N° d'ordre : 99

Année 1978

THÈSE

présentée à

l'Université des Sciences et Techniques de Lille I

en vue de l'obtention du titre de

Docteur de l'Université

par

Victor BERWAERTS

Ingénieur Civil U. I. Lv
(Katholieke Universiteit Leuven)
Belgique
Ingénieur Industriel I. G. Lg
(Institut Gramme/Liège)
Belgique

Optimisation Heuristique du cycle de travail d'une perçeuse à commande numérique

Soutenue le 30 novembre 1978 devant la Commission d'Examen

MM	P. VIDAL	Président et rapporteur
	L. POVY R. SOENEN	Examinateurs
	R. BOLLEN T. PEETERS	Invités

AVANT - PROPOS

Nous sommes particulièrement sensible à l'honneur que nous fait Monsieur le Professeur Pierre VIDAL , en acceptant la présidence de notre Jury de Thèse.

Qu'il trouve ici l'expression de toute notre gratitude.

Nous sommes également très reconnaissant à Messieurs les membres du Jury , d'avoir bien voulu siéger à cette commission d'examen. Nous les remercions vivement.

Nous exprimons aussi notre reconnaissance à toutes les personnes qui nous ont prêté leur concours.

SOMMAIRE

****
*****
*

I N T R O D U C T I O N G E N E R A L E

****

Le perçage des trous dans les plaquettes à circuits imprimés est une opération relativement longue et coûteuse car le nombre de trous à percer peut atteindre 1500 à 2000 unités/plaquette .

L'utilisation optimale d'une perceuse à commande numérique nécessite de connaître l'itinéraire optimal reliant tous les points à percer de façon à minimaliser la durée de travail. En plus, les plaquettes étant généralement fabriquées en série, il est utile et commode que la broche de la perceuse se trouve au début et à la fin d'un cycle de perçage à l'origine du système d'axes.

Le problème consiste à trouver un certain itinéraire Γ^* , partant du point d'origine fixe et reliant tous les autres points, tel que le coût (durée) de sa réalisation soit minimal. Dans la littérature un problème analogue est connu sous le nom de "Travelling Salesman Problem" et peut s'énoncer, par exemple de la façon suivante :

Trouver l'itinéraire le moins coûteux pour un voyageur de commerce qui partant d'une ville donnée, doit visiter chacune des villes d'une certaine région, puis retourner à son point de départ.

D'une façon générale le "coût" pour aller d'un point p_i à un point p_j (noté $c_{ij} \geqslant 0$) n'est pas nébessairement le même que celui du parcours inverse :

$$\vec{p_j p_i} \qquad (\bar{c}_{ji} \geqslant 0)$$

Mathématiquement le problème posé peut être défini par une matrice des coûts :

$$[\bar{c}] = \{\bar{c}_{ij}\}$$

Chaque élément $\bar{c}_{ij}$ représente le coût (durée) nécessaire au système de translation de la machine pour effectuer le déplacement $(\vec{p_i}, \vec{p_j})$ $\vec{p_i}, \vec{p_j}$. En introduisant une variable bivalente telle que :

$x_{ij} = 1$ si l'arc $\vec{p_i}, \vec{p_j}$ existe $\forall_{i,j} = 1, 2, \ldots, N$

$x_{ij} = 0$ si l'arc $\vec{p_i}, \vec{p_j}$ n'existe pas $\forall_{i,j} = 1, 2, \ldots, N$

le problème se résout théoriquement en réalisant le programme suivant :

$$[\text{Min}]\, c = \sum_{i=1}^{N} \sum_{j=1}^{N} \bar{c}_{ij} \cdot x_{ij} \,|\, (i,j) \in \Gamma^*$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad j = 1, 2, \ldots, N$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad i = 1, 2, \ldots, N$$

$$u_i - u_j + (N-1)x_{ij} \leqslant N - 2 \qquad (1 \leqslant i \neq j \leqslant N-1)$$

$$x_{ij} = 0 \text{ ou } 1 \qquad \forall_{i,j} = 1, 2, \ldots, N$$

$$u_i \in R \text{ (nombres réels)} \qquad i = 1, 2, \ldots, N$$

(Réf. 8) .

La solution est donc un certain cycle Hamiltonien Γ^* tel que :

$$[\text{Min}]\, c = \sum_{i=1}^{N} (\bar{c}_{ij})(x_{ij}) \,|\, (i,j) \in \Gamma^*$$

Dans le cas du problème qui nous occupe, la matrice des coûts est une matrice symmétrique :

$$\bar{c}_{ij} = \bar{c}_{ji}$$

le nombre d'itinéraires possibles étant donné par la formule :

$$\frac{(N-1)!}{2}$$

et un ou plusieurs de ces parcours fournit la solution optimale. Cette solution optimale au problème posé peut être obtenu théoriquement par des Méthodes de constructions séquentielles de l'itinéraire dont la méthode de la programmation dynamique et la méthode de la Séparation et Evaluation Progressive fournit la solution exacte .

L'utilisation de la programmation dynamique nécessite cependant :

$$(N - 1)(N - 2)2^{N-3}$$

opérations de calcul et exige une capacité de mémoire du calculateur électronique de :

$$(N - 1)2^{N-1}$$

registres (Réf. 30) . Pratiquement cette procédure se limite à des problèmes contenant très peu de points.

La méthode de la Séparation et Evaluation progressive "(Branch and Bound : par exemple l'algorithme de Little)" peut s'appliquer à des problèmes plus importants, mais le temps de calcul nécessaire à l'optimalisation est imprévisible et croît vite avec le volume du problème. Sweeney et ses collaborateurs (Réf. 35) ont montré que le temps de calcul est multiplié par un facteur 30 pour une augmentation de 10 points sur l'itinéraire.

De ce qui précède on peut conclure qu'il est pratiquement impossible d'utiliser les méthodes précitées pour aborder des problèmes dans lesquels le nombre de trous à percer dépasse 50 unités .

En pratique on se contente très souvent de trouver à de tels problèmes une "bonne" solution en utilisant des méthodes heuristiques.

Le but du présent travail est de proposer un algorithme heuristique dont les résultats obtenus sur des multiples cas ont été jugés très satisfaisants.

****
*****
*

CHAPITRE I

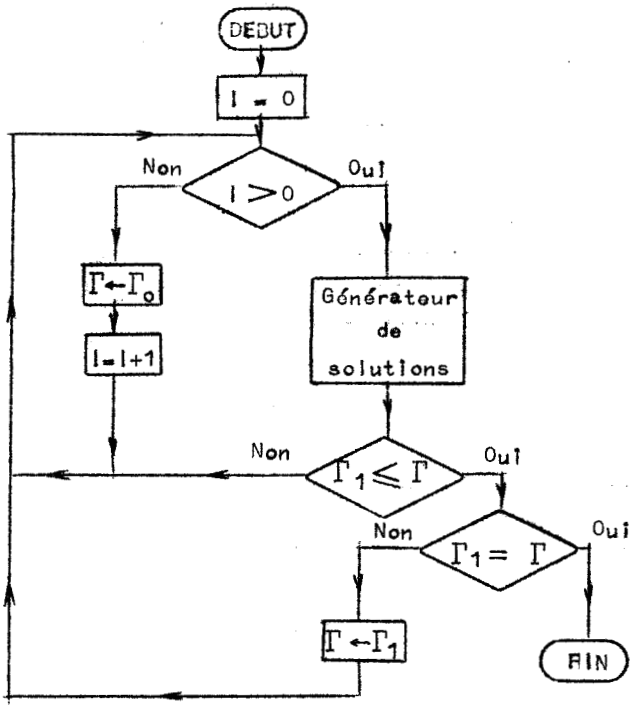
APERCU SOMMAIRE DES METHODES UTILISEES POUR RESOUDRE LE PROBLEME DU VOYAGEUR DE COMMERCE

——— ******* ———

Les méthodes utilisées pour aborder le problème posé peuvent être classées selon la procédure appliquée en quatre catégories.

1.1 METHODES UTILISANT DES CORRECTIONS DE L'ITINERAIRE INITIAL .

Le point de départ de cette procédure est un itinéraire initial arbitraire Γ_0 passant par tous les points prévus. Un générateur de solutions engendre en partant de Γ_0 un nouveau parcours Γ_1 . Ce dernier est comparé à la solution précédente au point de vue des coûts. Le processus de recherche continue aussi longtemps que la comparaison des coûts se révèle bénéfique. La figure 1.1 de la page suivante montre l'organigramme de cette procédure.



Toutes les techniques basées sur cette méthode donnent des résultats approximatifs.

Des bons résultats ont été enregistrés par LIN (Réf. 33) et REITER-SHERMAN (Réf. 37) .

Fig. 1.1

1.2 METHODES UTILISANT DES CONSTRUCTIONS SEQUENTIELLES DE L'ITINERAIRE.

1.2.1 Principe de la méthode.

On choisit un noeud initial arbitraire et à partir de ce sommet une séquence est construite en incluant l'un après l'autre tous les points de l'itinéraire suivant un algorithme donné.

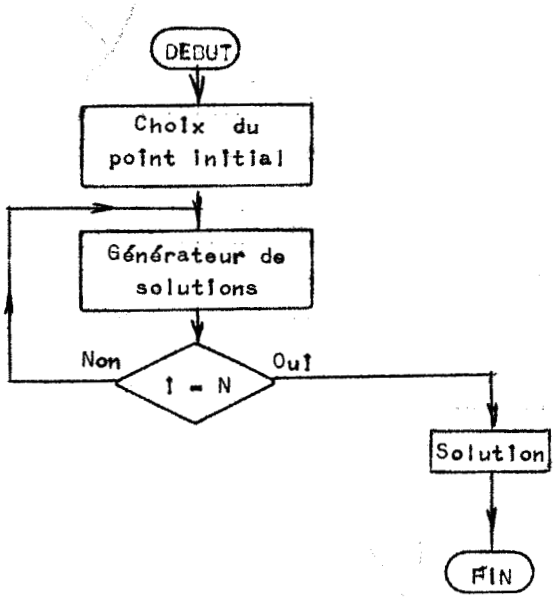


Fig. 1.2

La figure 1.2 ci-contre donne l'organigramme de cette procédure.

Un algorithme simple pour le générateur consiste à utiliser la règle du voisin le plus proche. Les résultats obtenus en adoptant cette règle sont approximatifs. Des résultats très encourageants ont cependant été enregistrés par KARG-THOMPSON (Réf. 31) et par RAYMOND (Réf. 36) .

Des résultats exacts peuvent être obtenus en utilisant la programmation dynamique ou la méthode de la séparation et évaluation progressive, par exemple l'algorithme de Little (Réf. 35) .

Etant donné que nous avons utilisé l'algorithme de Little comme référence lors de nos essais préliminaires nous nous permettons de rappeler quelques détails de cette méthode.

1.2.2 Méthode de Séparation et Evaluation Progressive S.E.P (Réf. 12) .

1.2.2.1 Principe.

Soit :

$$E = \{S_1, S_2, \ldots, S_n\}$$

un ensemble de solutions d'un problème.

Attachons à chaque solution S_i une fonction $f(S_i)$ et cherchons le sous-ensemble E_m des solutions correspondant à la valeur optimal de $f(S_i)$.

On suppose que :

$$E_m \neq \emptyset$$

Soit P_A une propriété permettant de faire une bipartition de E en deux sous-ensembles A et $\bar{A}$ ($\bar{A}$ est le complémentaire de A par rapport à E) et supposons que nous sommes en mesure de trouver une borne inférieure b_o à la valeur des solutions éléments de E , puis une borne inférieure :

$$b_1 \geqslant b_o$$

(dans le cas de la recherche d'un minimum) à la valeur des solutions des éléments de A et enfin une borne inférieure :

$$b'_1 \geqslant 0$$

à la valeur des solutions éléments de $\bar{A}$.

Si nous utilisons maintenant une deuxième propriété séparatrice P_B , la propriété :

$$P_A \cap P_B$$

(P_A et P_B) est alors associée à l'ensemble :

$$A \cap B$$

de même que : $P_A \cap \bar{P}_B$

à : $A \cap \bar{B}$

etc.

En continuant de la sorte c'est-à-dire en introduisant à chaque pas une nouvelle propriété séparatrice, P_C , P_D etc. on définit de nouveaux sous-ensembles. La relation d'inclusion entre ces sous-ensembles définit une arborescence.

Supposons que, au lieu de décrire l'arborescence complète, ce qui nous obligerait à décrire au $n^{\bullet}$ pas 2^n sommets pendants de l'arborescence et ne serait guère intéressante, on se limite à décrire à chaque pas uniquement deux sommets et que la règle de choix pour progresser dans les bipartitions successives soit la suivante : soit E_{pn} l'ensemble des sommets pendant au pas n, un sommet pendant pourra être bipartitionné si et seulement si sa borne inférieure n'est pas supérieure aux bornes inférieures des autres sommets appartenant à E_{pn}. On arrête la dichotomie lorsque le sous-ensemble borné, en respectant la règle donnée, ne contienne plus qu'un seul élément : alors la borne de cet élément ne peut être que sa valeur et cette valeur correspond à une solution minimale de E pour la fonction $f(S_1)$.

1.2.2.2 Description de l'algorithme de Little (Réf. 20).

J.D.C. Little a établi en 1963 un algorithme sur la recherche des circuits hamiltoniens de valeur minimale.

Nous décrirons ci-après brièvement cette procédure :

a) Utiliser la technique de Flood sur la matrice des coûts $[\bar{c}]$, il en résulte une matrice $[c]$ et la borne inférieure (R) de la racine de l'arborescence, racine correspondant à l'ensemble de toutes les solutions appelé E

Remarque

La technique de Flood consiste à exécuter les pas suivants :

- Déterminer le vecteur $[\bar{c}^{1}]$ tel que :

$$[\bar{c}^{1}]^T = [\bar{c}_1^1, \ldots, \bar{c}_m^1]$$

avec : $$\bar{c}_i^1 = \min_j \{\bar{c}_{ij}\}$$

- Déterminer la matrice $[\bar{c}^{(1)}]_{mxn}$ tel que :

$$[\bar{c}^{(1)}] = [\bar{c}_{ij}^{(1)}]$$

avec : $$\bar{c}_{ij}^{(1)} = \bar{c}_i^{(1)}$$

$$\forall_j = 1, 2, \ldots, n$$

$$\forall_i = 1, 2, \ldots, m$$

- Calculer $[c_1] = [\bar{c}] - [\bar{c}^{(1)}]$

- Déterminer le vecteur colonne $[\bar{c}^{(c)}]$ tel que :

$$[\bar{c}^{(c)}]^T = [\bar{c}_1^{(c)}, \ldots, \bar{c}_n^{(c)}]$$

avec : $$\bar{c}_j^{(c)} = \min_i \{c_{1ij}\}$$

- Déterminer la matrice :

$$[c_2]_{mxn} = [\overline{c}_{ij}^{(c)}]$$

avec :

$$\overline{c}_{ij}^{(c)} = \overline{c}_j^{(c)}$$

$$V_i = 1, 2, \ldots, m$$

$$V_j = 1, 2, \ldots, n$$

- Calculer :

$$[c] = [c_1] - [c_2]$$

$$R = \sum_{i=1}^{m} \overline{c}_i^{(c)} + \sum_{j=1}^{n} \overline{c}_j^{(c)}$$

b) Choisir : $c_{x,y} = (c_{ij}^{v} \mid v = \max\{u\})$

pour réaliser une bipartition de telle sorte que :

$$v = \max\{u\}$$

où :

$$u = (\min_{k \neq j}\{c_{ik}\} + \min_{l \neq i}\{c_{lj}\})$$

pour tous les éléments :

$$c_{ij} = 0$$

Ce choix consiste à considérer la propriété $\overline{P}_{ij}$: le circuit hamiltonien ne passe pas par $(i,j = x,y)$. Il doit donc passer, par définition, par les arcs (i,s) , $s \neq j$ et (r,j) , $r \neq i$; pour i et j tels que :

$$c_{ij} = 0$$

en calculant u on trouve un nombre qui ajouté à la borne correspondant au sommet de l'arborescence à partir duquel on effectue la bipartition donne un nouveau minorant plus élevé ou égal au précédent.

c) Mettre en place un sommet et un arc correspondant à la propriété $\overline{P}_{x,y}$. La borne inférieure de ce sommet antérieur augmenté de :

$$v = \max\{u\}$$

d) Mettre en place un sommet et un arc correspondant à la propriété $P_{x,y}$. Les circuits passent par (x,y) . Supprimer dans la matrice $[c]$ la ligne x et la colonne y . Placer une valeur infinie dans toutes les cases correspondant à un arc qui réaliserait un circuit de longueur inférieure au nombre de sommet du graphe initial.

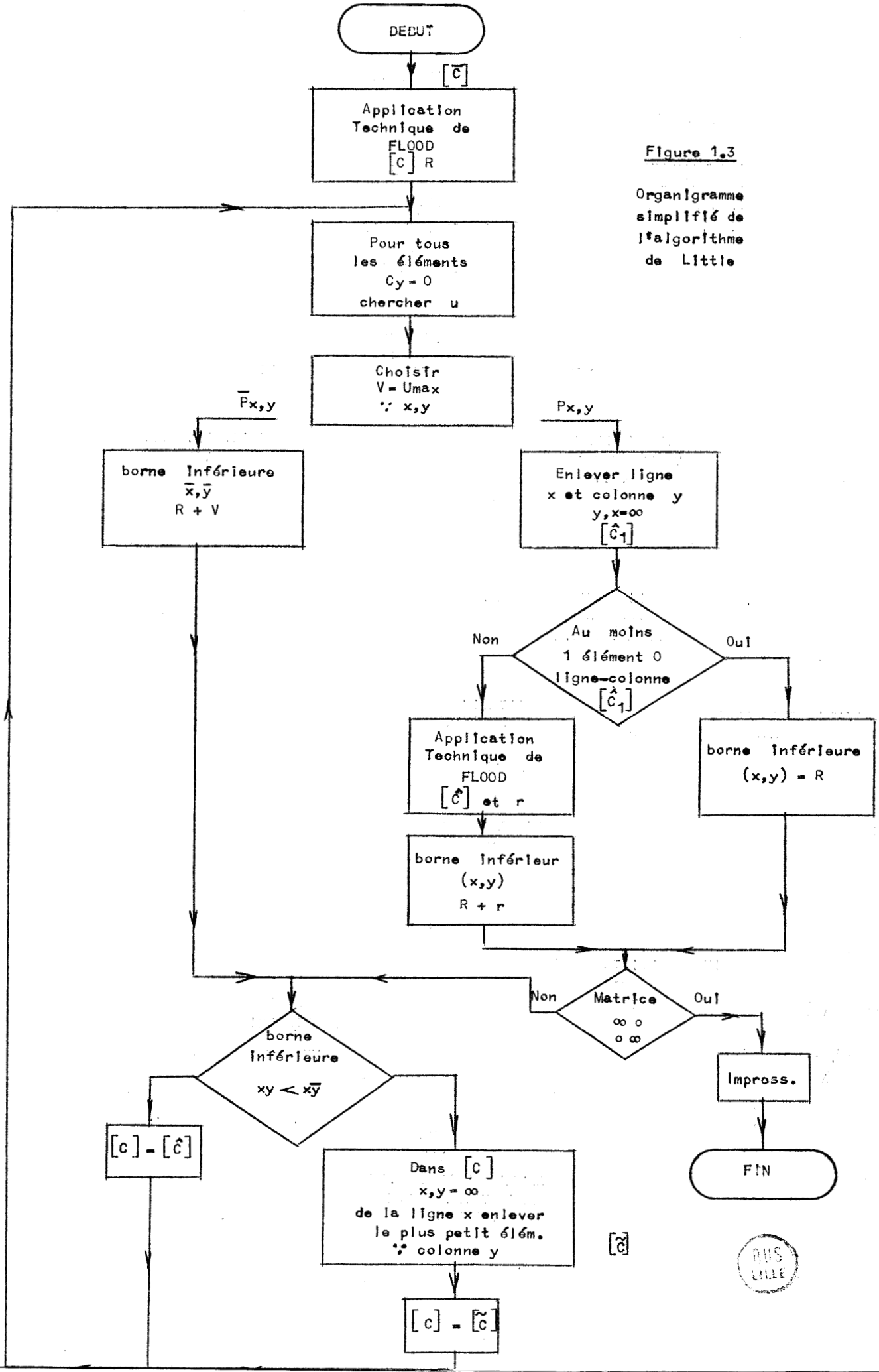


Figure 1.3

Organigramme simplifié de l'algorithme de Little

e) Identique à a)

La somme de tous les éléments soustraits est ajoutée à la borne du sommet antérieur et donne la borne relative au sommet de l'arborescence où la propriété $P_{x,y}$ est vérifiée.

f) Si la matrice obtenue est de la forme :

$$\begin{bmatrix} \infty & 0 \\ 0 & \infty \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} 0 & \infty \\ \infty & 0 \end{bmatrix}$$

les calculs sont terminés sinon on passe en h)

h) Examiner la valeur des bornes obtenues pour tous les sommets pendants et sélectionner la plus petite borne (s'il en existe plusieurs en choisir une arbitrairement) . Si le sommet choisi correspond à (x,y) on passe en b) sinon en i) .

i) Dans la matrice correspondante $[c]_{-}$, mettre ∞ dans la case (x,y) où x,y sont ceux de $P_{x,y}$. Enlever à la ligne x et à la colonne y leurs plus petits éléments respectifs. On trouve ainsi la matrice $[\tilde{c}]$ et passe ensuite en b) .

La Figure 1.3 de la page précédente représente l'organigramme simplifié de l'algorithme de Little. (Le lecteur trouvera en annexe A l'organigramme et le programme complet réalisant la procédure de Little) .

1.3 METHODES ENUMERATIVES.

Comme déjà signalé le nombre d'itinéraires possibles peut devenir vite extrêmement grand :

$$(N - 1)! \quad \text{ou} \quad \frac{(N - 1)!}{2}$$

unités. Il n'est pas possible d'évaluer tous ces itinéraires.

Les méthodes énumératives consistent à éliminer un grand nombre de possibilités en incluant dans le générateur de solutions certaines règles et tests qui permettent, dans certains cas, de juger sur la non-optimalité des solutions rejetées.

L'algorithme de ROBERTS et FLORES (Réf. 38) est basé sur cette technique.

1.4 METHODES UTILISANT L'ELIMINATION DE TOURS ELEMENTAIRES.

Le point de départ de cette méthode est la solution optimale du problème d'affectation relatif à la matrice des coûts $[c]$. Les éléments diagonaux de la matrice étant égal à l'infini (∞) .

Si la solution au problème d'affectation est un contour unique, ce dernier donne également la solution au problème du voyageur de commerce.

Si, par contre, la solution au problème d'affectation n'est pas présentée par un contour unique, elle est alors formée d'un certain nombre de contours élémentaires qu'il s'agit d'éliminer de façon à former un cycle Hamiltonien.

Des solutions exactes ont été obtenues par :

- la méthode de la séparation et évaluation progressive de EASTMAN (Réf. 27) et SHAPIRO (Réf. 39) ,
- la méthode de la programmation linéaire en nombre entiers de DANTZIG , FULKERSON et JOHNSON (Réf. 26) ,
- la méthode de GILMORE-GOMORY (Réf. 29) ; cette dernière procédure utilise une matrice des coûts de structure spéciale .

Signalons pour terminer cet aperçu sommaire que KRUSKAL (Réf. 33) a trouvé une relation entre l'énoncé du problème posé et les arbres minimaux dans un graphe.

****
****
*

CHAPITRE II

P R E S E N T A T I O N

D E

L ' A L G O R I T H M E H E U R I S T I Q U E

****

" Heuristic is a word now commonly in use in this regard but difficult to define well"

(BOWMAN - FETTER) (Réf. 25) .

2.1 GENERALITES SUR LES METHODES HEURISTIQUES.

On entend par heuristique ou méthode heuristique une stratégie, un artifice, une simplification ou une suite de règles, introduites parfois par la seule intuition et l'esprit de création ; permettant de réduire les efforts de recherche des solutions de problèmes complexes en diminuant, par un choix judicieux, le nombre de bonnes solutions pour atteindre ainsi plus vite des solutions pratiques.

Toutefois, ces solutions pratiques ne sont pas nécessairement optimales.

