon Whiteson, and Richard Dazeley (2013). A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research (JAIR)* 48.

Ros, G., L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez (June 2016). The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, pp. 3234–3243.

Rosolia, Ugo and Francesco Borrelli (Dec. 2019). Sample-Based Learning Model Predictive Control for Linear Uncertain Systems. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. Nice, France, pp. 2702–2707.

Ross, Stéphane, Geoffrey J. Gordon, and J. Andrew Bagnell (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Journal of Machine Learning Research*.

Sadeghian, Amir, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese (June 2019). SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. In *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, pp. 1349–1358.

Sadeghian, Amir, Ferdinand Legros, Maxime Voisin, Ricky Vesel, Alexandre Alahi, and Silvio Savarese (Sept. 2018). CAR-Net: Clairvoyant Attentive Recurrent Network. In *Computer Vision - ECCV 2018 - 15th European Conference*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Vol. 11215. Lecture Notes in Computer Science. Munich, Germany: Springer, pp. 162–180.

Sadigh, Dorsa, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan (June 2016). Planning for autonomous cars that leverage effects on human actions. In *Proceedings of Robotics: Science and Systems*. AnnArbor, Michigan.

Sánchez L., Abraham, René Zapata, and J. Abraham Arenas B. (2002). Motion Planning for Car-Like Robots Using Lazy Probabilistic Roadmap Method. In *MICAI 2002: Advances in Artificial Intelligence*. Ed. by Carlos A. Coello Coello, Alvaro de Albornoz, Luis Enrique Sucar, and Osvaldo Cairó Battistutti. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–10.

Sastry, Shankar, Marc Bodson, and James F. Bartram (1990). Adaptive Control: Stability, Convergence, and Robustness. *The Journal of the Acoustical Society of America* 88.1, pp. 588–589.

Saunders, William, Girish Sastry, Andreas Stuhlmüller, and Owain Evans (July 2018). Trial without Error: Towards Safe Reinforcement Learning via Human Intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, pp. 2067–2069.

Schelling, Thomas C., Martin J. Bailey, and Gary Fromm (Sept. 1968). The life you save may be your own. *Problems in Public Expenditure.* Studies of Government Finance. Pp. 127–162.

Schneider, Jeff G. (1997). Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning. In *Advances in Neural Information Processing Systems 9*. Ed. by M. C. Mozer, M. I. Jordan, and T. Petsche. MIT Press, pp. 1047–1053.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz (July 2015). Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1889–1897.

Shah, Shital, Debadeepta Dey, Chris Lovett, and Ashish Kapoor (2017). *AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles*.

Shalev-Shwartz, Shai, Shaked Shammah, and Amnon Shashua (2016). *Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving*. preprint.

— (2017). *On a Formal Model of Safe and Scalable Self-driving Cars*. preprint.

Shamma, J. and J. Cloutier (1993). Gain-scheduled missile autopilot design using linear parameter-varying transformations. *J. Guidance, Control, Dynamics* 16.2, pp. 256–261.

Shamma, J.S. (2012). Control of Linear Parameter Varying Systems with Applications. In Springer. Chap. An overview of LPV systems, pp. 1–22.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, et al. (Jan. 2016). Mastering the game of Go with deep neural networks and tree search. *Nature* 529.7587, pp. 484–489.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362.6419, pp. 1140–1144.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, et al. (Oct. 2017). Mastering the game of Go without human knowledge. *Nature* 550.7676, pp. 354–359.

Silver, David and Joel Veness (Dec. 2010). Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Vancouver, BC, Canada: Curran Associates, Inc., pp. 2164–2172.

Smith, H.L. (1995). *Monotone Dynamical Systems: An Introduction to the Theory of Competitive and Cooperative Systems*. Vol. 41. Surveys and Monographs. Providence: AMS.

Song, Weilong, Guangming Xiong, and Huiyan Chen (Apr. 2016). Intention-Aware Autonomous Driving Decision-Making in an Uncontrolled Intersection. *Mathematical Problems in Engineering* 2016, p. 1025349.

Sontag, Eduardo D. (2001). The ISS philosophy as a unifying framework for stability-like behavior. In *Nonlinear control in the year 2000, Vol. 2 (Paris)*. Vol. 259. Lecture Notes in Control and Inform. Sci. London: Springer, pp. 443–467.

Stentz, Anthony (May 1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation (ICRA)*. San Diego, CA, USA, 3310–3317 vol.4.

Sun, Liting, Wei Zhan, Ching Yao Chan, and Masayoshi Tomizuka (June 2019). Behavior Planning of Autonomous Cars with Social Perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, pp. 207–213.