En appliquant une méthode heuristique, l'accent est plutôt mis sur une "bonne solution" peu coûteuse et utilisable que sur une solution optimale. Cette dernière, en effet est dans certains cas difficile à obtenir même en utilisant les technologies modernes.

L'évaluation des techniques heuristiques se fait généralement d'une façon inductive: ces techniques sont appliquées parce-que les expériences montrent qu'elles donnent de très bons résultats pour la résolution de certains problèmes pour lesquels il est difficile d'obtenir à l'heure actuelle des solutions analytiques.

" ..... It (the heuristic approach to solving problem) offers hope where many other analytic approaches, in their current state of development, are not applicable to problems of major intrest to management"

A.A KUEHN (Réf. 32) .

2.2 INTRODUCTION PRELIMINAIRE.

Considérons un ensemble E de N points situés dans une surface plane S .

$$E = \{p_1 , p_2 , \dots , p_N\}$$

Les coordonnées des points sont connues par rapport à la position de repos (origine du système d'axes) de la broche de la machine.

La plaquette à percer étant fixée sur la table de travail de la perceuse possède de ce fait deux degrés de liberté :

- un déplacement suivant l'axe des x ,
- un déplacement suivant l'axe des y .

Pour déterminer le coût d'un déplacement élémentaire considérons deux points arbitrairs p_i et p_j situés dans le plan de travail de la machine et évaluons le temps nécessaire au chariot de commande pour accomplir le trajet $p_i \vec{p}_j$ (Fig. 2.1) (page suivante) .

Sur de nombreuses machines actuelles le déplacement du chariot de travail est obtenu à l'aide de deux moteurs d'entrainement tournant à la même vitesse constante :

$$V_x = V_y = V_A$$

On a donc :

$$V_x = \begin{cases} V_A & \text{pour} \quad x_j - x_i > 0 \\ 0 & \text{pour} \quad x_j - x_i = 0 \end{cases}$$

$$V_y = \begin{cases} V_A & \text{pour} \quad y_j - y_i > 0 \\ 0 & \text{pour} \quad y_j - y_i = 0 \end{cases}$$

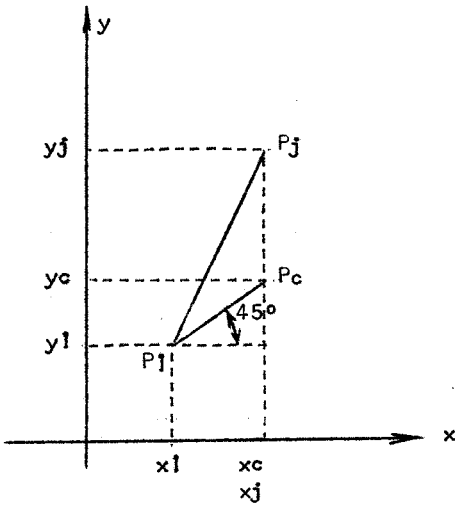


Fig. 2.1

Au début du déplacement,

$$x_j - x_1 > 0$$

$$y_j - y_1 > 0$$

les deux moteurs d'entrainement tournent et :

$$V_x = V_y = V_A$$

Le déplacement résultant de la plaquette se fait dans le sens $\vec{p_1 p_c}$

$$\measuredangle_x \vec{p_1 p_c} = 45°$$

Le temps nécessaire pour réaliser le parcours $\vec{p_1 p_c}$ vaut :

$$T_{\vec{p_1 p_c}} = \frac{x_c - x_1}{V_A} = \frac{y_c - y_1}{V_A}$$

Le point p_c étant atteint, on a :

$$x_j - x_1 = 0$$

le moteur d'entrainement suivant l'axe des x s'arrête.

Avant d'atteindre le point final p_j, il est en plus nécessaire de déplacer la plaquette de la quantité $\vec{p_c p_j}$ c'est-à-dire de :

$$y_j - y_c$$

Le temps correspondant à ce dernier déplacement s'exprime par la formule :

$$T_{\vec{p_c p_j}} = \frac{y_j - y_c}{V_A}$$

La durée totale pour accomplir le trajet prévu vaut donc :

$$T_{\vec{p_1 p_j}} = \frac{y_c - y_1}{V_A} + \frac{y_j - y_c}{V_A} = \frac{y_j - y_1}{V_A}$$

Dans les hypothèses émises la durée nécessaire à la machine pour accomplir $\vec{p_1 p_j}$ est donc proportionnel à :

$$y_j - y_1$$

D'une façon générale et toujours dans les hypothèses du problème on peut conclure que le temps nécessaire pour relier deux points sera proportionnel au plus grand écart :

$$y_j - y_1 \qquad \text{ou} \qquad x_j - x_1$$

Il faudra bien se souvenir de cette définition du coût dans l'interprétation des résultats de certains exemples.

2.3 PRINCIPE DE L'ALGORITHME HEURISTIQUE.

Le but poursuivi consistait à trouver en un temps de calcul raisonnable une "bonne solution" au problème posé et si possible la solution optimale.

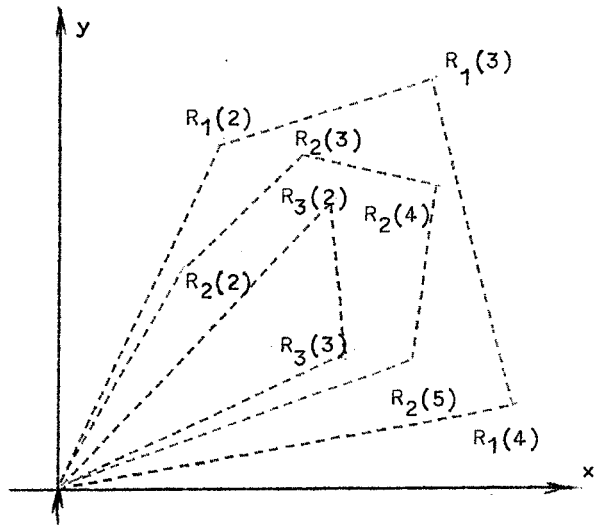
Il s'agit donc de trouver dès le début des opérations de calcul un certain nombre de noeuds (points), situés d'une façon sûre, sur l'itinéraire optimal.

On sait (Réf. 24) que les points formant les extrémités d'un polygône convexe englobant tous les noeuds font partie de l'itinéraire optimal.

De plus, ces points se succèdent sur le parcours optimal dans le même ordre que celui existant sur le polygône.

Le premier pas de l'algorithme proposé consiste donc à trouver les extrémités du polygône convexe global. Pour placer ensuite le plus vite possible les autres noeuds, nous avons groupés les points restants sur des polygônes (convexes) de plus en plus petit en excluant des nouveaux polygônes les points déjà repris sur les polygônes précédents.

Comme les points faisant partie du polygône extérieur font d'une façon sûre partie de l'itinéraire optimal et que les points situés sur le premier polygône intérieur se trouvent le plus près du polygône convexe global, il est logique d'insérer en premier lieu dans le polygône convexe initial les noeuds du premier polygône intérieur. Une fois les extrémités du premier polygône intérieur insérées dans le polygône extérieur, le deuxième polygône intérieur prend la place du premier et l'opération continue jusqu'à épuisement des points.



$R_1(1) = R_2(1) = R_3(1)$ = point de repos de la broche de la perceuse.

Quant à l'insertion des noeuds, deux questions se posent :

- quel noeud insérer ?
- où insérer le noeud ?

Pour répondre à ces deux questions nous nous sommes inspirés des critères avancés par Tc RAYMOND (Réf. 36) (voir aussi le paragraphe suivant).

La figure ci-contre éclaire ce qui a été énoncé ci-dessus.

2.4 FORMALISATION MATHEMATIQUE DE L'ALGORITHME HEURISTIQUE.

1. Générer à travers l'ensemble P des noeuds à relier un certain nombre de graphes initiaux G_1 , G_2 , ...

$$G_1 , G_2 , \ldots \subset F = P \times P$$

$$(p_i , p_j) \varepsilon F , \forall_{p_i, p_j} \varepsilon P$$

$$G_1 \triangleq \{R_1(i) | R_1(i) \varepsilon P \subset E^2 | R_1(i) = \sum_{i=1}^{a} \mu_c R_1(i) ;$$
$$\mu_i \geq 0 ; \forall_i = 1, 2, \ldots , a ; \sum_{i=1}^{a} \mu = 1 \} \subset F$$

$$G_2 \triangleq \{R_2(i) | R_2(i) \varepsilon P - a \subset E^2 | R_2(i) = \sum_{i=1}^{b} \mu_i R_2(i) ;$$
$$\mu_i \geq 0 ; \forall_i = 1, 2, \ldots , b ; \sum_{i=1}^{b} \mu_i = 1 \} \subset F$$

$$G_3 \triangleq \{R_3(i) | R_3(i) \varepsilon P - a - b \subset E^2 | R_3(i) = \sum_{i=1}^{c} \mu_i R_3(i) ;$$
$$\mu_i \geq 0 ; \forall_i = 1, 2, \ldots , c ; \sum_{i=1}^{c} \mu_i = 1 \} \subset F$$

$G_4 \triangleq$ .....

$G_5 \triangleq$ .....

Remarquons que les noeuds $R_1(i)$, $R_2(i)$, $R_3(i)$, $R_4(i)$, ... sont identiques pour :

$$i = 1$$

et se confondent à l'origine des coordonnées dans la surface plane :

$$R_1(1) \equiv R_2(1) \equiv R_3(1) \equiv$$

n'est par conséquent repris qu'une fois dans les graphes, par exemple, dans G_1 .

2. Evaluer les "coûts marginaux" en utilisant la formule :

$$c[R_1(i) , R_2(c)] + c[R_1(i+1) , R_2(c)]$$
$$= c[R_1(i) , R(1 , i+1)] = \lambda^{i, i+1}_{2c}$$
$$\forall_i = 1, 2, \ldots , a$$
$$\forall_c = 1, 2.$$

$$c[R_1(i)\ ,\ R_2(c)]$$

représente le coût (2.2) entre les noeuds :

$$R_1(i) \quad \text{et} \quad R_2(i)$$

3. Choisir parmi les éléments du vecteur :

$$[\lambda_{2c}^{l,l+1}] \quad \forall_l = 1,2, \ldots , a$$

$$\forall_c = 1, 2$$

l'élément le plus petit et l'élément de valeur juste supérieure, c'est-à-dire :

$$\min_1 \lambda_{2c}^{l,l+1} \quad \text{et} \quad \min_2 \lambda_{2c}^{l,l+1} \qquad \forall_l$$

$$\forall_c = 1, 2$$

4. Effectuer la différence :

$$V(c) = \min_2 \lambda_{2c}^{l,l+1} - \min_1 \lambda_{2c}^{l,l+1} \qquad \forall_c = 1, 2$$

5. Comparer : $V(1)$ à $V(2)$

Si : $V(2) > V(1)$

introduire le noeud $R_2(2)$ à l'endroit calculé dans le graphe G_1 , et éliminer ce même noeud dans le graphe G_2 .

Si : $V(1) > V(2)$

l'élément $R_2(1)$ entrera à la bonne place dans le graphe G_1 et sera éliminé du graphe G_2 .

6. Reprendre les opérations (2) , (3) , (4) et (5) jusqu'à épuisement du graphe G_2 .

7. Imprimer le graphe G_1.

8. Calculer le coût de l'itinéraire obtenu G_1.

9. Remplacer le graphe :

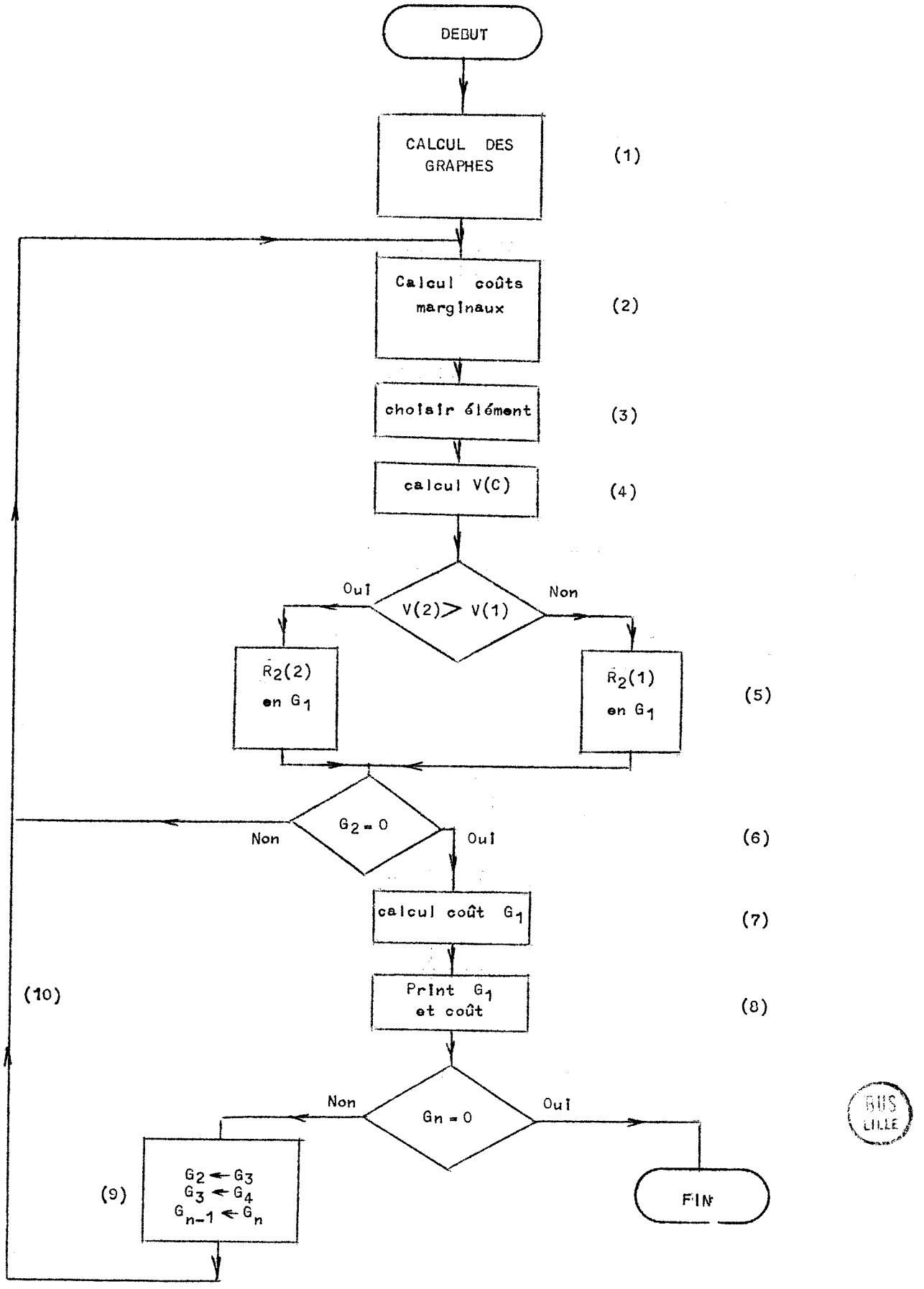
$$G_2 = 0$$

par G_3 , G_3 par G_4 etc.

10. Recommencer le cycle de l'algorithme en (2) et cela jusqu'à épuisement total des graphes.

Le lecteur trouvera en annexe B , l'organigramme et le programme de l'heuristique.

2.5 ORGANIGRAMME SIMPLIFIE OU L'ALGORITHME.



2.6 EVALUATION DU TEMPS DE CALCUL NECESSAIRE POUR OBTENIR L'ITINERAIRE. (Les graphes étant connus) .

Initialement le nombre de points situés sur le graphe G_1 vaut a unités. Afin de compléter G_1 de façon à former un itinéraire il est donc nécessaire d'introduire encore $(N - a)$ points.

Représentons par t le temps élémentaire nécessaire pour évaluer le coût marginal de deux noeuds de G_2 par rapport au segment $R1(i)$, $R1(i+1)$.

Nous pouvons alors établir le tableau suivant :

Opération N°	Nombre de points sur G_1	Nombre de points à insérer	Temps de calcul
1	a	N - a	ta
2	a + 1	N - a - 1	$t_a + t$
3	a + 2	N - a - 2	ta + 2t
⋮	⋮	⋮	⋮
N - a + 1	N	(N - a) - (N - a)	ta + (N - a)t

Le temps de calcul :

$$T = (N - a + 1)ta + t + 2t + \ldots + (N - a)t$$

or :

$$t + 2t + 3t + \ldots + (N - a)t = \frac{(N - a)(N - a + 1)}{2} t$$

et en posant : $\alpha = \frac{a}{N}$

nous obtenons :

$$T = (\frac{1 - \alpha^2}{2})N^2 t + (\frac{1 + \alpha}{2}) Nt$$

ou avec :

$$A = \frac{1 - \alpha^2}{2} t$$

$$B = \frac{1 + \alpha}{2} t$$

$$T = AN^2 + BN = N^{\sigma}$$

avec :

$$c = \frac{\log(AN^2 + BN)}{\log N}$$

Le temps de calcul relatif s'exprime par :

$$\frac{T}{t} = \left(\frac{1-\alpha^2}{2}\right)N^2 + \left(\frac{1+\alpha}{2}\right)N = A_1 N^2 + B_1 N$$

avec :
$$A_1 = \frac{1-\alpha^2}{2}$$

et :
$$B_1 = \frac{1+\alpha}{2}$$

Le lecteur trouvera dans le paragraphe essais industriels la courbe tracée par une formule de la forme :

$$T_c = KN^2 + K_1 N$$

et les résultats de calcul obtenus.

2.7 PARTICULARITES DE LA METHODE.

2.6.1 Le choix initial n'est ni un contour arbitraire complet (voir 1.1) ni un ou deux points (voir 1.2) , mais une suite de noeuds situé sur un graphe orienté.

2.6.2 L'ordre des noeuds situés sur le graphe G_1 :

$$G_1 \triangleq \{ R_1(i) \,|\, R_1(i) \in P \subset E^2 \,|\, R_1(i) = \sum_{i=1}^{a} \mu_i R_1(i) \; ; \; \mu_i \geqslant 0 \; ; \; \forall_i = 1, 2, \ldots, a \; ; \; \sum_{i=1}^{a} \mu_i = 1 \} \subset F$$

se retrouve dans le contour optimal (Réf. 24) .

2.6.3 Etant donné la définition initiale des graphes, la "consommation" des points se fait de la façon la moins coûteuse d'où un temps de calcul réduit.

2.6.4 Les opérations de calcul â effectuer sont extrêmement simples, le temps de calcul â l'ordinateur se trouve ainsi fortement réduit.

2.6.5 Le nombre d'itinéraires possible passe grâce au choix de G_1 de :

$$\left(\frac{N-1}{2}\right)! \quad \text{â} \quad \frac{(N-1)!}{(a-1)!}$$

****
*****
*

CHAPITRE III

E S S A I S E T R E S U L T A T S

****

Les essais ont été groupés en deux catégories :

- les essais préliminaires effectués sur un nombre de noeuds très restreint et réalisés sur l'ordinateur PET COMMODORE en utilisant le programme donné en annexe B .
- les essais industriels calculés sur l'ordinateur HP 21 MX en utilisant le programme complet.

3.1 ESSAIS PRELIMINAIRES.

3.1.1 Commentaires.

Les essais 3.1.2.1 jusque 3.1.2.9 représentent un ensemble de 9 noeuds dont l'itinéraire reliant les sommets à été obtenu à l'aide de l'algorithme heuristique. Les graphiques correspondants montrent les itinéraires obtenus de même que les coûts.

Les matrices des coûts $[\bar{c}]$ correspondantes et permettant d'utiliser la méthode de la séparation et l'évaluation progressive sont également données.

Les résultats obtenus en appliquant l'algorithme de Little sont figurés sous forme de l'arborescence des graphes Es. 1 , ... , Es. 9 et sous forme de graphiques.

Nous constatons que les résultats obtenus avec les deux méthodes concordent parfaitement.

Ce résultat inattendu est probablement dû au nombre très restreint de noeuds à relier - la position des sommets dans le plan étant, en effet, tout à fait arbitraire.

L'identitω des résultats relevée ne peut pas être généralisée car nous savons que l'heuristique utilisé ne garantit en aucun cas l'optimalité de l'itinéraire obtenu.

Comme c'était à prévoir nous avons observé que le temps de calcul

$$T_c$$

nécessaire à l'évaluation de l'itinéraire dépend du facteur :

$$\alpha = \frac{a}{N}$$

et il est nullement fonction de la position des points.

Le graphique de l'exemple 3.1.2.10 représente l'itinéraire obtenu pour un ensemble de 25 points. Nous remarquons qu'il y a des croisements dans l'itinéraire obtenu. A première vue on pourrait conclure à la non-optimalité du résultat relevé, notons que cette conclusion pourrait être trop hâtive. En effet les éléments c_{ij} représentent dans le problème qui nous occupe des durées et non des distances (2.2) .

D'ailleurs le temps nécessaire pour faire l'itinéraire :

$$\vec{n} , \vec{25} , \vec{10} , \vec{18}$$

$$c = 7$$

est exactement le même que celui nécessaire pour parcourir un éventuel autre itinéraire :

$$\vec{11} , \vec{10} , \vec{25} , \vec{18}$$

$$c = 7$$

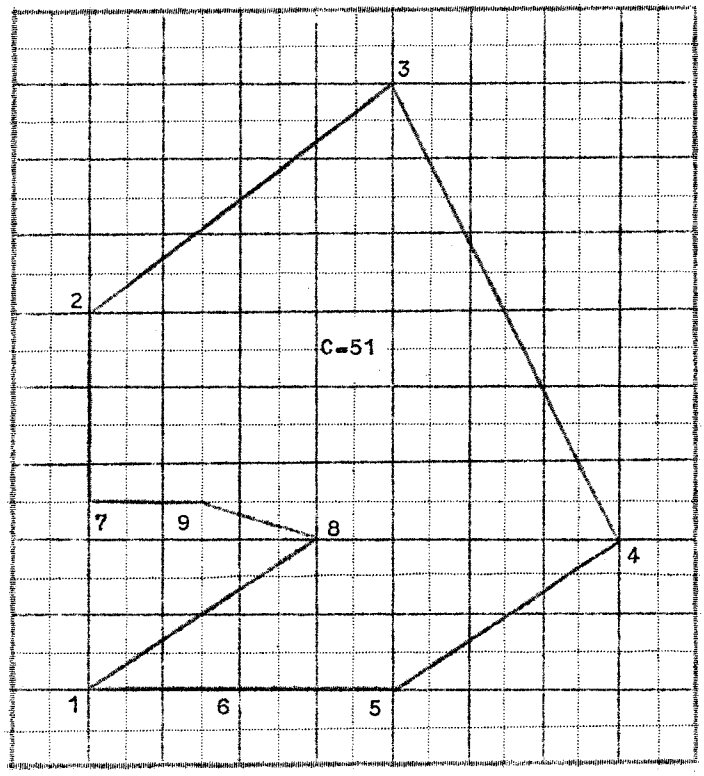
3.1.2 Résultats.