Sunberg, Zachary N., Christopher J. Ho, and Mykel J. Kochenderfer (May 2017). The value of inferring the internal state of traffic participants for autonomous freeway driving. In *2017 American Control Conference (ACC)*. Seattle, WA, USA, pp. 3004–3010.

Sutton, Richard S., Doina Precup, and Satinder Singh (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112.1, pp. 181–211.

Szörényi, Balázs, Gunnar Kedenburg, and Remi Munos (2014). Optimistic Planning in Markov Decision Processes Using a Generative Model. In *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., pp. 1035–1043.

Tamar, Aviv, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor (2015). Policy Gradient for Coherent Risk Measures. In *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., pp. 1468–1476.

Tamar, Aviv, Dotan Di Castro, and Shie Mannor (June 2012). Policy Gradients with Variance Related Risk Criteria. In *Proceedings of the 29th International Coference on International Conference on Machine Learning* (*ICML 2012*). ICML'12. Edinburgh, Scotland: Omnipress, pp. 1651–1658.

Tamar, Aviv, Shie Mannor, and Huan Xu (2014). Scaling up robust MDPs using function approximation. In *31st International Conference on Machine Learning, ICML 2014*.

Tan, W. (1997). Applications of Linear Parameter-Varying Control Theory. PhD thesis. Dept. of Mechanical Engineering, University of California at Berkeley.

Tanaskovic, Marko, Lorenzo Fagiano, Roy Smith, and Manfred Morari (2014). Adaptive receding horizon control for constrained MIMO systems. *Automatica* 50.12, pp. 3019–3029.

Tang, Haoran, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, et al. (Dec. 2017). #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et al. Long Beach, CA, USA: Curran Associates, Inc., pp. 2753–2762.

Terry, Justin K., Benjamin Black, Mario Jayakumar, Ananth Hari, Luis Santos, Clemens Dieffendahl, et al. (2020). *PettingZoo: Gym for Multi-Agent Reinforcement Learning*. preprint.

Tessler, Chen, Daniel J. Mankowitz, and Shie Mannor (May 2019). Reward constrained policy optimization. In *7th International Conference on Learning Representations, ICLR 2019*. New Orleans, LA, USA.

Thomas, Philip, Georgios Theocharous, and Mohammad Ghavamzadeh (July 2015). High Confidence Policy Improvement. In *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 2380–2388.

Tirole, Jean and Steven Rendall (2017). *Economics for the Common Good*. Princeton University Press.

Tobin, Josh, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel (Sept. 2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*). Vancouver, BC, Canada, pp. 23–30.

Torossian, Léonard, Aurélien Garivier, and Victor Picheny (Nov. 2019). $\mathcal{X}$-Armed Bandits: Optimizing Quantiles, CVaR and Other Risks. In *Proceedings of The Eleventh Asian Conference on Machine Learning*. Ed. by Wee Sun Lee and Taiji Suzuki. Vol. 101. Proceedings of Machine Learning Research. Nagoya, Japan: PMLR, pp. 252–267.

Trautman, Peter and Andreas Krause (Oct. 2010). Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*). Taipei, Taiwan, pp. 797–803.

Treiber, Martin, Ansgar Hennecke, and Dirk Helbing (2000). Congested traffic states in empirical observations and microscopic simulations. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* 62.2, pp. 1805–1824.

Turchetta, Matteo, Felix Berkenkamp, and Andreas Krause (Dec. 2016). Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Barcelona, Spain: Curran Associates, Inc., pp. 4312–4320.

Ulbrich, Simon and Markus Maurer (Oct. 2013). Probabilistic online POMDP decision making for lane changes in fully automated driving. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. The Hague, Netherlands, pp. 2063–2067.

Undurti, Aditya, Alborz Geramifard, and Jonathan P. How (2011). *Function Approximation for Continuous Constrained MDPs*. Tech. rep. MIT Computer Science and Artificial Intelligence Lab.

Urmson, Chris, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, et al. (2008). Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics* 25.8, pp. 425–466.

Vanderbei, Robert (1996). *Optimal sailing strategies. Statistics and operations research program*. https://vanderbei.princeton.edu/sail/sail.html. (accessed July 22, 2020).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, et al. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et al. Curran Associates, Inc., pp. 5998–6008.

Vemula, Anirudh, Katharina Muelling, and Jean Oh (May 2018). Social Attention: Modeling Attention in Human Crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD, Australia, pp. 4601–4607.

Vezhnevets, Alexander Sasha, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, et al. (Aug. 2017). FeUdal Networks for Hierarchical Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR, pp. 3540–3549.

Vinodh Kumar, E. and Jovitha Jerome (2013). Robust LQR Controller Design for Stabilizing and Trajectory Tracking of Inverted Pendulum. *Procedia Engineering* 64. International Conference on Design and Manufacturing (IConDM2013), pp. 169–178.