3.1.2.1 Essai 1 .

3.1.2.1.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	0	10
3	8	16
4	14	4
5	8	0
6	4	0
7	0	5
8	6	4
9	3	5



Itinéraire obtenu

1 - 8 - 9 - 7 - 2 - 3 - 4 - 5 - 6 - 1

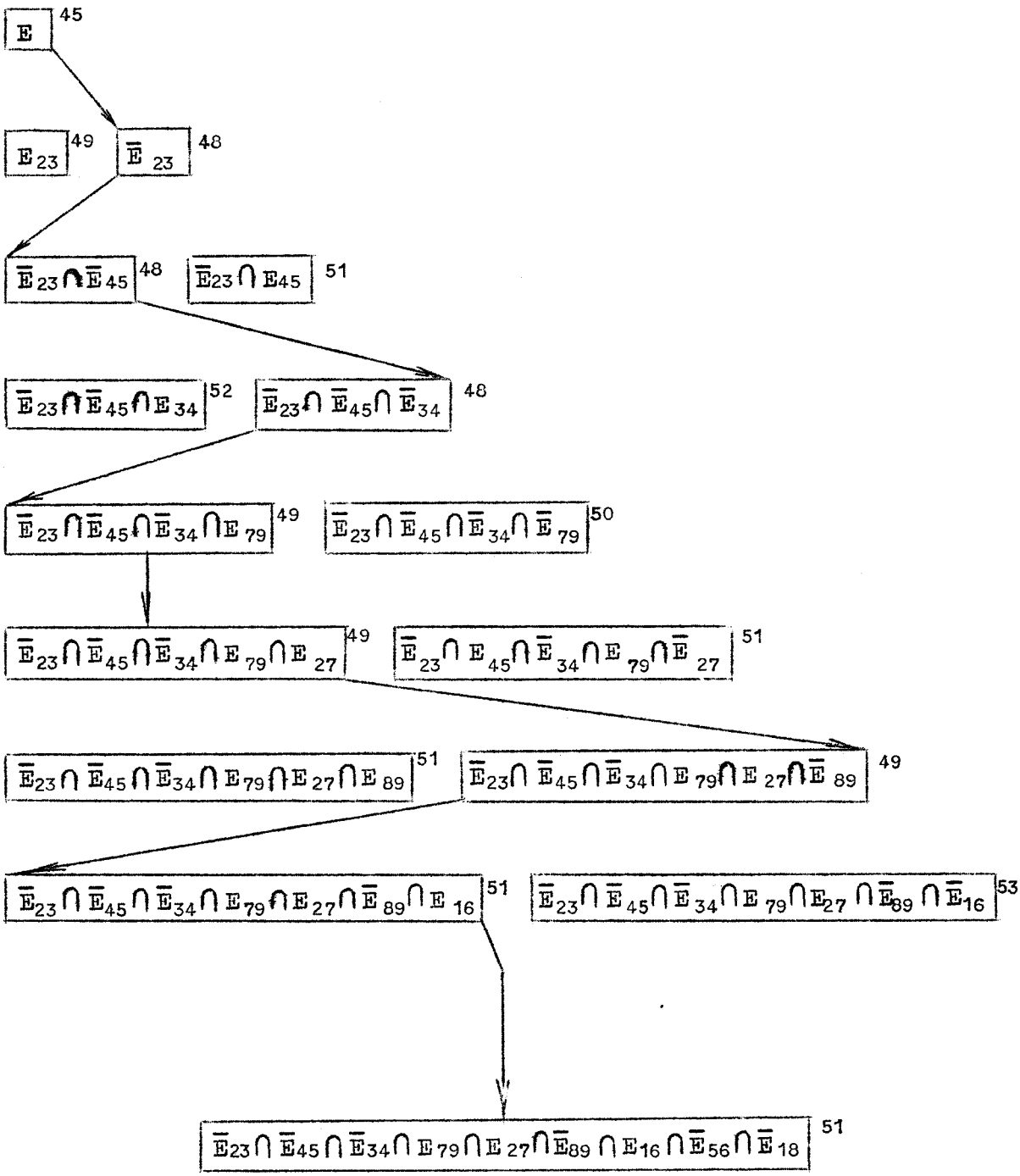
Coût = 51

Matrice des coûts

$$[\overline{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 10 & 16 & 14 & 8 & 4 & 5 & 6 & 5 \\ 10 & \infty & 8 & 14 & 10 & 10 & 5 & 6 & 5 \\ 16 & 8 & \infty & 12 & 16 & 16 & 11 & 12 & 11 \\ 14 & 14 & 12 & \infty & 6 & 10 & 14 & 8 & 11 \\ 8 & 10 & 16 & 6 & \infty & 4 & 8 & 4 & 5 \\ 4 & 10 & 16 & 10 & 4 & \infty & 5 & 4 & 5 \\ 5 & 5 & 11 & 14 & 8 & 5 & \infty & 6 & 3 \\ 6 & 6 & 12 & 8 & 4 & 4 & 6 & \infty & 3 \\ 5 & 5 & 11 & 11 & 5 & 5 & 3 & 3 & \infty \end{bmatrix} \end{array}$$

3.1.2.1.2 Application de l'algorithme de Little.

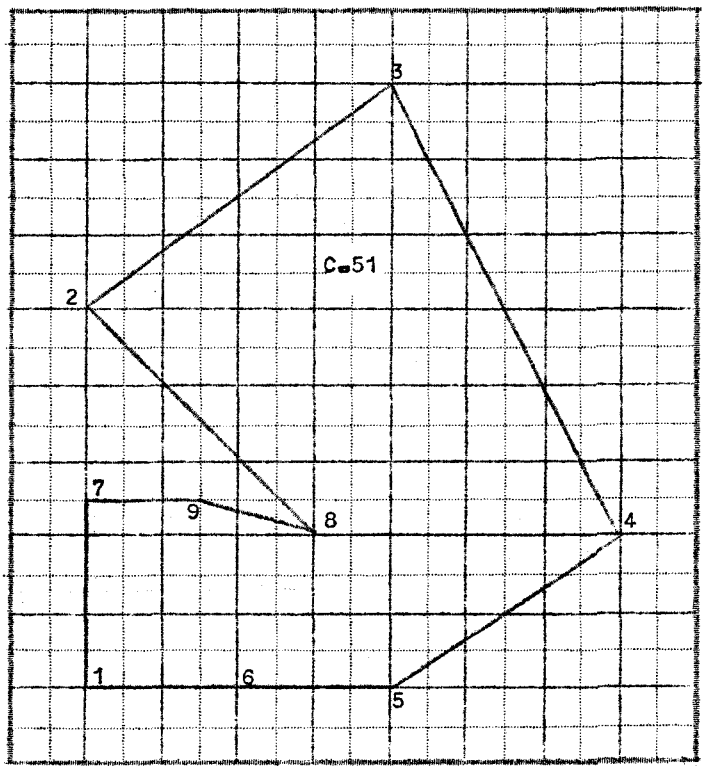
Arborescence du graphe Es. 1

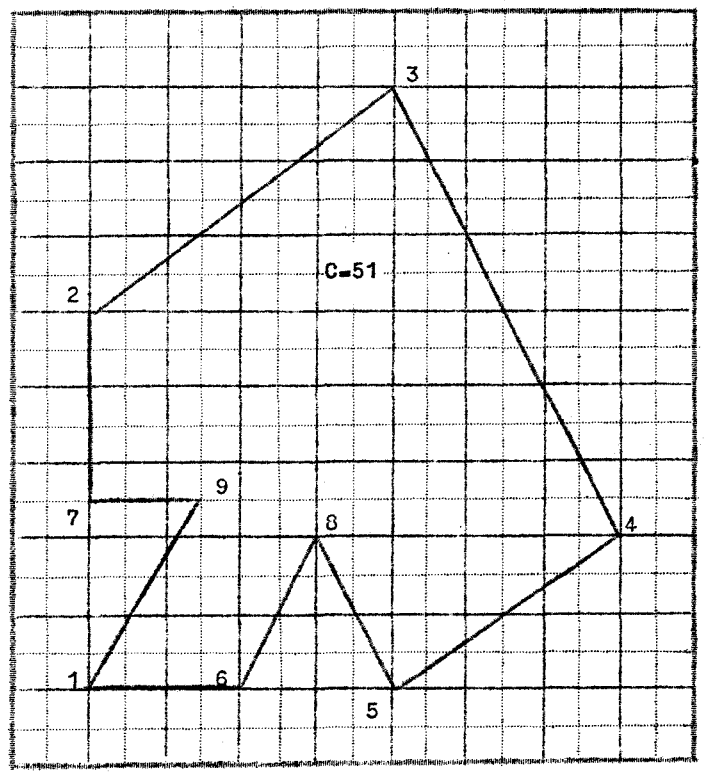


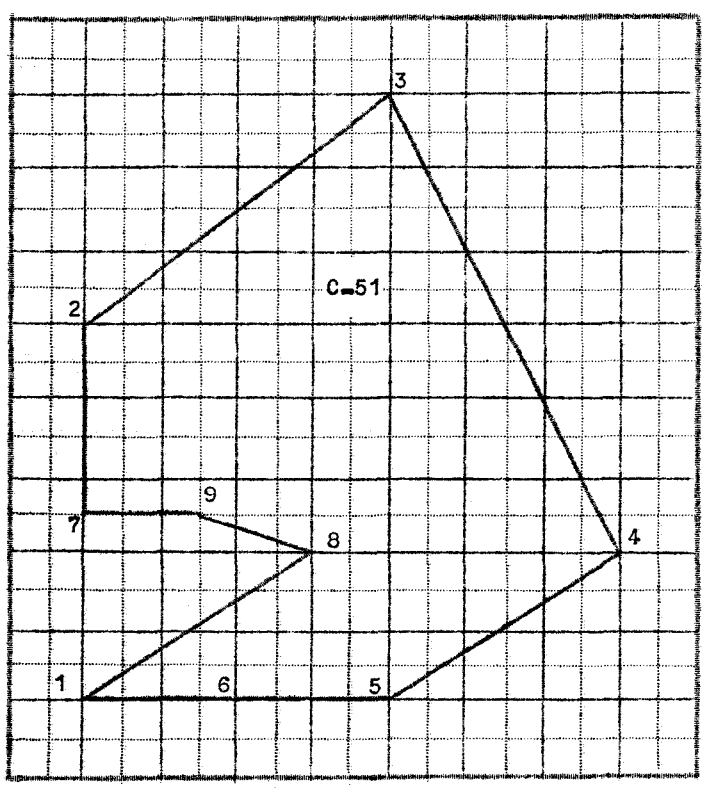
Itinéraire optimal

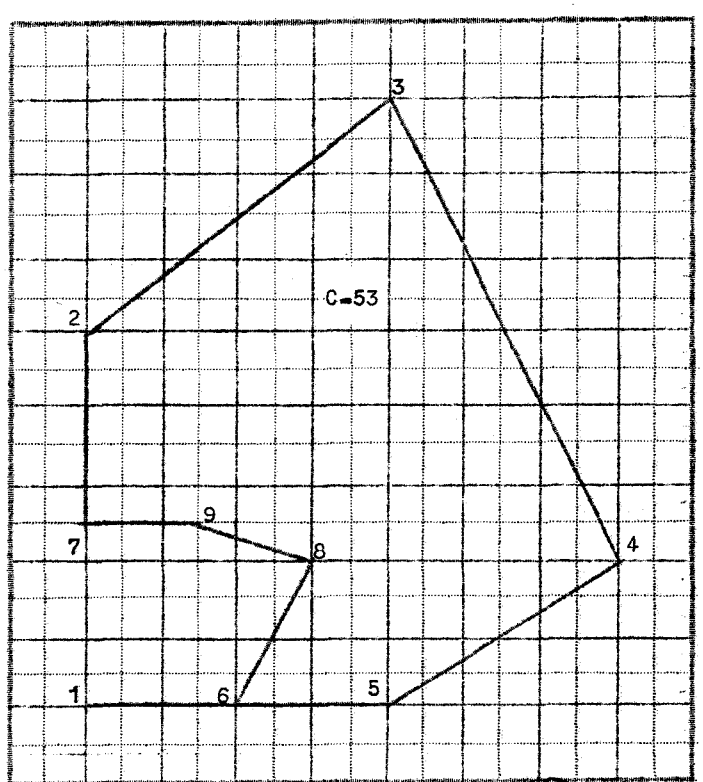
23 - 54 - 43 - 79 - 27 - 89 - 16 - 65 - 81

Coût = 51







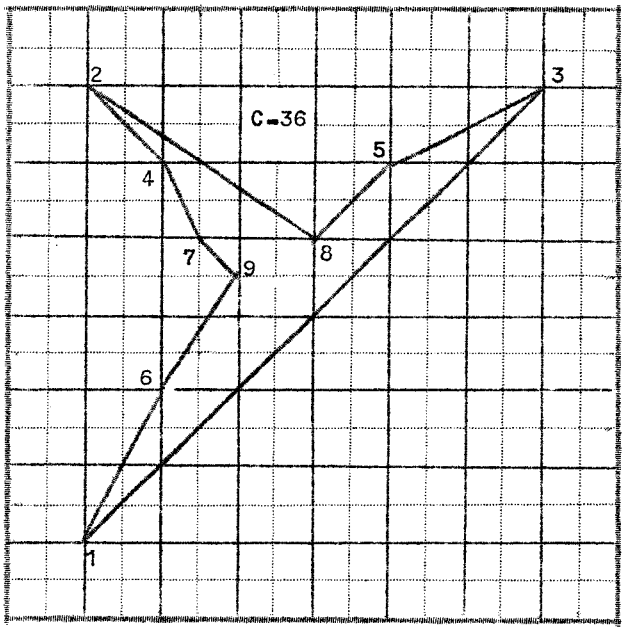


3.1.2.2 Essai 2 .

3.1.2.2.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	0	12
3	12	12
4	2	10
5	8	10
6	2	4
7	3	8
8	6	8
9	4	7



Itinéraire obtenu

1 - 6 - 9 - 7 - 4 - 2 - 8 - 5 - 3

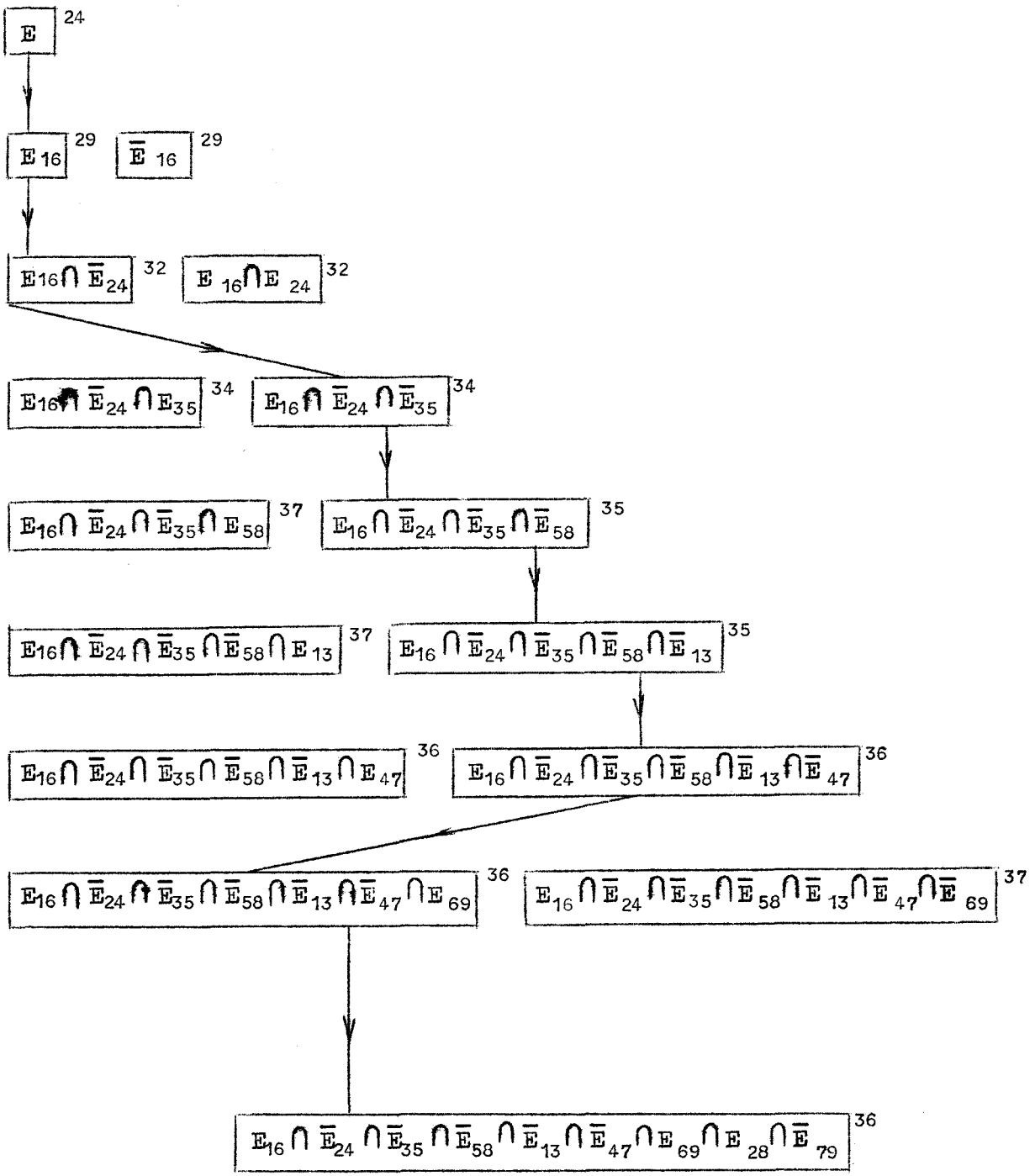
Coût = 36

Matrice des coûts

$$[\overline{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 12 & 12 & 10 & 10 & 4 & 8 & 8 & 7 \\ 12 & \infty & 12 & 2 & 8 & 8 & 4 & 6 & 5 \\ 12 & 12 & \infty & 10 & 4 & 10 & 9 & 6 & 8 \\ 10 & 2 & 10 & \infty & 6 & 6 & 2 & 4 & 3 \\ 10 & 8 & 4 & 6 & \infty & 6 & 5 & 2 & 4 \\ 4 & 8 & 10 & 6 & 6 & \infty & 4 & 4 & 3 \\ 8 & 4 & 9 & 2 & 5 & 4 & \infty & 3 & 1 \\ 8 & 6 & 6 & 4 & 2 & 4 & 3 & \infty & 2 \\ 7 & 5 & 8 & 3 & 4 & 3 & 1 & 2 & \infty \end{bmatrix} \end{array}$$

3.1.2.2.2 Application de l'algorithme de Little.

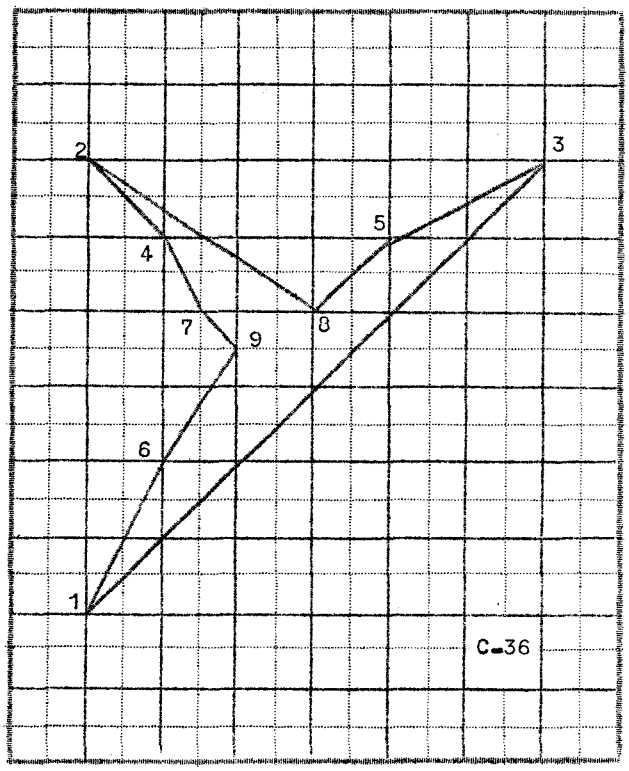
Arborescence du graphe Es. 2

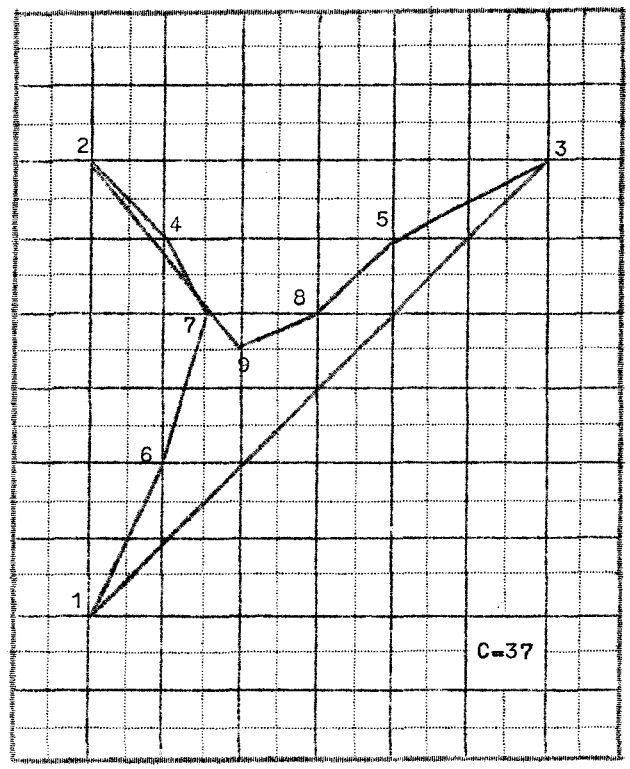


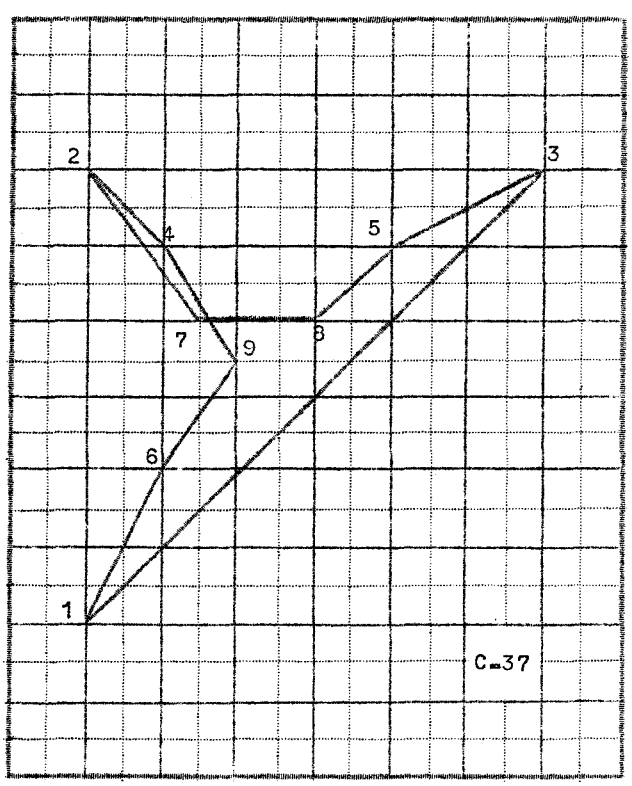
Itinéraire optimal

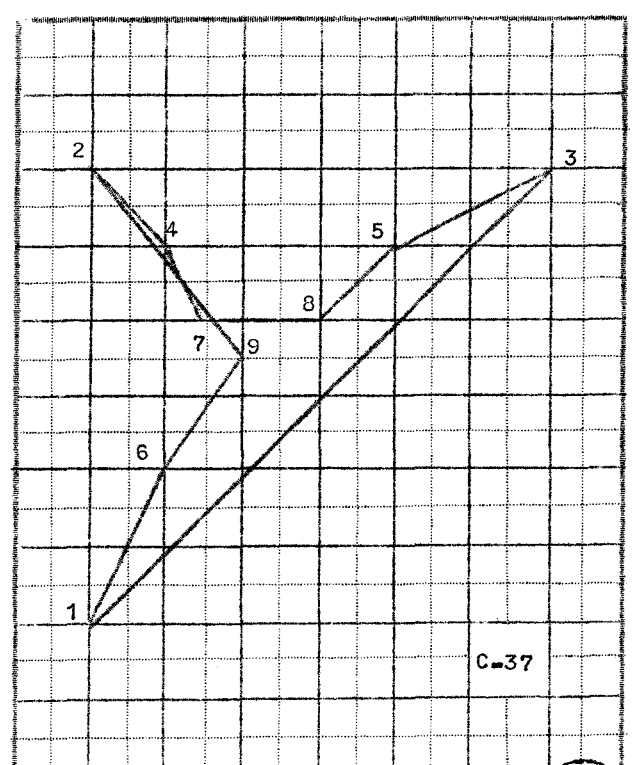
16 - 42 - 53 - 85 - 31 - 74 - 69 - 28 - 97