Wang, Yizao, Jean-yves Audibert, and Rémi Munos (Dec. 2009). Algorithms for Infinitely Many-Armed Bandits. In *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Vancouver, B.C., Canada: Curran Associates, Inc., pp. 1729–1736.

Watkins, Christopher J. C. H. and Peter Dayan (May 1992). Q-learning. *Machine Learning* 8.3, pp. 279–292.

Weinstein, Ari and Michael L. Littman (June 2012). Bandit-Based Planning and Learning in Continuous-Action Markov Decision Processes. In *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling (ICAPS 2012)*. ICAPS'12. Atibaia, São Paulo, Brazil: AAAI Press, pp. 306–314.

Wiesemann, Wolfram, Daniel Kuhn, and Berç Rustem (2013). Robust Markov Decision Processes. In *Mathematics of Operations Research*.

Wymann, Bernhard, Christos Dimitrakakisy, Andrew Sumnery, and Christophe Guionneauz (2015). *TORCS: The open racing car simulator*.

Xinlei Pan Yurong You, Ziyan Wang and Cewu Lu (Sept. 2017). Virtual to Real Reinforcement Learning for Autonomous Driving. In *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Gabriel Brostow Tae-Kyun Kim Stefanos Zafeiriou and Krystian Mikolajczyk. London, United Kingdom: BMVA Press, pp. 11.1–11.13.

Xu, Huazhe, Yang Gao, Fisher Yu, and Trevor Darrell (July 2017). End-to-end learning of driving models from large-scale video datasets. In *30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, pp. 3530–3538.

Xu, Mengdi, Wenhao Ding, Jiacheng Zhu, Zuxin Liu, Baiming Chen, and Ding Zhao (2020). *Task-Agnostic Online Reinforcement Learning with an Infinite Mixture of Gaussian Processes*. preprint.

Xu, Wenda, Jia Pan, Junqing Wei, and John M. Dolan (June 2014). Motion planning under uncertainty for on-road autonomous driving. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, pp. 2507–2512.

Xu, Wenda, Junqing Wei, John M. Dolan, Huijing Zhao, and Hongbin Zha (May 2012). A real-time motion planner with trajectory optimization for autonomous vehicles. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN, USA, pp. 2061–2067.

Yang, Yaodong, Rasul Tutunov, Phu Sakulwongtana, and Haitham Bou-Ammar (May 2020). α^α-Rank: Practically Scaling α-Rank through Stochastic Optimisation. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*. Ed. by Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith. Auckland, New-Zealand: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1575–1583.

Zhang, Haifeng, Weizhe Chen, Zeren Huang, Minne Li, Yaodong Yang, Weinan Zhang, et al. (Feb. 2020). Bi-level Actor-Critic for Multi-agent Coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. New York, pp. 7325–7332.

Zhang, S., Y. Wu, and H. Ogai (Sept. 2020). Spatial Attention for Autonomous Decision-making in Highway Scene. In *59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. Chiang Mai, Thailand, pp. 1435–1440.

Zhou, Ming, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, et al. (Nov. 2020). SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. In *Conference on Robot Learning (CoRL)*.

Ziebart, Brian D., Andrew L. Maas, Anind K. Dey, and J. Andrew Bagnell (Sept. 2008). Navigate like a Cabbie: Probabilistic Reasoning from Observed Context-Aware Behavior. In *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp)*. New York, NY, USA: Association for Computing Machinery, pp. 322–331.

Ziebart, Brian D., Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, et al. (Oct. 2009). Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, pp. 3931–3936.

Colophon

Translation of the epigraph of Chapter 2, by Edmund Keeley

Hope your road is a long one.
.....
and may you visit many Egyptian cities
to learn and go on learning from their scholars.

Constantine Cavafy, *Ithaka*.

Translation of the epigraph of Parts II and III, by Mary Ann Caws and Nancy Kline

Act as a primitive, foresee as a strategist.

René Char, *Leaves of Hypnos* (72).

Translation of the epigraph of Chapter 3, by Mary Ann Caws and Nancy Kline

Our inheritance is preceded by no testament.
You only fight well for causes you yourself have shaped,
with which you identify–and burn

René Char, *Leaves of Hypnos* (62-63).

Translation of the epigraph of Chapter 6, by William Aggeler

This fire burns our brains so fiercely, we wish to plunge
To the abyss' depths, Heaven or Hell, does it matter?
To the depths of the Unknown to find something new!

Charles Baudelaire, *The Voyage*.

Translation of the epigraph of Chapter 8

Our workers
On the lanes
Slow down

Vinci Autoroutes.