Coût = 36







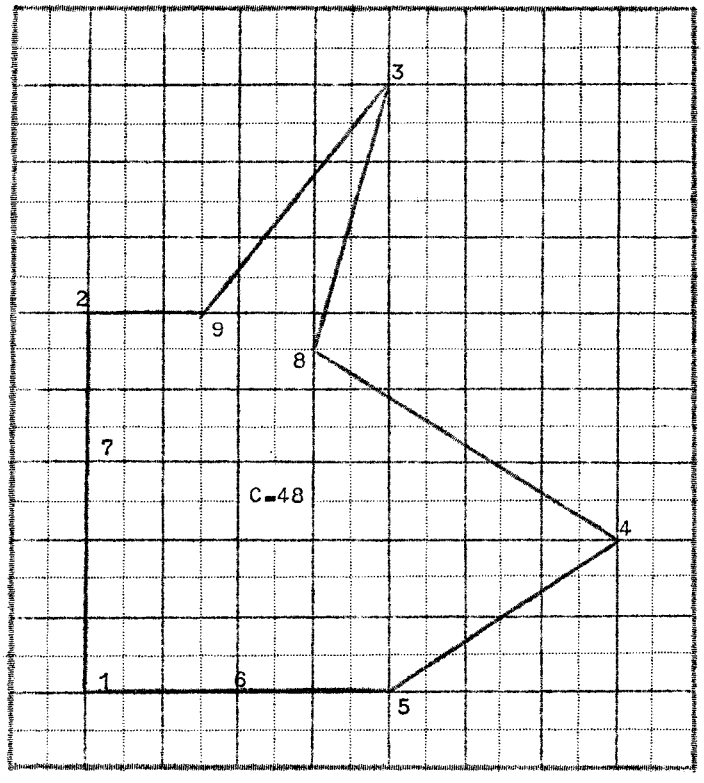


3.1.2.3 Essai 3 .

3.1.2.3.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	0	10
3	8	16
4	14	4
5	8	0
6	4	0
7	0	5
8	6	9
9	3	10



Itinéraire obtenu

1 - 7 - 2 - 9 - 3 - 8 - 2 - 5 - 6 - 1

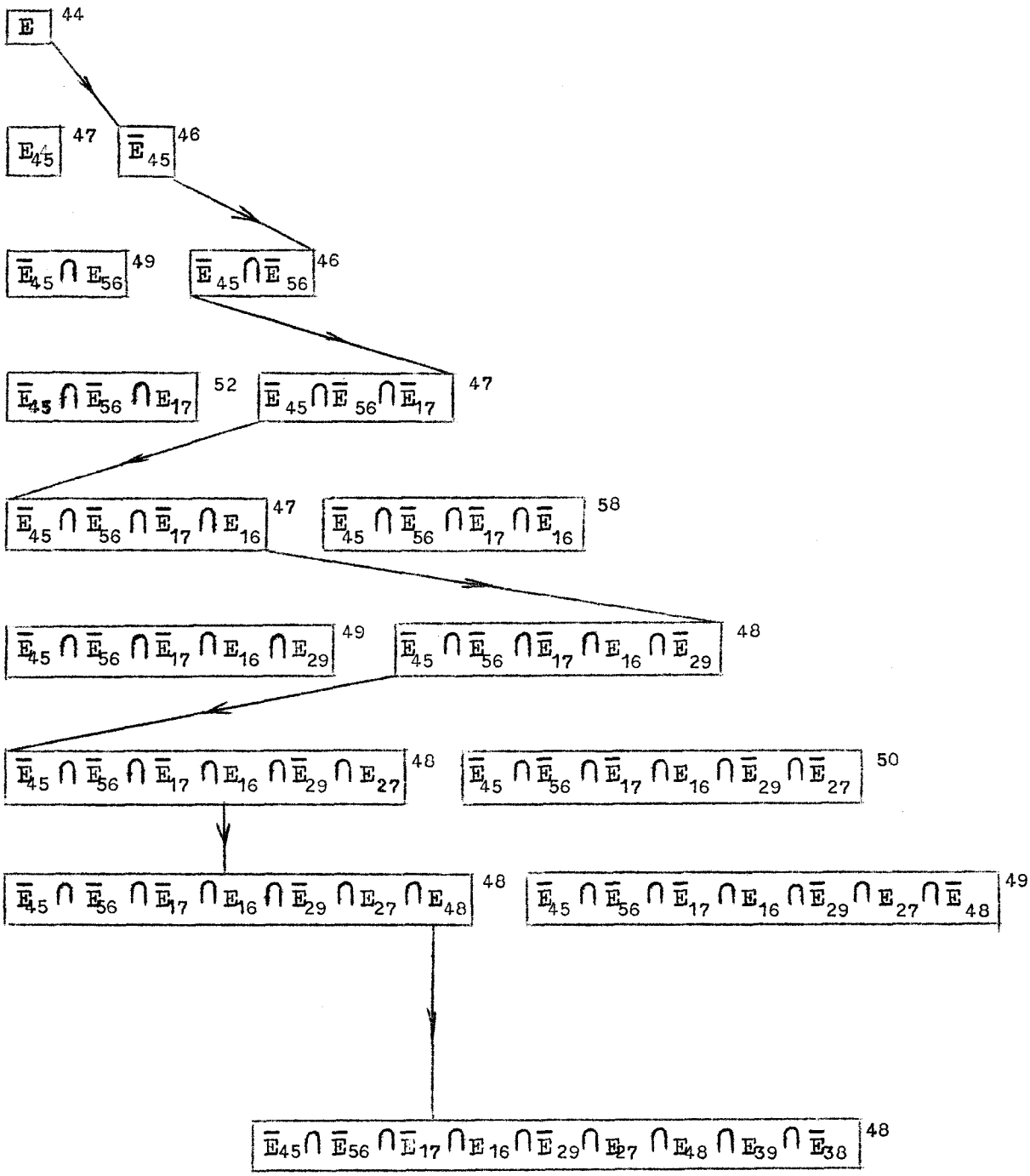
Coût = 48

Matrice des coûts

$$[\bar{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 10 & 16 & 14 & 8 & 4 & 5 & 9 & 10 \\ 10 & \infty & 8 & 14 & 10 & 10 & 5 & 6 & 3 \\ 16 & 8 & \infty & 12 & 16 & 16 & 11 & 7 & 6 \\ 14 & 14 & 12 & \infty & 6 & 10 & 14 & 8 & 11 \\ 8 & 10 & 16 & 6 & \infty & 4 & 8 & 9 & 10 \\ 4 & 10 & 16 & 10 & 4 & \infty & 5 & 9 & 10 \\ 5 & 5 & 11 & 14 & 8 & 5 & \infty & 6 & 5 \\ 9 & 6 & 7 & 8 & 9 & 9 & 6 & \infty & 3 \\ 10 & 3 & 6 & 11 & 10 & 10 & 5 & 3 & \infty \end{bmatrix} \end{array}$$

3.1.2.3.2 Application de l'algorithme de Little.

Arborescence du graphe Es. 3



Itinéraire optimal

54 - 65 - 71 - 16 - 92 - 27 - 48 - 39 - 83

Coût = 48

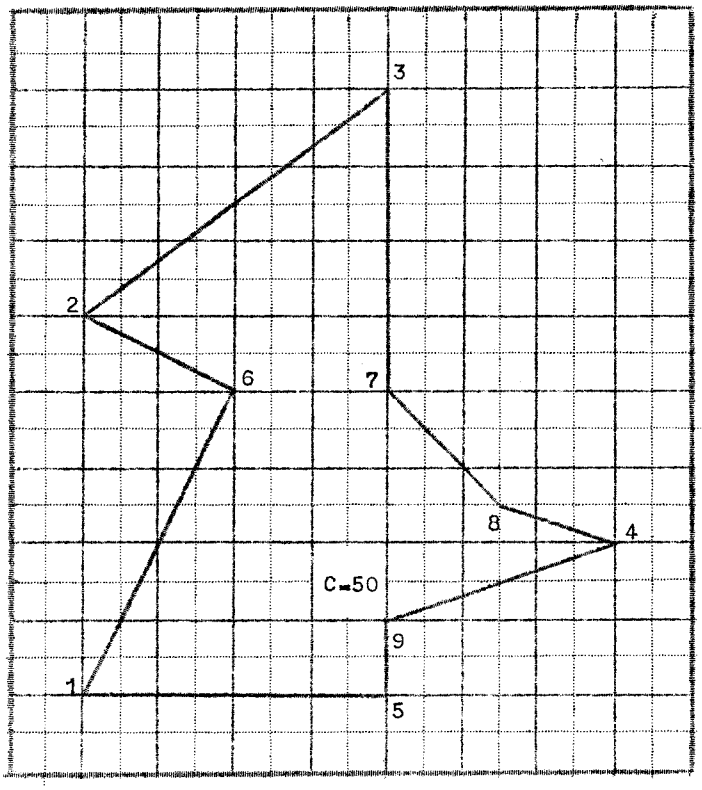
(même figure que la précédente)

3.1.2.4 Essai 4 .

3.1.2.4.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	0	10
3	8	16
4	14	4
5	8	0
6	4	8
7	8	8
8	11	5
9	8	2



Itinéraire obtenu

1 - 6 - 2 - 3 - 7 - 8 - 4 - 9 - 5 - 1

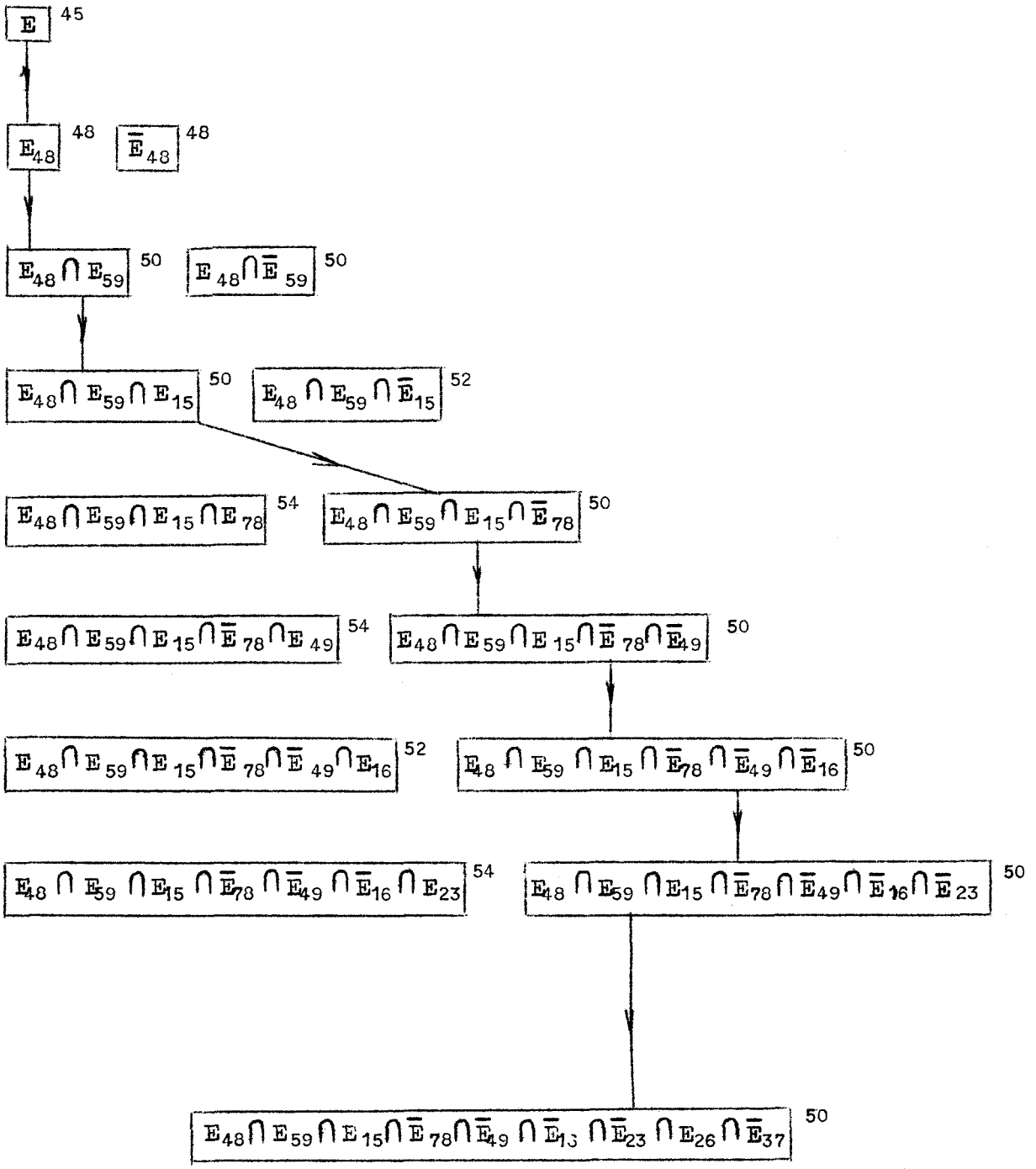
Coût = 50

Matrice des coûts

$$[\bar{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 10 & 16 & 14 & 8 & 8 & 8 & 11 & 8 \\ 10 & \infty & 8 & 14 & 10 & 4 & 8 & 11 & 8 \\ 16 & 8 & \infty & 12 & 16 & 8 & 8 & 11 & 14 \\ 14 & 14 & 12 & \infty & 6 & 10 & 6 & 3 & 6 \\ 8 & 10 & 16 & 6 & \infty & 8 & 8 & 5 & 2 \\ 8 & 4 & 8 & 10 & 8 & \infty & 4 & 7 & 6 \\ 8 & 8 & 8 & 6 & 8 & 4 & \infty & 3 & 6 \\ 11 & 11 & 11 & 3 & 5 & 7 & 3 & \infty & 3 \\ 8 & 8 & 14 & 6 & 2 & 6 & 6 & 3 & \infty \end{bmatrix} \end{array}$$

3.1.2.4.2 Application de l'algorithme de Little.

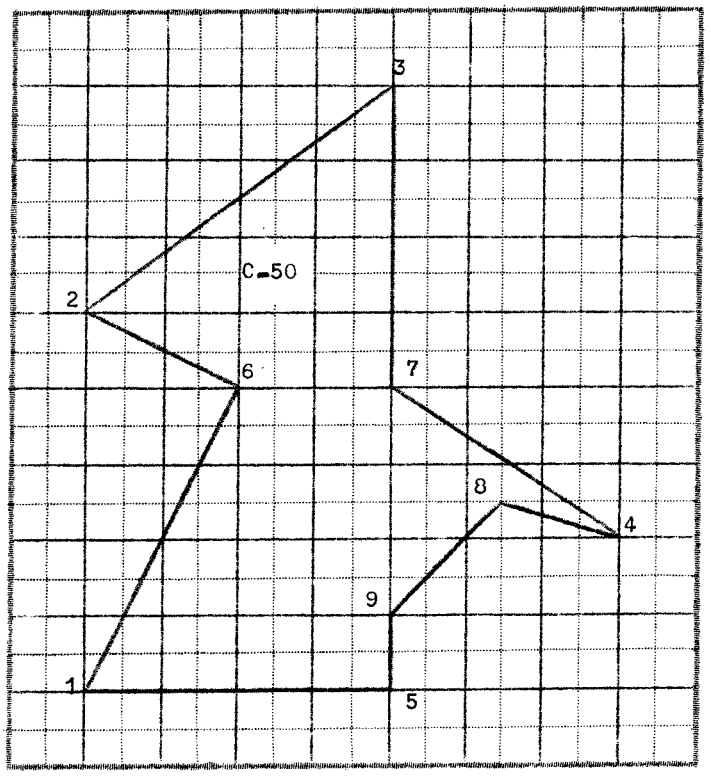
Arborescence du graphe Es. 4

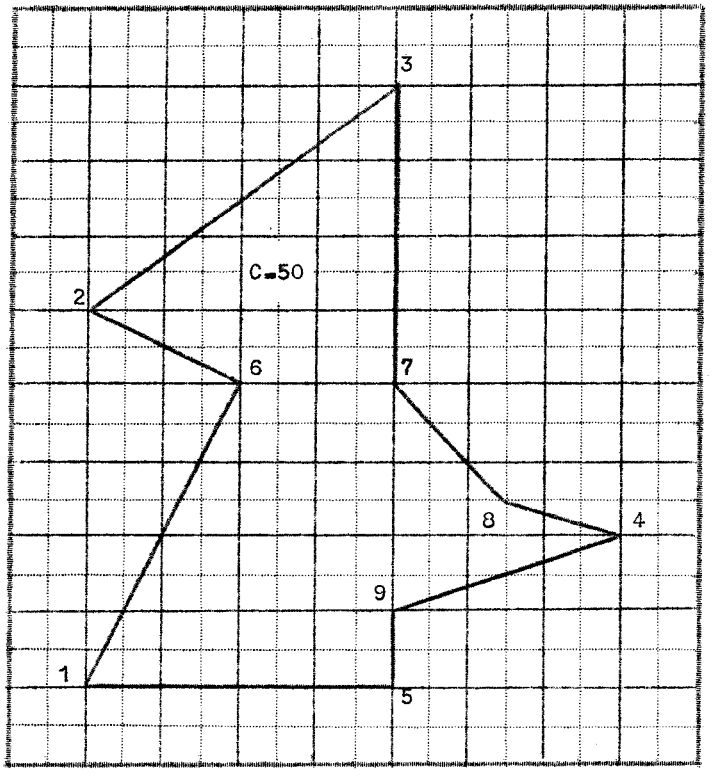


Itinéraire optimal

48 - 59 - 15 - 87 - 94 - 61 - 32 - 26 - 73

Coût = 50



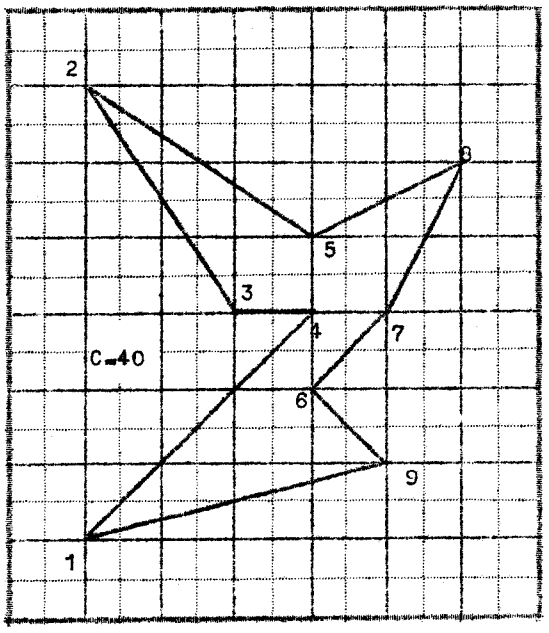


3.1.2.5 Essai 5 .

3.1.2.5.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	0	12
3	4	6
4	6	6
5	6	8
6	6	4
7	8	6
8	10	10
9	8	2



Itinéraire obtenu

1 - 4 - 3 - 2 - 5 - 8 - 7 - 6 - 9 - 1

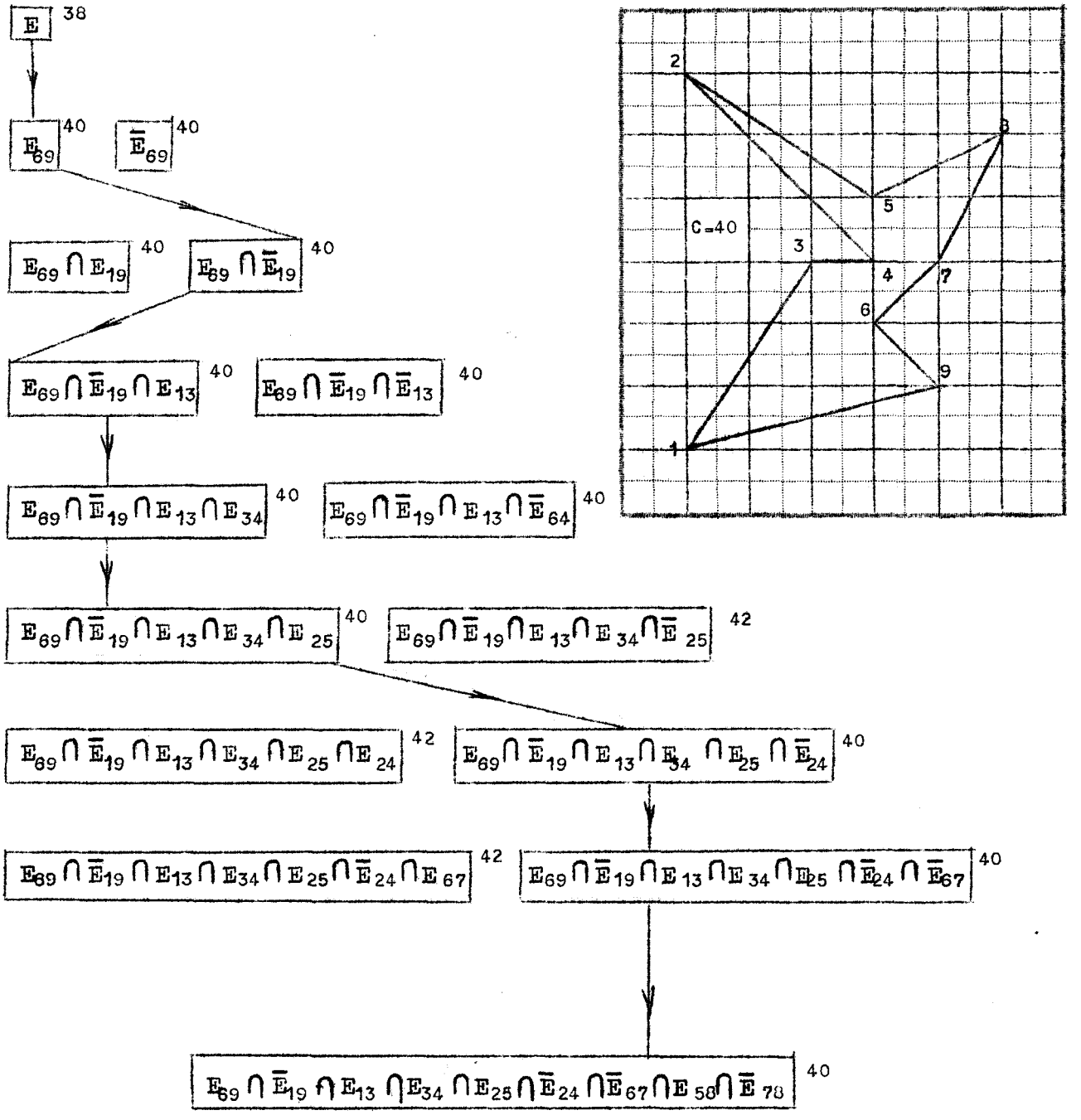
Coût = 40

Matrice des coûts

$$[\bar{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 12 & 6 & 6 & 8 & 6 & 8 & 10 & 8 \\ 12 & \infty & 6 & 6 & 6 & 8 & 8 & 10 & 10 \\ 6 & 6 & \infty & 2 & 2 & 2 & 4 & 6 & 4 \\ 6 & 6 & 2 & \infty & 2 & 2 & 2 & 4 & 4 \\ 8 & 6 & 2 & 2 & \infty & 4 & 2 & 4 & 6 \\ 6 & 8 & 2 & 2 & 4 & \infty & 2 & 6 & 2 \\ 8 & 8 & 4 & 2 & 2 & 2 & \infty & 4 & 4 \\ 10 & 10 & 6 & 4 & 4 & 6 & 4 & \infty & 8 \\ 8 & 10 & 4 & 4 & 6 & 2 & 4 & 8 & \infty \end{bmatrix} \end{array}$$

3.1.2.5.2 Application de l'algorithme de Little.

Arborescence du graphe Es. 5



Itinéraire optimal

69 - 91 - 13 - 34 - 25 - 42 - 76 - 58 - 87

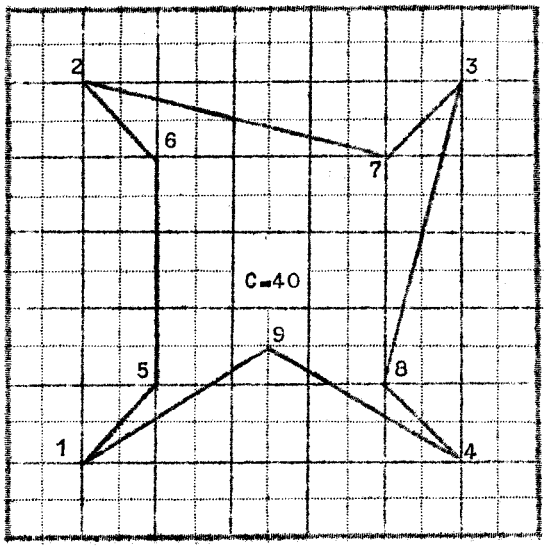
Coût = 40

3.1.2.6 Essai 6 .

3.1.2.6.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	0	10
3	10	10
4	10	0
5	2	2
6	2	8
7	8	8
8	8	2
9	5	3



Itinéraire obtenu

1 - 5 - 6 - 2 - 7 - 3 - 8 - 4 - 9 - 1

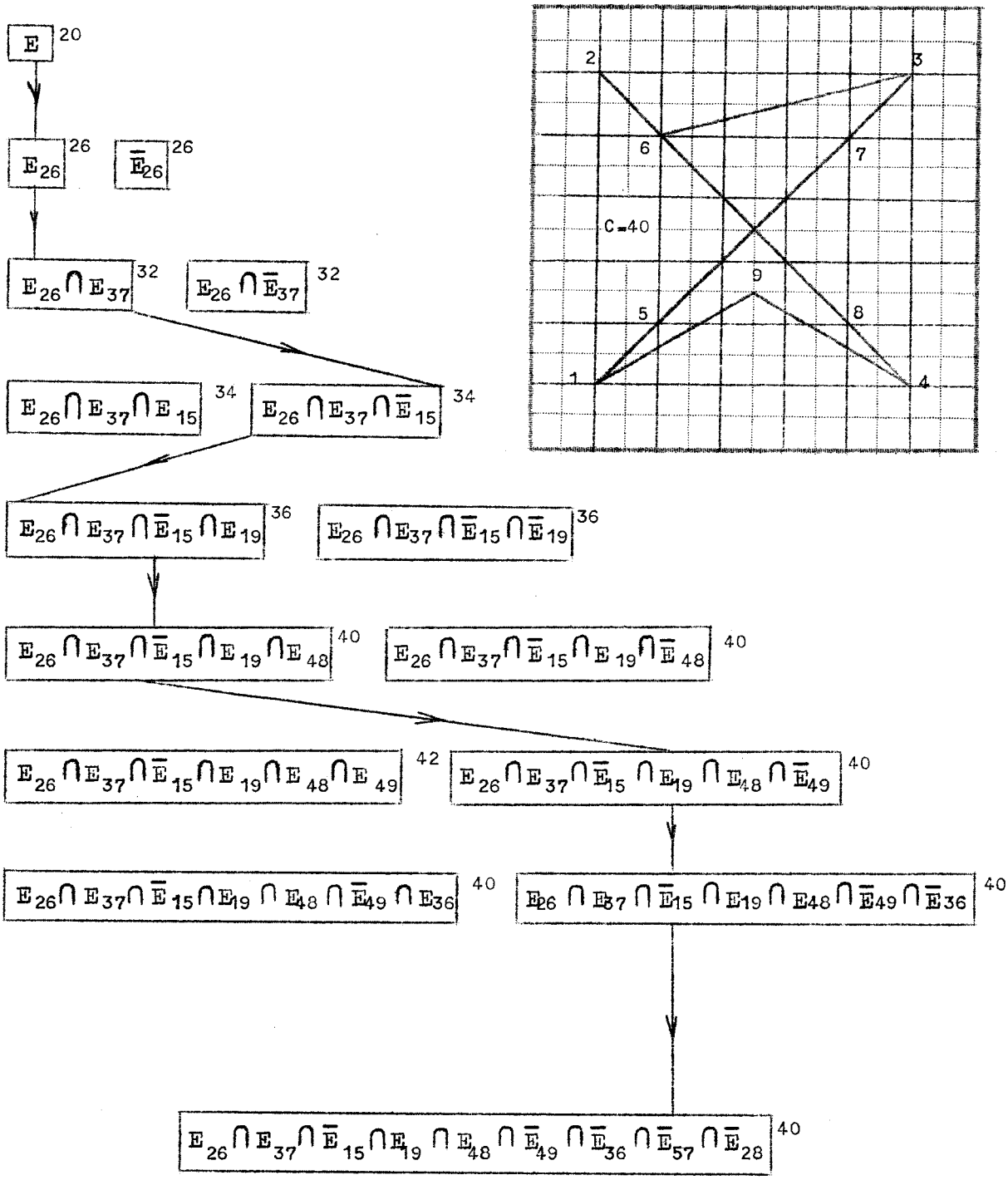
Coût = 40

Matrice des coûts

$$[\bar{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 10 & 10 & 10 & 2 & 8 & 8 & 8 & 5 \\ 10 & \infty & 10 & 10 & 8 & 2 & 8 & 8 & 7 \\ 10 & 10 & \infty & 10 & 8 & 8 & 2 & 8 & 7 \\ 10 & 10 & 10 & \infty & 8 & 8 & 8 & 2 & 5 \\ 2 & 8 & 8 & 8 & \infty & 6 & 6 & 6 & 3 \\ 8 & 2 & 8 & 8 & 6 & \infty & 6 & 6 & 5 \\ 8 & 8 & 2 & 8 & 6 & 6 & \infty & 6 & 5 \\ 8 & 8 & 8 & 2 & 6 & 6 & 6 & \infty & 3 \\ 5 & 7 & 7 & 5 & 3 & 5 & 5 & 3 & \infty \end{bmatrix} \end{array}$$

3.1.2.6.2 Application de l'algorithme de Little.

Arborescence du graphe Es. 6



Itinéraire optimal

26 - 37 - 51 - 19 - 48 - 94 - 63 - 75 - 82

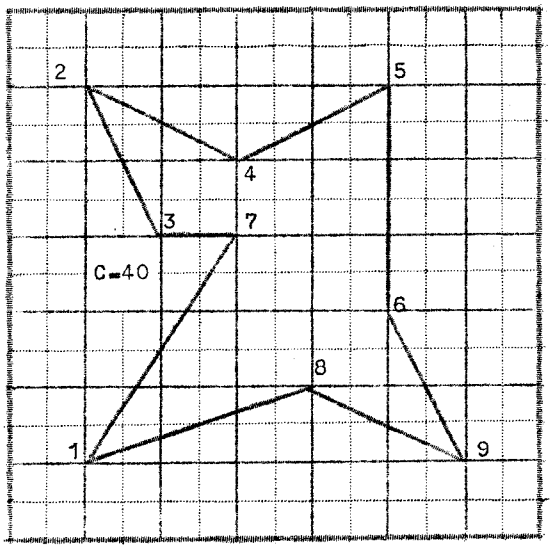
Coût = 40

3.1.2.7 Essai 7 .

3.1.2.7.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	0	10
3	2	6
4	4	8
5	8	10
6	8	4
7	4	6
8	6	2
9	10	0



Itinéraire obtenu

1 - 7 - 3 - 2 - 4 - 5 - 6 - 9 - 8 - 1

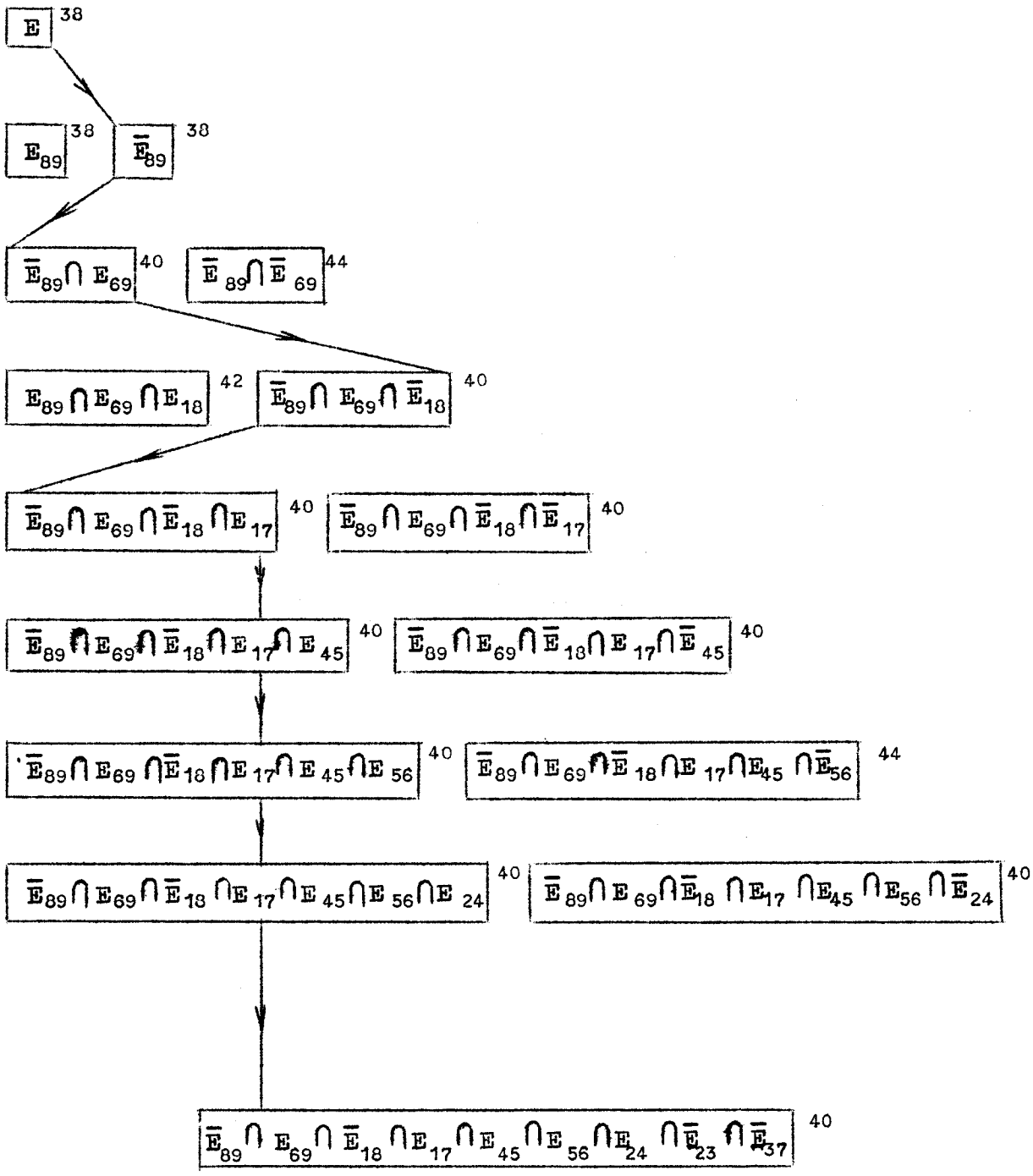
Coût = 40

Matrice des coûts

$$[\overline{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 10 & 6 & 8 & 10 & 8 & 6 & 6 & 10 \\ 10 & \infty & 4 & 4 & 8 & 8 & 4 & 8 & 10 \\ 6 & 4 & \infty & 2 & 6 & 6 & 2 & 4 & 8 \\ 8 & 4 & 2 & \infty & 4 & 4 & 2 & 6 & 8 \\ 10 & 8 & 6 & 4 & \infty & 6 & 4 & 8 & 10 \\ 8 & 8 & 6 & 4 & 6 & \infty & 4 & 2 & 4 \\ 6 & 4 & 2 & 2 & 4 & 4 & \infty & 4 & 6 \\ 6 & 8 & 4 & 6 & 8 & 2 & 4 & \infty & 4 \\ 10 & 10 & 8 & 8 & 10 & 4 & 6 & 4 & \infty \end{bmatrix} \end{array}$$

3.1.2.7.2 Application de l'algorithme de Little.

Arborescence du graphe Es. 7



Itinéraire optimal

98 - 69 - 81 - 17 - 45 - 56 - 24 - 32 - 73

Coût = 40

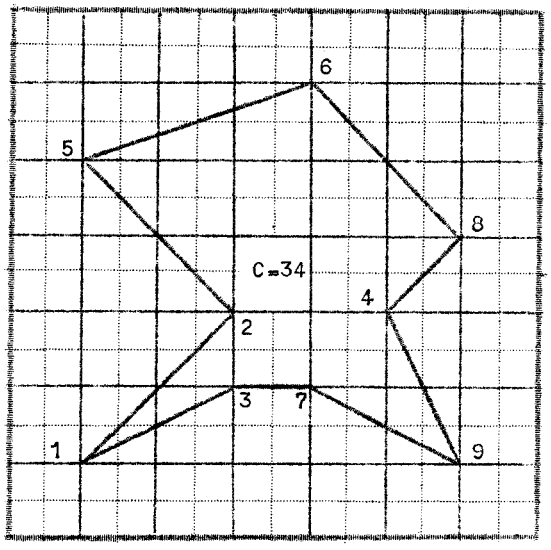
(même figure que la précédente)

3.1.2.8 Essai 8 .

3.1.2.8.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	4	4
3	4	2
4	8	4
5	0	8
6	6	10
7	6	2
8	10	6
9	10	0



Itinéraire obtenu

1 - 2 - 5 - 6 - 8 - 4 - 9 - 7 - 3 - 1

Coût = 34

Matrice des coûts

$$[\bar{C}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 4 & 4 & 8 & 8 & 10 & 6 & 10 & 10 \\ 4 & \infty & 2 & 4 & 4 & 6 & 2 & 6 & 6 \\ 4 & 2 & \infty & 4 & 6 & 8 & 2 & 6 & 6 \\ 8 & 4 & 4 & \infty & 8 & 6 & 2 & 2 & 4 \\ 8 & 4 & 6 & 8 & \infty & 6 & 6 & 10 & 10 \\ 10 & 6 & 8 & 6 & 6 & \infty & 8 & 4 & 10 \\ 6 & 2 & 2 & 2 & 6 & 8 & \infty & 4 & 4 \\ 10 & 6 & 6 & 2 & 10 & 4 & 4 & \infty & 6 \\ 10 & 6 & 6 & 4 & 10 & 10 & 4 & 6 & \infty \end{bmatrix} \end{array}$$

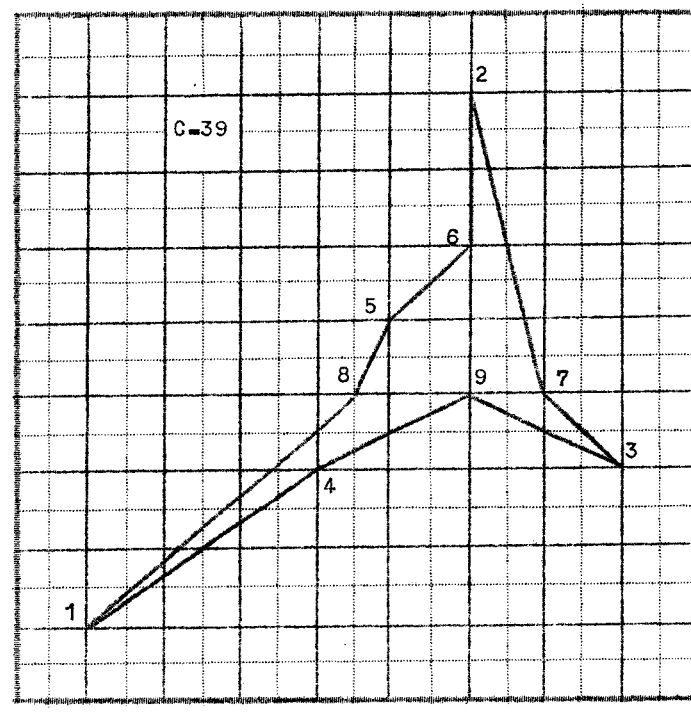
La borne inférieure = 34 . Le circuit obtenu est optimal.

3.1.2.9 Essai 9 .

3.1.2.9.1 Application de l'algorithme heuristique.

Coordonnées des points

1	0	0
2	10	14
3	14	4
4	6	4
5	8	8
6	10	10
7	12	6
8	7	6
9	10	6



Itinéraire obtenu

1 - 8 - 5 - 6 - 2 - 7 - 3 - 9 - 4 - 1

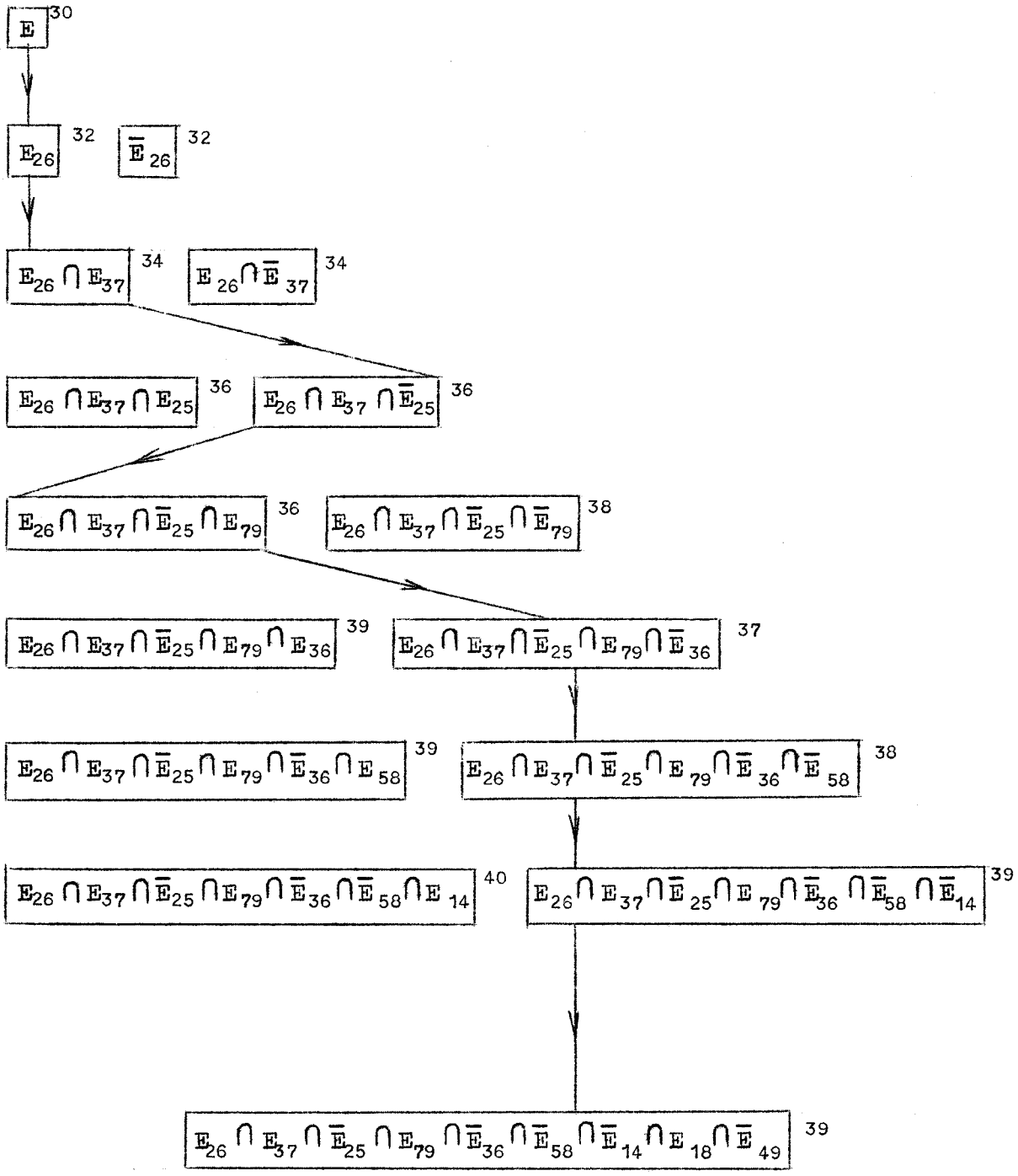
Coût = 39

Matrice des coûts

$$[\bar{C}] = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & \infty & 14 & 14 & 6 & 8 & 10 & 12 & 7 & 10 \\ 2 & 14 & \infty & 10 & 10 & 6 & 4 & 8 & 8 & 8 \\ 3 & 14 & 10 & \infty & 8 & 6 & 6 & 2 & 7 & 4 \\ 4 & 6 & 10 & 8 & \infty & 4 & 6 & 6 & 2 & 4 \\ 5 & 8 & 6 & 6 & 4 & \infty & 2 & 4 & 2 & 2 \\ 6 & 10 & 4 & 6 & 6 & 2 & \infty & 4 & 4 & 4 \\ 7 & 12 & 8 & 2 & 6 & 4 & 4 & \infty & 5 & 2 \\ 8 & 7 & 8 & 7 & 2 & 2 & 4 & 5 & \infty & 3 \\ 9 & 10 & 8 & 4 & 4 & 2 & 4 & 2 & 3 & \infty \end{array}$$

3.1.2.9.2 Application de l'algorithme de Little.

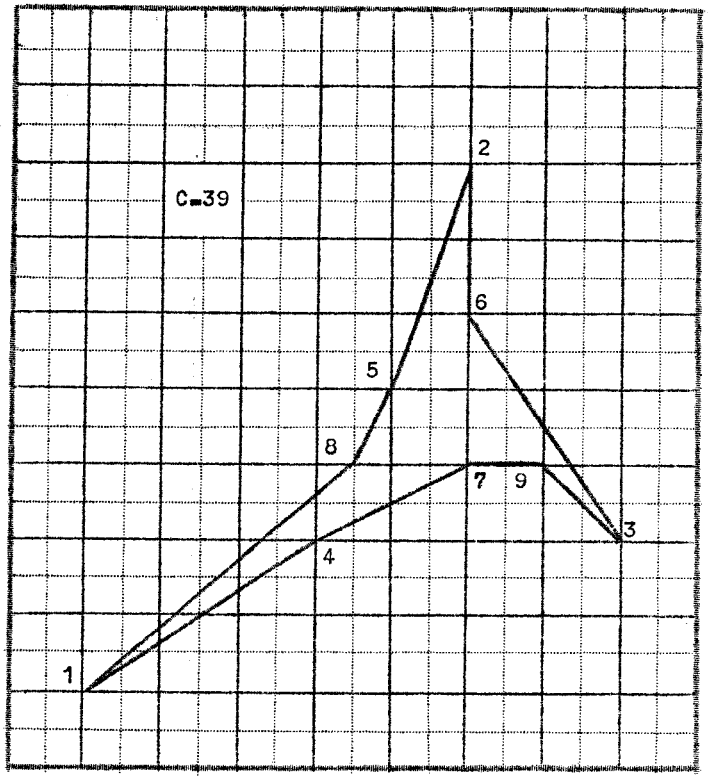
Arborescence du graphe Es. 9

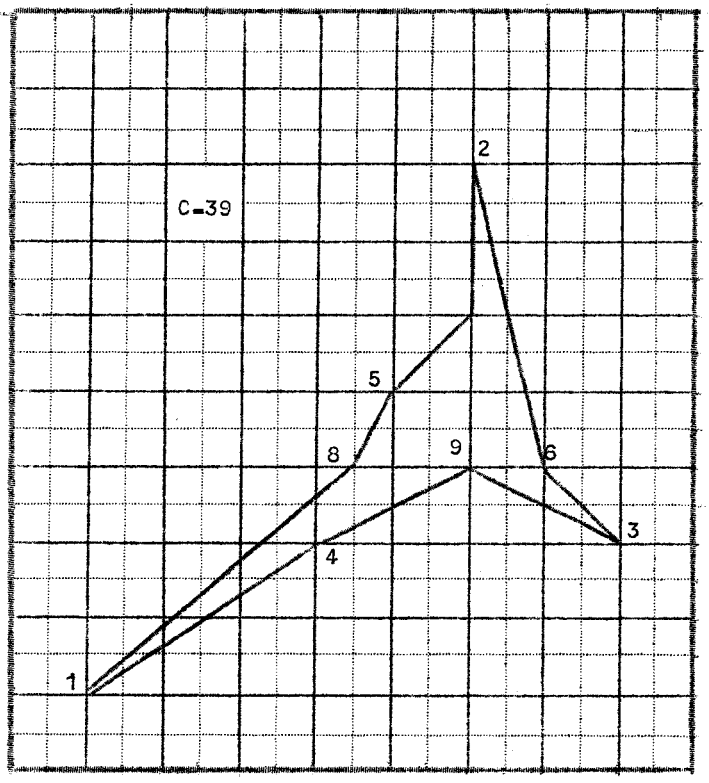


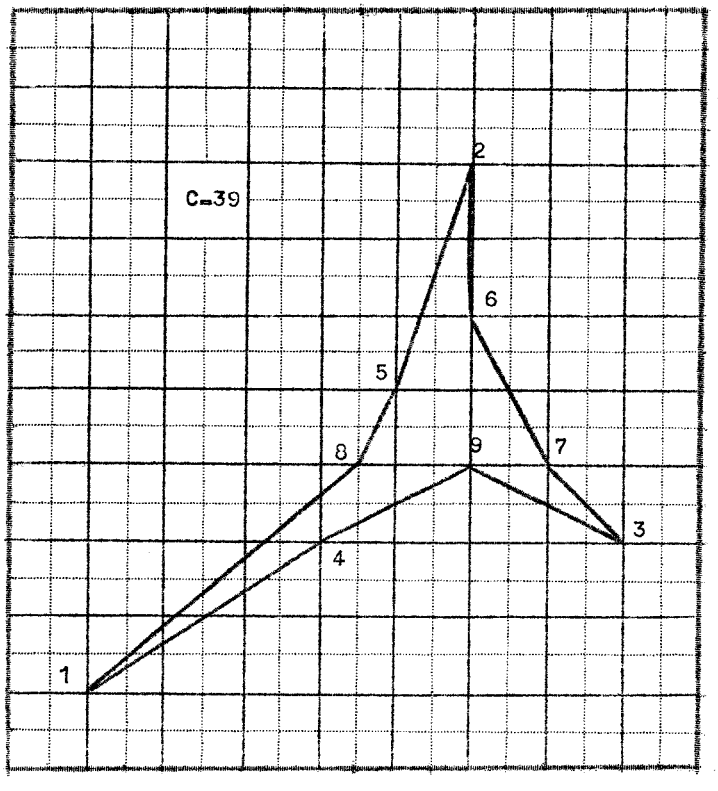
Itinéraire optimal

26 - 37 - 52 - 79 - 63 - 85 - 41 - 18 - 94

Coût = 39





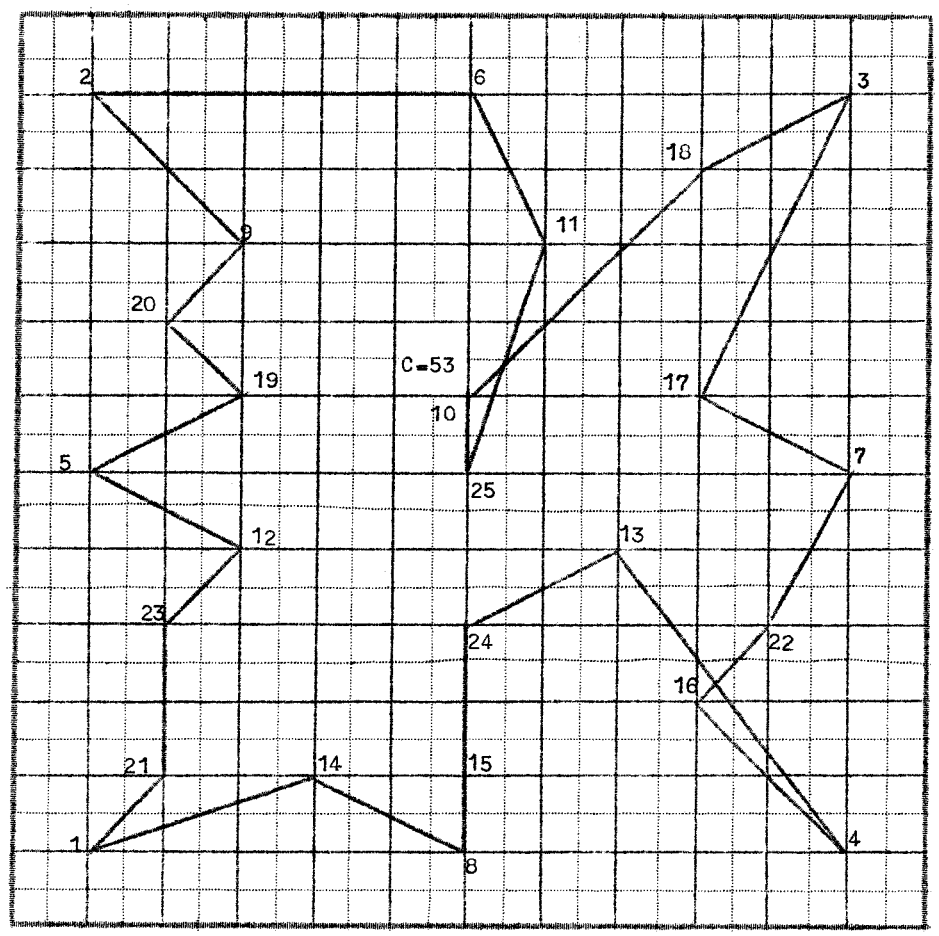


3.1.2.10 Essai 10 .

Application de l'algorithme heuristique

Coordonnées des points

1	0	0
2	0	10
3	10	10
4	10	0
5	0	5
6	5	10
7	10	5
8	5	0
9	2	8
10	5	6
11	6	8
12	2	4
13	7	4
14	3	1
15	5	1
16	8	2
17	8	6
18	8	9
19	2	6
20	1	7
21	1	1
22	9	3
23	1	3
24	5	3
25	5	5



Itinéraire obtenu

1 - 21 - 23 - 12 - 5 - 19 - 20 - 9 - 2 - 6 - 11 - 25 - 10 - 18 - 3 - 17 -
- 7 - 22 - 16 - 4 - 13 - 1

Coût = 53

3.2 ESSAIS INDUSTRIELS.

3.2.1 Commentaires.

Une multitude d'essais a été effectué sur l'ordinateur HP 21 MX en utilisant le programme complet. Nous donnons ci-après un aperçu de certain de ces essais en donnant quelques résultats.

N° de la plaquette	Nombre de points	Ancien coût	Nouveau coût	Gain réalisé %	T_c
Es.1 610505	163	684	549	19,7	29,65
Es.2 671397 (a)	487	1202	904	24,8	246,18
Es.3 671397 (b)	373	1404	945	32,7	146
Es.4 671397 (c)	97	719	380	47,2	11,13
Es.5 2413005 (a)	133	650	237	63,5	19,16
Es.6 2413005 (b)	266	2152	986	54,2	73,20
Es.7 2413005 (c)	890	2052	1818	11,4	827,59
Es.8 24151301 (a)	265	3363	1076	68	72,46
Es.9 24151301 (b)	224	2051	902	56	53,11
Es.10 709512	1041	2107	1772	15,9	1112,34
Es.11 709635	376	2156	1544	53,7	146,32
Es.12 709502	508	2346	1544	34Ω2	274,5
Es.13 2413004	1037	2489	2160	13,2	1114,2
Es.14 671396	530	1625	1312	19,24	234,25

Dans les paragraphe 3.2.2.1 , 3.2.2.2 , 3.2.2.3 et 3.2.2.4 sont figurés respectivement les résultats des plaquettes Es.1 , Es.5 , Es.8 et Es.14 .

La figure du paragraphe 3.2.2.5 représente le temps de calcul T en fonction du nombre de points à releve. Nous constatons que T (temps réel) suit pratiquement la formule calculée :

$$T_c = KN^2 + K_1 N$$

$$T_c \approx 0{,}00105\ N^2$$

Remarquons que nous aurions pu compliquer l'algorithme avec certaines règles, données par BARACHET (Réf. 24) et ou par RAYMOND (Réf. 36) pour améliorer le degré d'optimalité de l'itinéraire mais cette opération entraînerait un temps de calcul plus long. Le nouveau bénéfice réalisé est faible et dans beaucoup de cas nul.

En utilisant le critère des moindres carrés l'expression du temps de calcul :

$$T_c = KN^2 + K_1 N$$

peut se mettre sous la forme traditionnelle :

$$T_c = a\,N^b$$

avec :

$$a = e^c$$

$$c = \frac{1}{\Delta}\begin{vmatrix} \Sigma Y_i & \Sigma X_i \\ \Sigma X_i Y_i & \Sigma X_i^2 \end{vmatrix}$$

$$b = \frac{1}{\Delta}\begin{vmatrix} N' & \Sigma Y_i \\ \Sigma X_i & \Sigma X_i Y_i \end{vmatrix}$$

$$\Delta = \begin{vmatrix} N' & \Sigma X_i \\ \Sigma X_i & \Sigma X_i^2 \end{vmatrix}$$

$$X_i = \ln x_i \quad \text{et} \quad Y_i = \ln y_i$$

N' = nombre de points expérimentaux considéré :

$$(x_i, y_i) \quad \forall i = 1, 2, \ldots, N'$$

La variance sur les coefficients a, b et c est donnée par :

$$\sigma_a^2 = a^2 \sigma_c^2 \qquad \sigma_b^2 = N' \frac{\sigma^2}{\Delta} \qquad \sigma_c^2 = \frac{\sigma^2}{\Delta} \Sigma x_i^2$$

$$\sigma^2 = \frac{1}{N' - 2} (Y_i - c - bX_i)^2$$

En partant de cette technique l'expression du temps de calcul de l'algorithme proposé devient :

$$\boxed{T_c = 0{,}00138\ N^{1{,}955}}$$

L'algorithme heuristique connu comme étant le meilleur à l'heure actuelle est celui de WEBB (Thèse de VAN DER CRUYSSE Université d'Anvers - 1976) avec un temps de calcul $T_c = 0{,}00625\ N^{1{,}8}$

Pour un problème à 10 noeuds, l'algorithme de WEBB donne une erreur de 4,36% sur l'optimalité et un T_c de 0,39 sec. L'algorithme heuristique que nous proposons donne pour un problème identique respectivement 0 % d'erreur et $T_c = 0{,}125$ sec.

(mais $\hat{C}_y$ = durée)

Note :

$$T_c(\text{S.E.P}) = 0{,}00339\ N^{3{,}36}$$

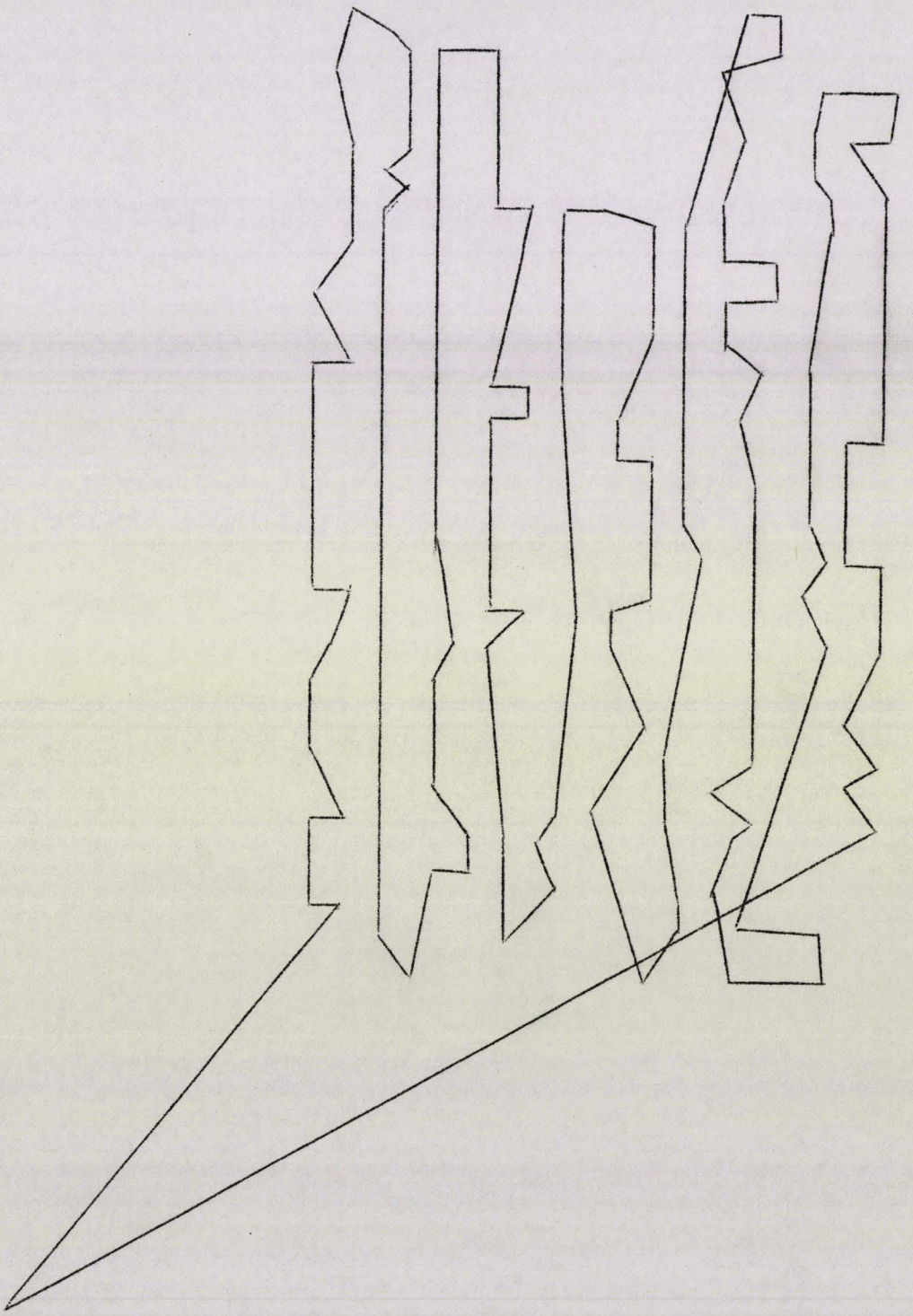
erreur = 0 %

3.2.2 Résultats.

3.2.2.1 Essai 1 .

3.2.2.1.1 Application de l'heuristique.

3.2.2.1.1.1 Ancien itinéraire (N = 163) .

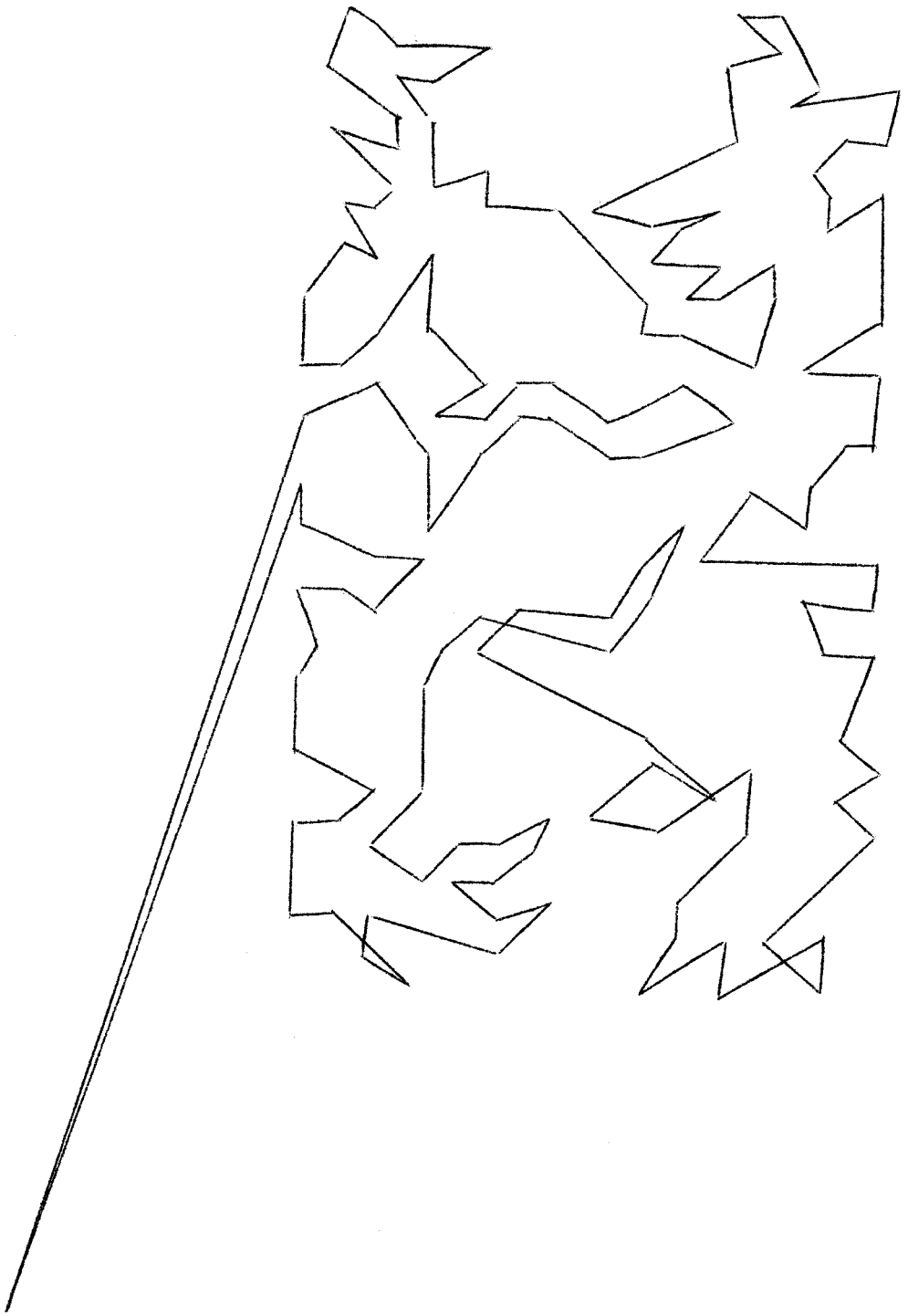


3.2.2.1.1.2 Nouvel itinéraire.

Gain 19,75 %

T = 29,65 sec

N = 163



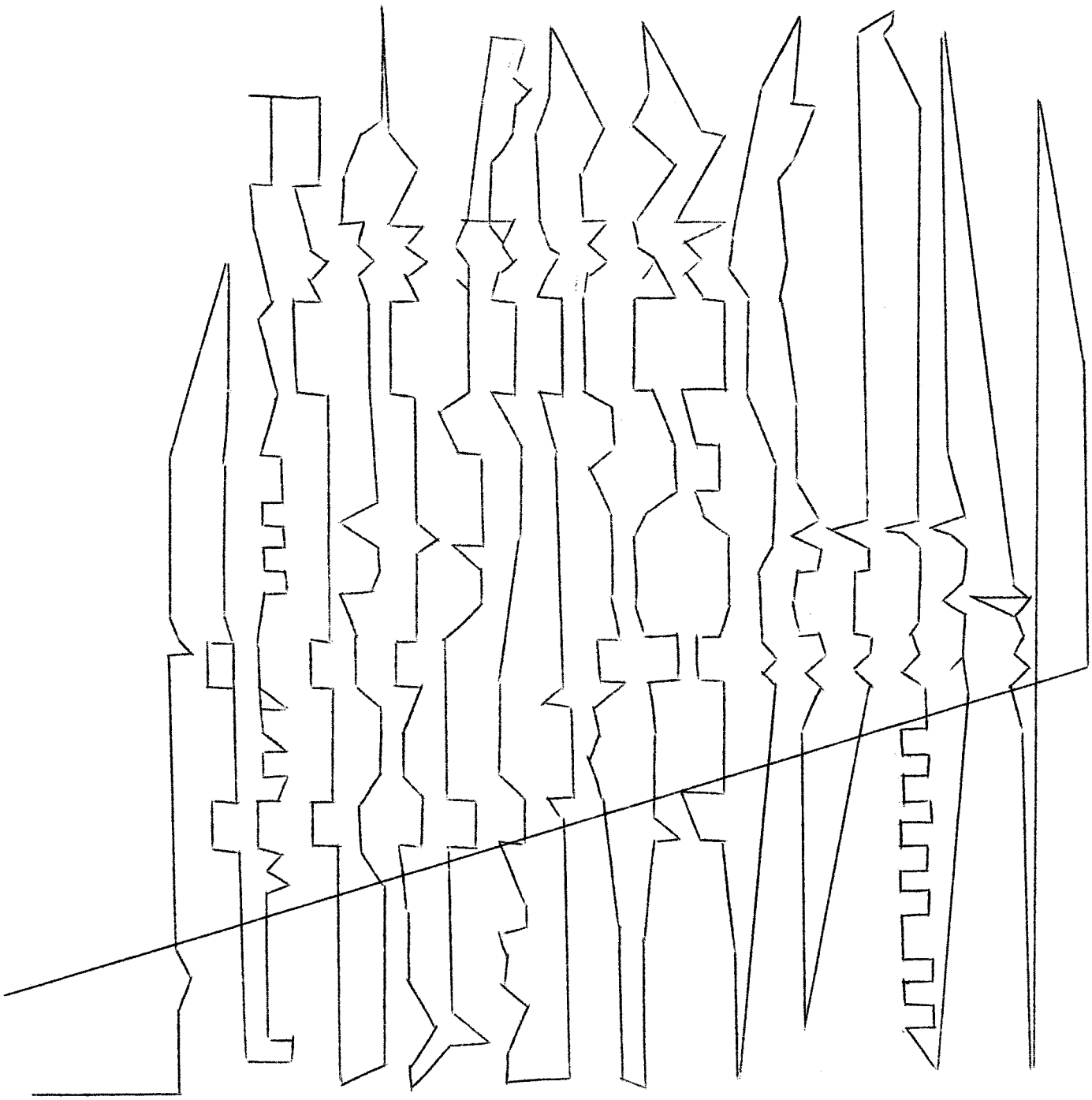
3.2.2.1.1.3 Extrait de la feuille de calcul.

```
 ER ZIJN   163 PUNTEN IN BESTAND AP2
ER ZIJN   163 PUNTEN IN BESTAND AP2
 OORSPRONKELIJKE WEGLENGTE =    684.975
OORSPRONKELIJKE WEGLENGTE =    684.975
  1Ø H  43 M  54 S  46.
  1Ø H  43 M  54 S  46.
LOOP<  2 =   23
LOOP<  3 =  113


LOOP<  4> =  146
LOOP<  5> =  147
LOOP<  6> =  15Ø
LOOP<  7> =  153
LOOP<  8> =  151
LOOP<  9> =  163
LOOP< 1Ø> =  132
LOOP< 11> =    1
 VERBETERINGSFACTOR :      1. ØØ : NA   1 LOOPS ZIJN ER
 VERBETERINGSFACTOR :      1. ØØ : NA   1 LOOPS ZIJN ER
   1Ø H  44 M  23 S  52.
  1Ø H  44 M  23 S  52.
OUDE WEG =     684.97
NIEUWE WEG =     549.64
WINST =    19.76 %

    1  15  34  53  52  51  71  69  83  92
  93 124 1Ø7  91  84  68  7Ø  54  67  55
  56  57  33  17  16  18  19  32  2Ø  31
  3Ø  29  21  28  27  22  23  24  25  62
  63  61  26  5Ø  59  58  64  65  66  85
  89  9Ø 1Ø8 122 123 12Ø 119 121 1Ø9 118
 11Ø  98 111 117  87  86 116 115 114 112
 113 145 144 146 147 148 143 142 141 149
 15Ø 151 14Ø 152 153 154 155 139 138 125
 1Ø5 137 156 157 158 136 135 159 16Ø 161
 162 163 134 132 133 131 13Ø 1ØØ 1Ø1 1Ø2
 129 128 126 1Ø3  99 1Ø4 127  98  74  73
  82  96  94 1Ø6  95  97  72  49  48  47
  46  45  38  42  44  75  81  8Ø  76  43
  77  79  78  39  4Ø  41   2   3   4   5
   6  37   7   8   9  1Ø  12  11  36  5Ø
  35  13  14   1
```

3.2.2.2 Essais 14 .

3.2.2.2.1 Application de l'heuristique.

3.2.2.2.1.1 Ancien itinéraire (N = 530) .

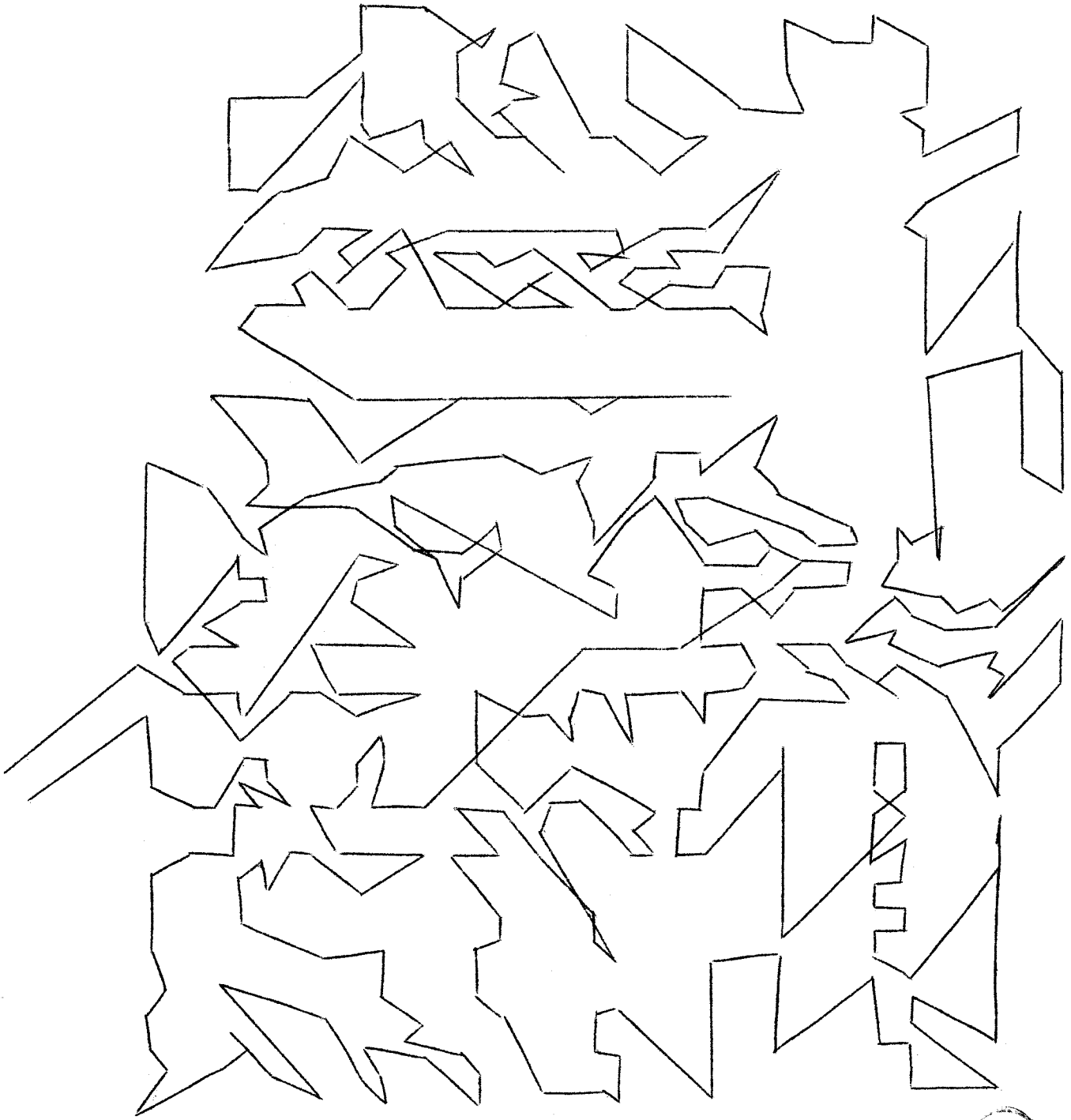


3.2.2.2.1.2 Nouvel itinéraire.

Gain 19,24 %

T = 234,25 sec

N = 530



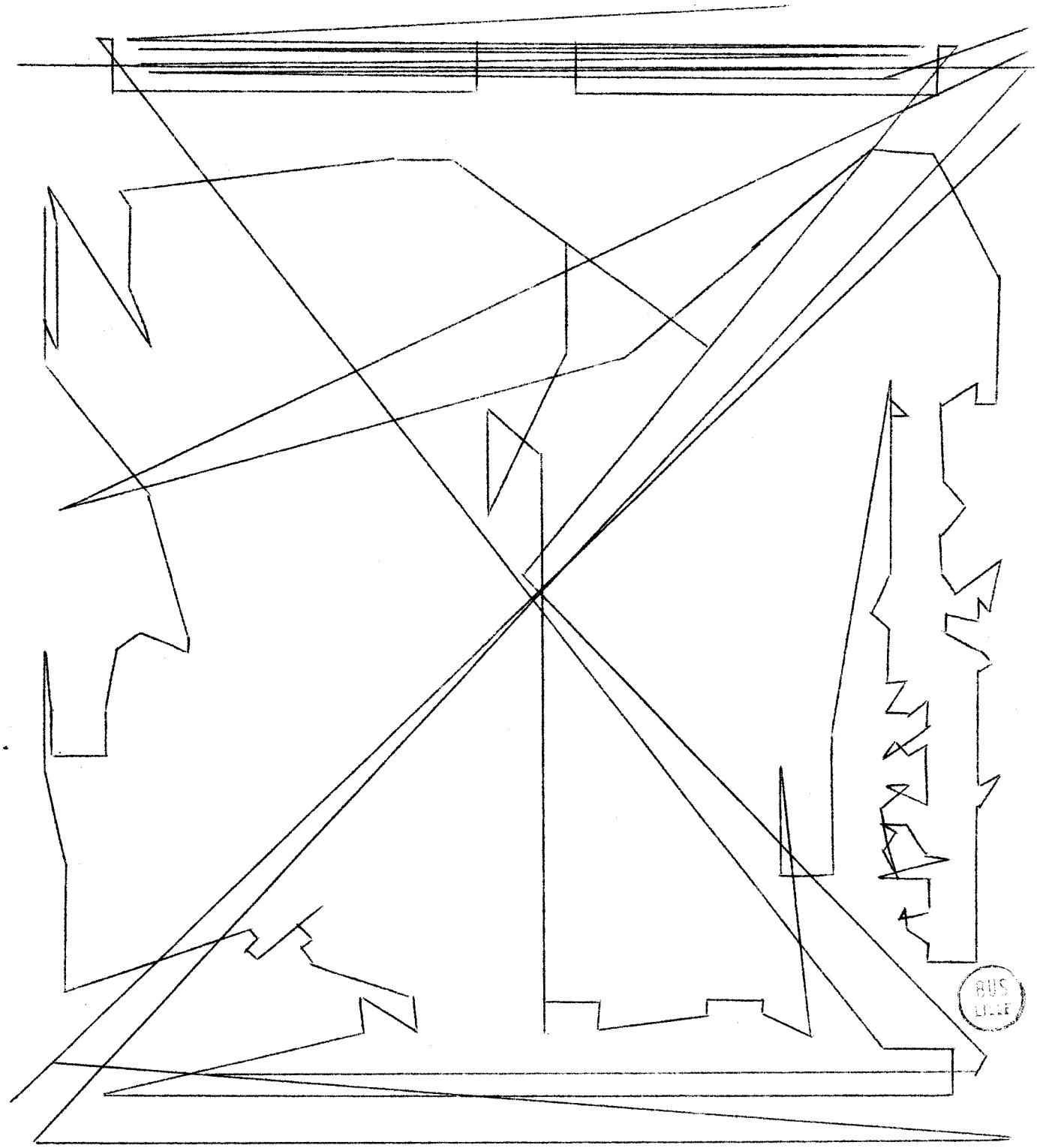
3.2.2.2.1.3 Extrait de la feuille de calcul.

```
 ER ZIJN   53Ø PUNTEN IN BESTAND AP19
ER ZIJN   53Ø PUNTEN IN BESTAND AP19
 OORSPRONKELIJKE WEGLENGTE =   1626.129
OORSPRONKELIJKE WEGLENGTE =   1626.129
  12 H  55 M  42 S  51.
  12 H  55 M  42 S  51.
 LOOP<  2> =   79
LOOP<  3> =  1.35
LOOP<  4> =  1.99
LOOP<  5> =  4.18

LOOP<  7> =  523


LOOP<  8> = 424
LOOP<  9> = 525
LOOP< 1Ø> = 5Ø8
LOOP< 11> =   1
 VERBETERINGSFACTOR :          1. ØØ : NA   1 LOOPS ZIJN ER
 VERBETERINGSFACTOR :          1. ØØ : NA   1 LOOPS ZIJN ER
   13 H   Ø M  33 S  76.
  13 H   Ø M  33 S  76.
OUDE WEG =   1.626.13
NIEUWE WEG =    1.313.33
WINST =    19.24 %

   1  14  31  32  57  55  93 113 116 152
 115 114 177 153 154 112  92 111 155 156
 176 11Ø  95  94  56  54  15  3Ø  29  58
 4Ø2 352 353 354 368 263 281 282 285 284
 283 235 232 231 23Ø 229 228 227 173 225
 225 174 223 224 235 279 28Ø 287 237 239
 278 288 289 29Ø 291 292 277 238 222 221
 22Ø 241 24Ø 276 275 273 274 293 294 295
 344 346 345 357 358 359 342 341 34Ø 36Ø
 287 385 412 413 414 384 386 343 296 297
 271 272 242 175 160 159 158 157 189 188
 1Ø7  96 1Ø6 162 161 172 1Ø5  99  98  44
  46  45  43  42 1ØØ 163 164 165 171 186
 17Ø 233 234 169 1Ø4 167 168 1Ø3 1Ø1  37
  4Ø 1Ø2  39  41  38   2 53Ø   3   4   5
   6   7   8   9  35  36  47  48  49  51
  97  5Ø  53  52  33  34  1Ø  11  12  13
```

3.2.2.3 Essais 5 .

3.2.2.3.1 Application de l'heuristique.

3.2.2.3.1.1 Ancien itinéraire (N = 133) .

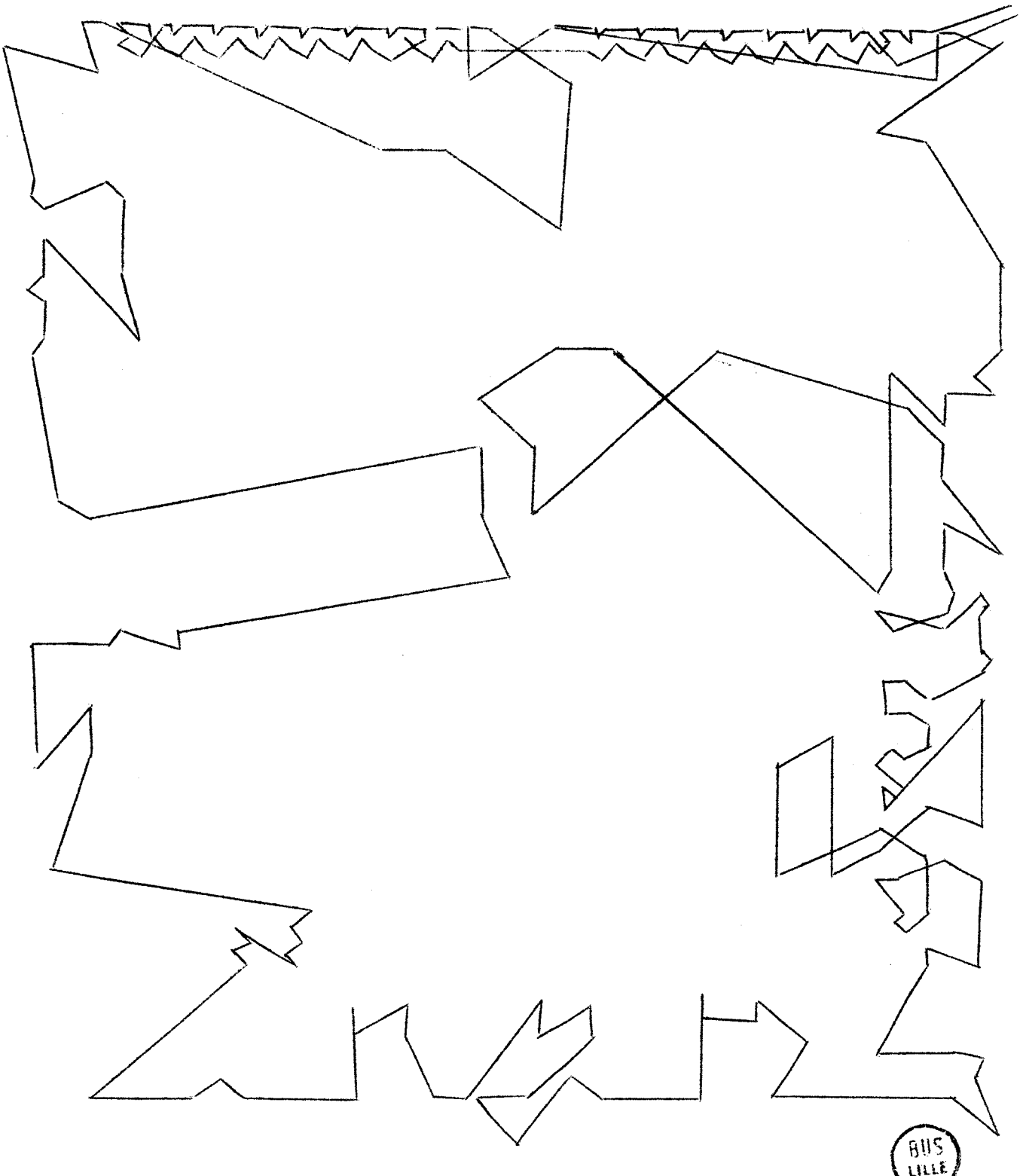


3.2.2.3.1.2 Nouvel itinéraire.

Gain 63,50 %

T = 19,160 sec

N = 133



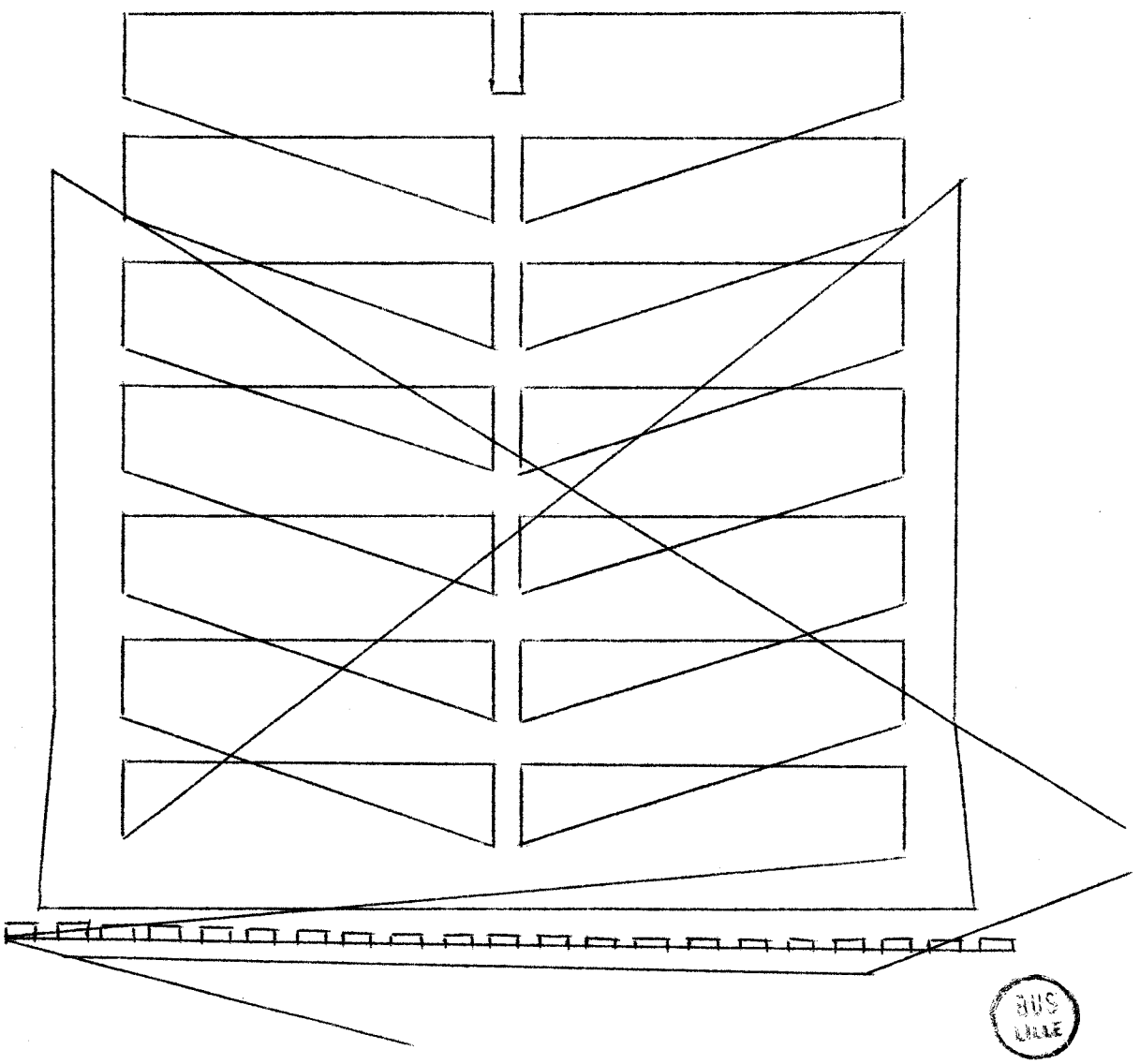
3.2.2.3.1.3 Extrait de la feuille de calcul.

```
 ER ZIJN   133 PUNTEN IN BESTAND AP8
ER ZIJN   133 PUNTEN IN BESTAND AP8
 OORSPRONKELIJKE WEGLENGTE =   650.000
OORSPRONKELIJKE WEGLENGTE =   650.000
  11 H   2 M  57 S  43.
  11 H   2 M  57 S  43.
LOOP<  2> = 133
LOOP<  3> = 118
LOOP<  4> = 100
LOOP<  5> =   3
LOOP<  6> =   1
 VERBETERINGSFACTOR :      1. 00 : NA   1 LOOPS ZIJN ER
 VERBETERINGSFACTIR :      1. 00 : NA   1 LOOPS ZIJN ER
  11 H   3 M  16 S  59.
  11 H   3 M  16 S  59.
OUDE WEG =     650.00
NIEUWE WEG =     237.00
WINST =    63.54 %

   1 117 133  62  61  95  60  59  58  57
  94  56  55  54  53  92  52  51  50  49
  90  48  47  46  45  88  44  43  42  41
  86  40  39  38  37  84  36  35  34  33
  32  82  31  30  29  28  80  27  26  25
  24  78  23  22  21  20  76  19  18  17
  16  74  15  14  13  12  72  11  10   9
   8  70   7   6 118 100  68   2   3   4
   5  69 101 119  71 102 120  73 103 121
  75 104 122  77 105 123  79 106 124  81
 107 125  83 108 109 126  85 110 127  87
 111 128  99 112 129  91 113 130  93 114
 131  95 115 132  97 115  98  63  64  65
  99  66  67   1
```

3.2.2.4 Essais 8 .

3.2.2.4.1 Application de l'heuristique.

3.2.2.4.1.1 Ancien itinéraire (N = 265) .

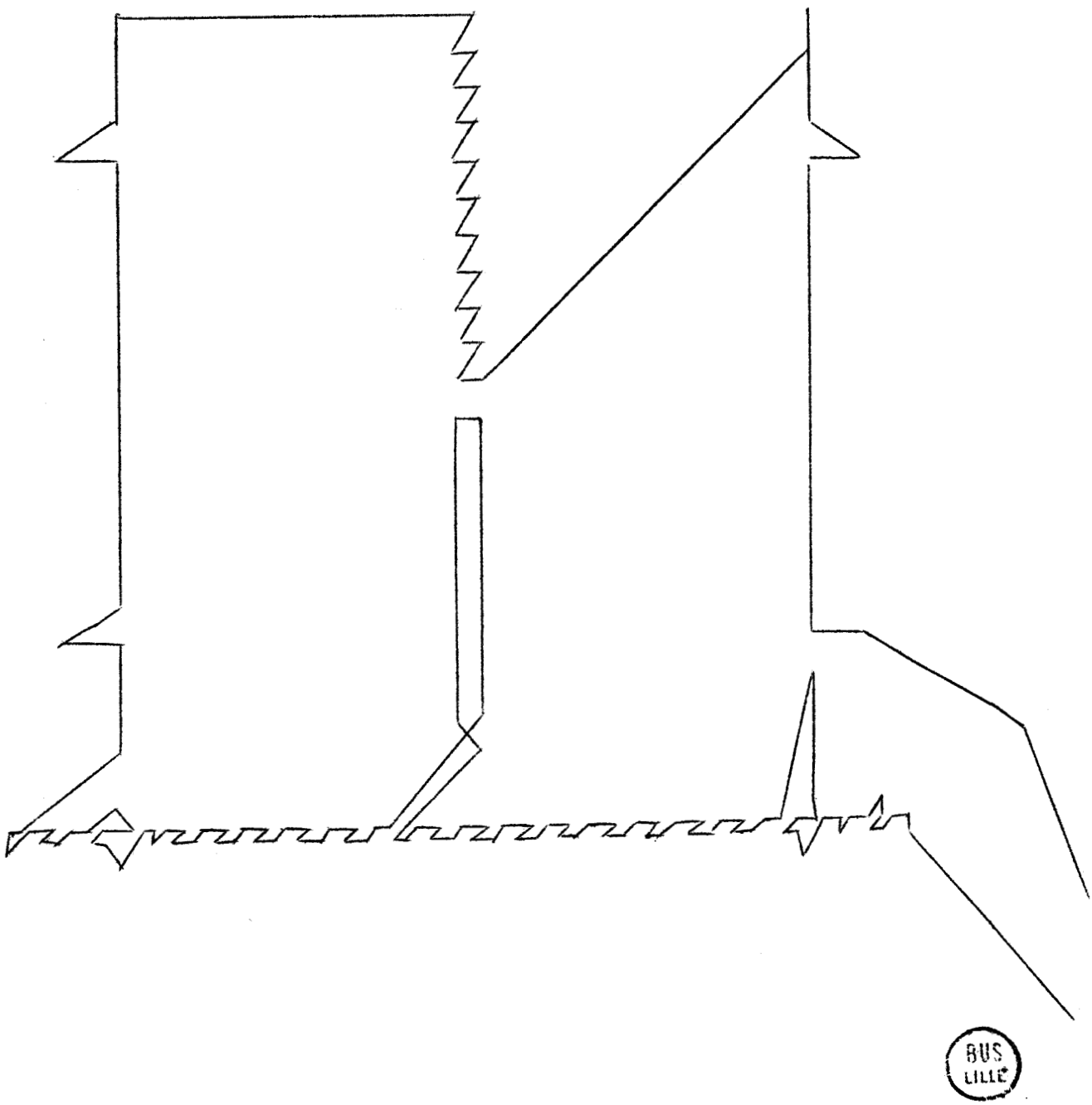


3.2.2.4.1.2 Nouvel itinéraire.

Gain 68 %

T = 72,460 sec

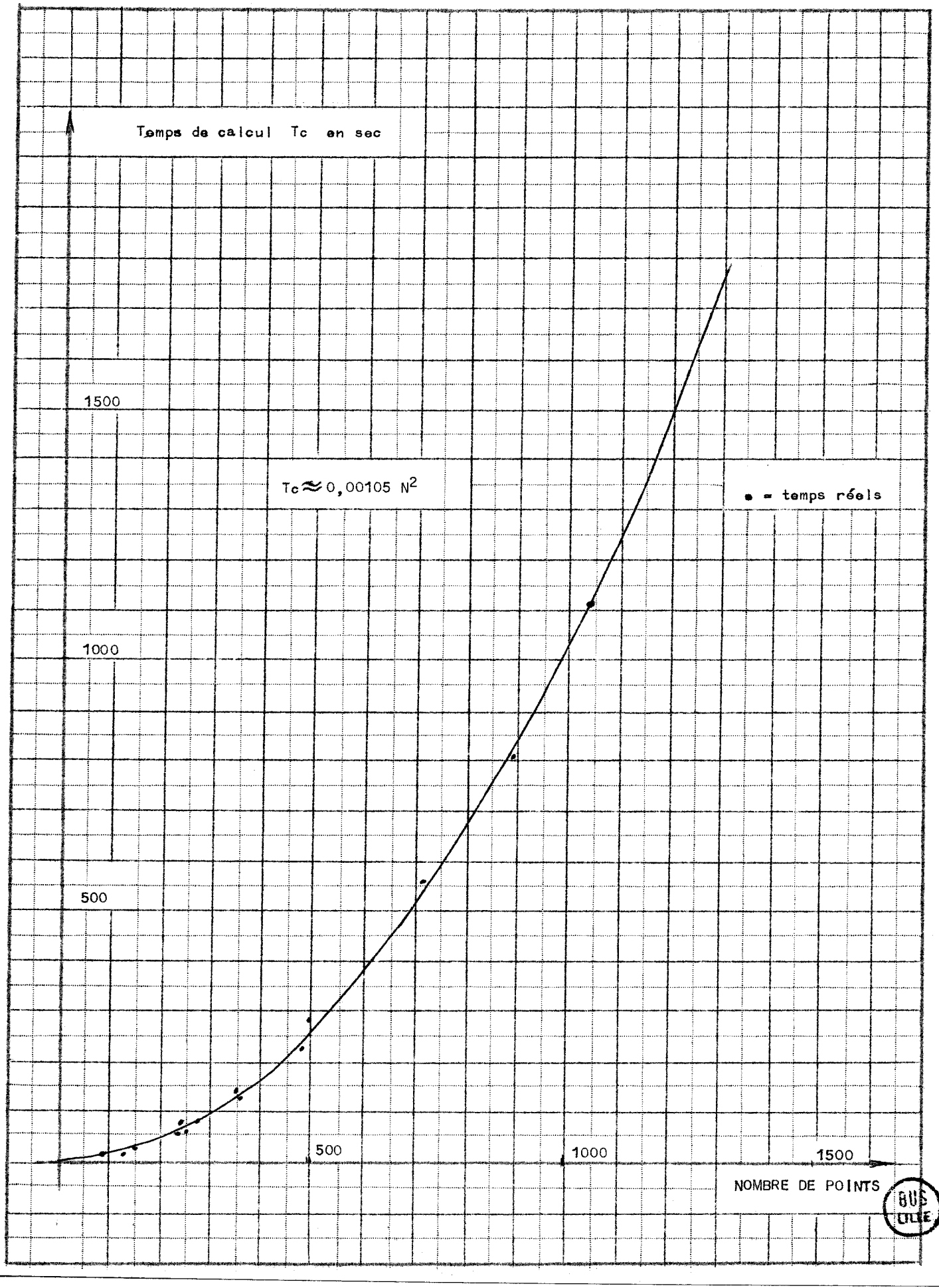
N = 265



3.2.2.4.1.3 Extrait de la feuille de calcul.

```
 ER ZIJN   265 PUNTEN IN BESTAND AP11
ER ZIJN   265 PUNTEN IN BESTAND AP11
 OORSPRONKELIJKE WEGLENGTE =    3363.5ØØ
OORSPRONKELIJKE WEGLENGTE =    3363.5ØØ
  11 H  21 M  55 S  83.
  11 H  21 M  55 S  83.
LOOP<  2> =   2
LOOP<  3> = 169
LOOP<  4> = 252
LOOP<  5> = 216
LOOP<  6> = 2Ø8
LOOP<  7> = 255
LOOP<  9> = 263

 VERBETERINGSFACTOR :       1.ØØ : NA   1 LOOPS ZIJN ER
 VERBETERINGSFACTOR :       1.ØØ : NA   1 LOOPS ZIJN ER
  11 H  23 M   8 S  29.
  11 H  23 M   8 S  29.
OUDE WEG =     3363.5Ø
NIEUWE WEG =     1Ø76.75
WINST =     67.99 %

   1   3   4   5   6   7   8  1Ø  11   9
257  12  13  14  15  18  16  17  19 264
 26  27  25  24  23  22  21  2Ø 258 171
172 173  28 229  3Ø  31  32  34  35  33
 36  37  38  39  4Ø  42  43  41  44  45
 46  47  48  5Ø  51  49  52  53  54  55
 56  58  59  57  6Ø  61  62  63  64  66
 67  65  68  69  7Ø  71  72  74  75  73
```

3.2.2.5 Temps de calcul (T_c) .



CONCLUSION GENERALE

L'algorithme heuristique que nous avons présenté, ne garantit pas la solution optimale (ce serait trop beau) mais il donne une très "bonne" solution pratique en un temps de calcul très court.

Compte tenu de ce temps de calcul et du degré d'optimalité obtenu, il fournit à notre humble avis, des résultats meilleurs que ceux obtenus avec les algorithmes heuristiques de KARG-THOMPSON (Réf. 31) et de WEBB.

Nous pensons que l'originalité de la méthode se trouve dans l'introduction et la formation initiale des graphes traversant l'ensemble des points. Ces graphes permettent une très bonne sélection des points en vue de l'évaluation de l'itinéraire. En plus, en considérant comme point de départ de la procédure les éléments a situés sur le graphe G1 on divise le nombre de possibilités par un facteur :

$$\frac{a-1}{2}\ !$$

L'algorithme heuristique décrit dans ce travail ne prétend pas avoir résolu le problème difficile posé, mais constitue un outil assez puissant pour obtenir une bonne solution pratique - et pratiquement il vaut mieux de disposer d'un petit outil de travail que de devoir trop longtemps attendre le "bulldozer scientifique analytique".

L'adoption d'une méthode heuristique est uniquement basée sur les résultats qu'elle procure; il est d'ailleurs, à l'heure actuelle, assez difficile de tester analytiquement la solution obtenue pour un ensemble élevé de points

" Such a judgment is, of course, bound to be condemned as subjective and unscientific however, we simply cannot wait until the theorists have developed a truly exact method and, as engineers, must proceed with the job of solving the problem with the ressources at hand"

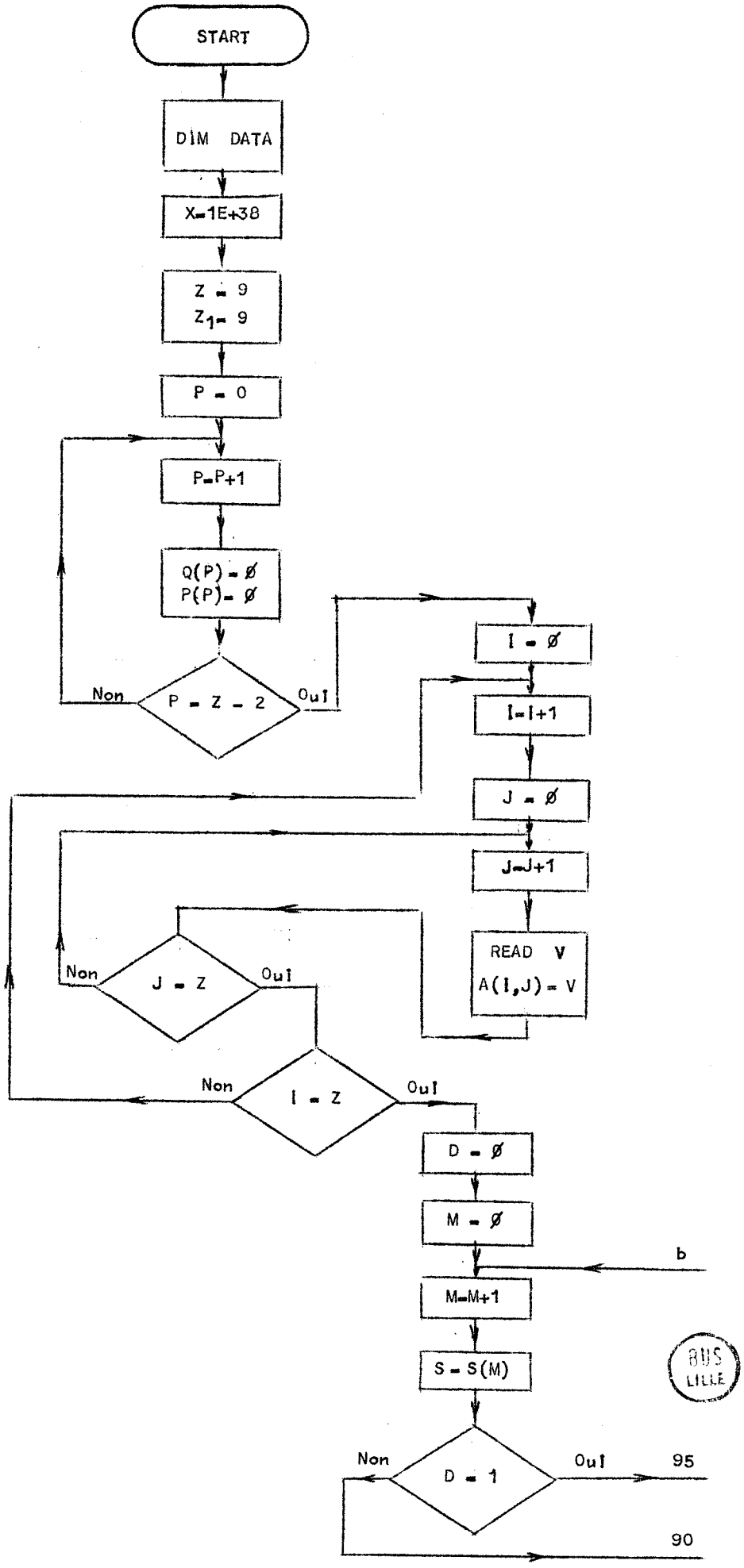
IGNIZIO , WYSKIDA , WILHELM (Réf. 40)

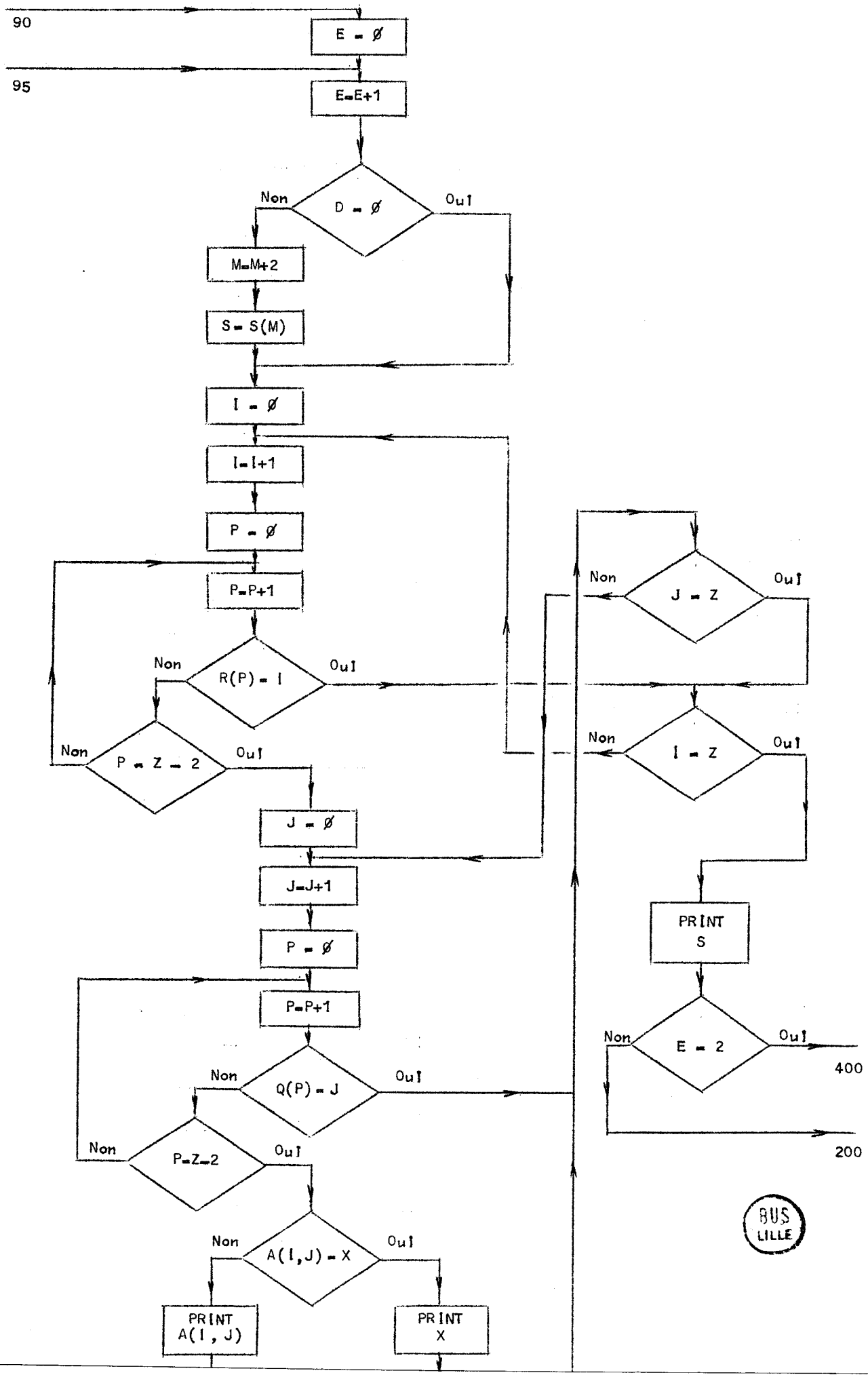
****
*****

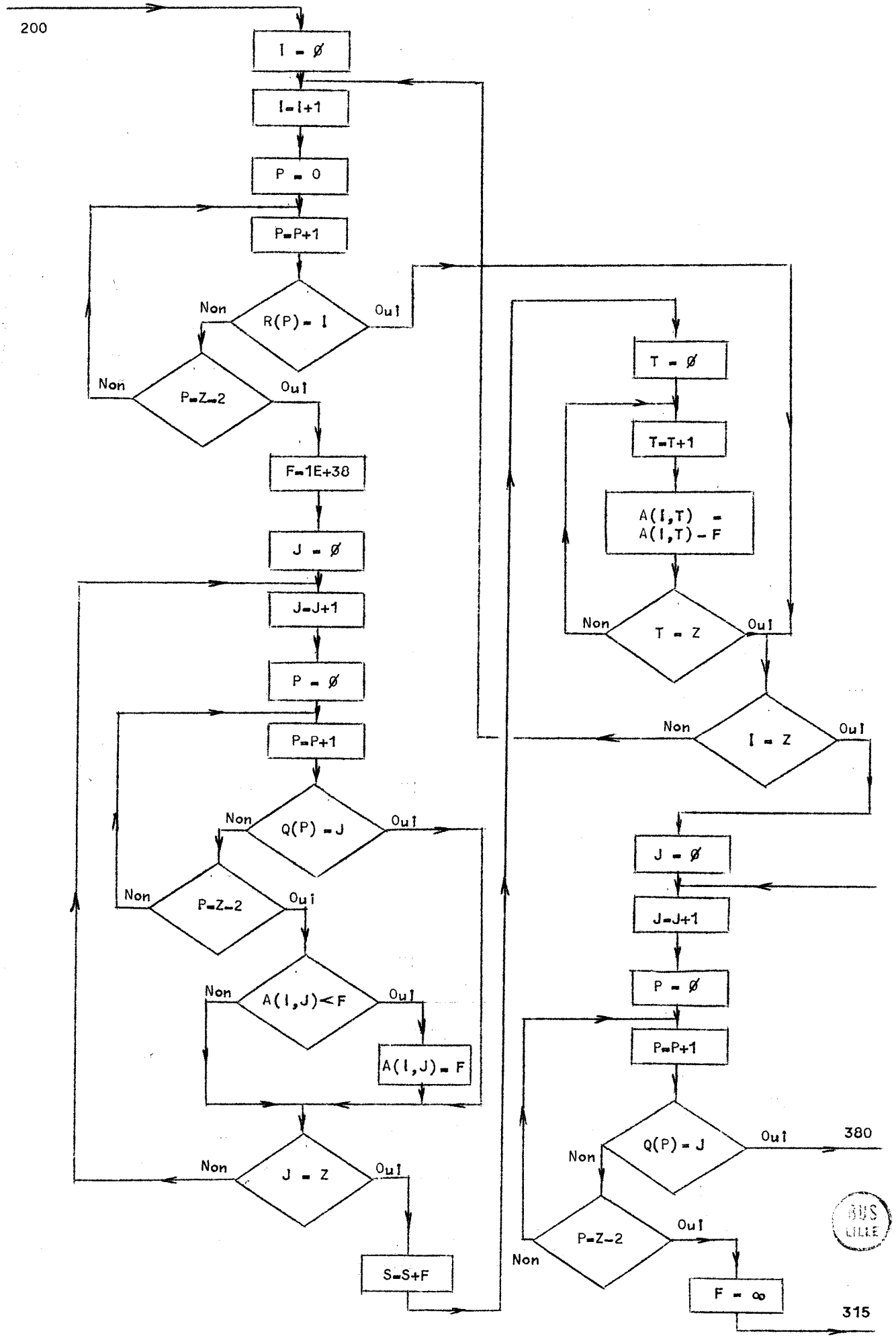
A N N E X E S

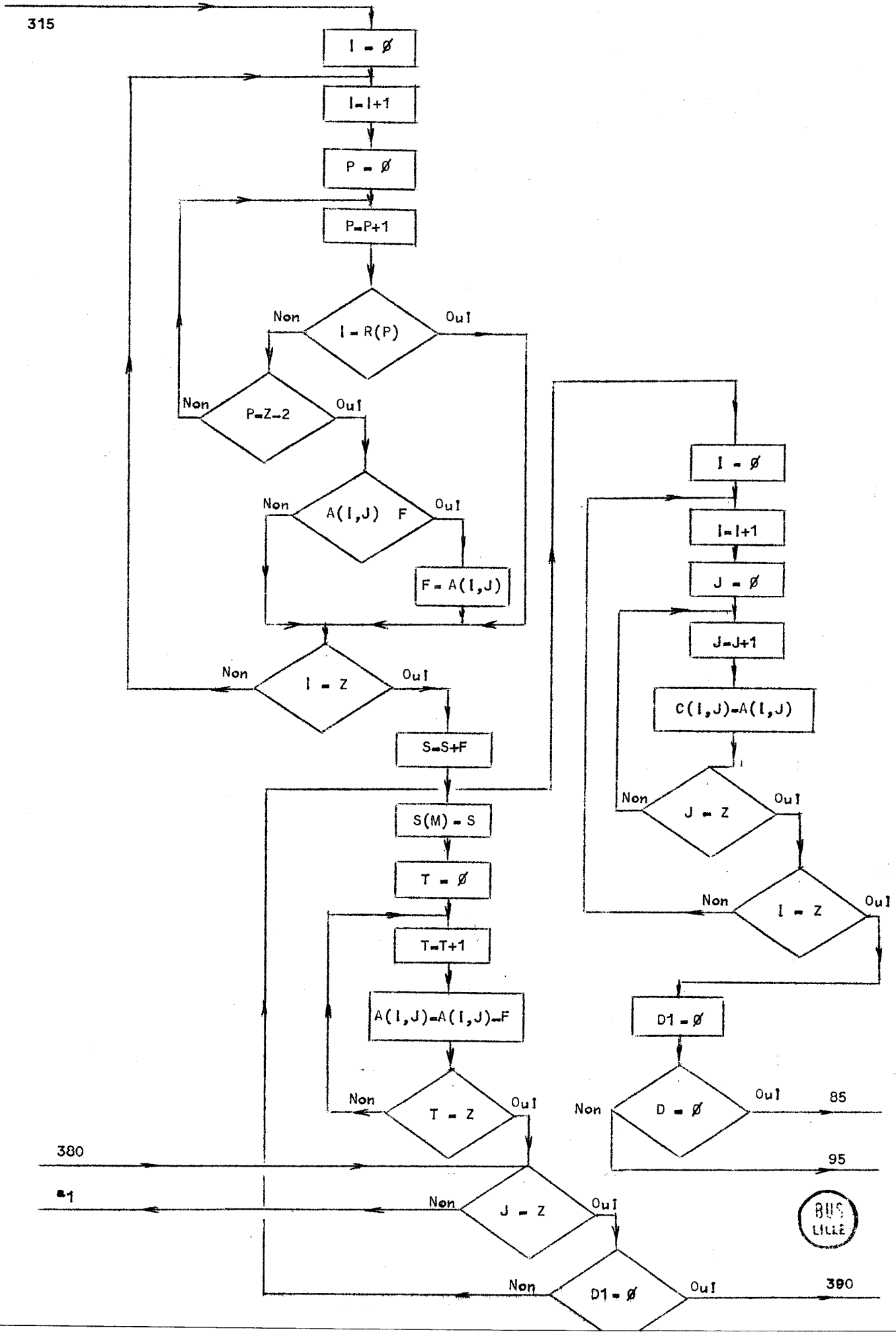
****

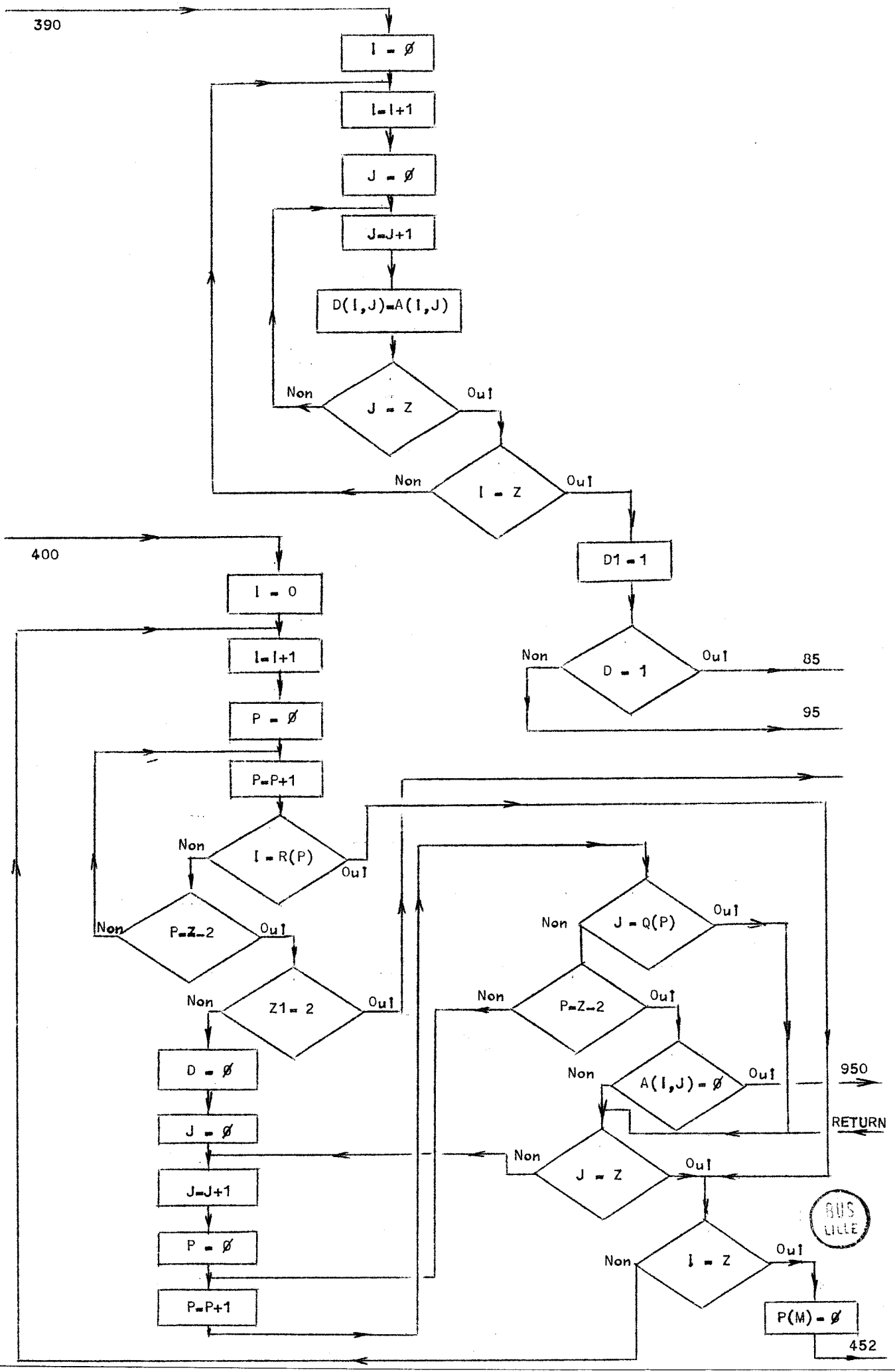
ANNEXE A1

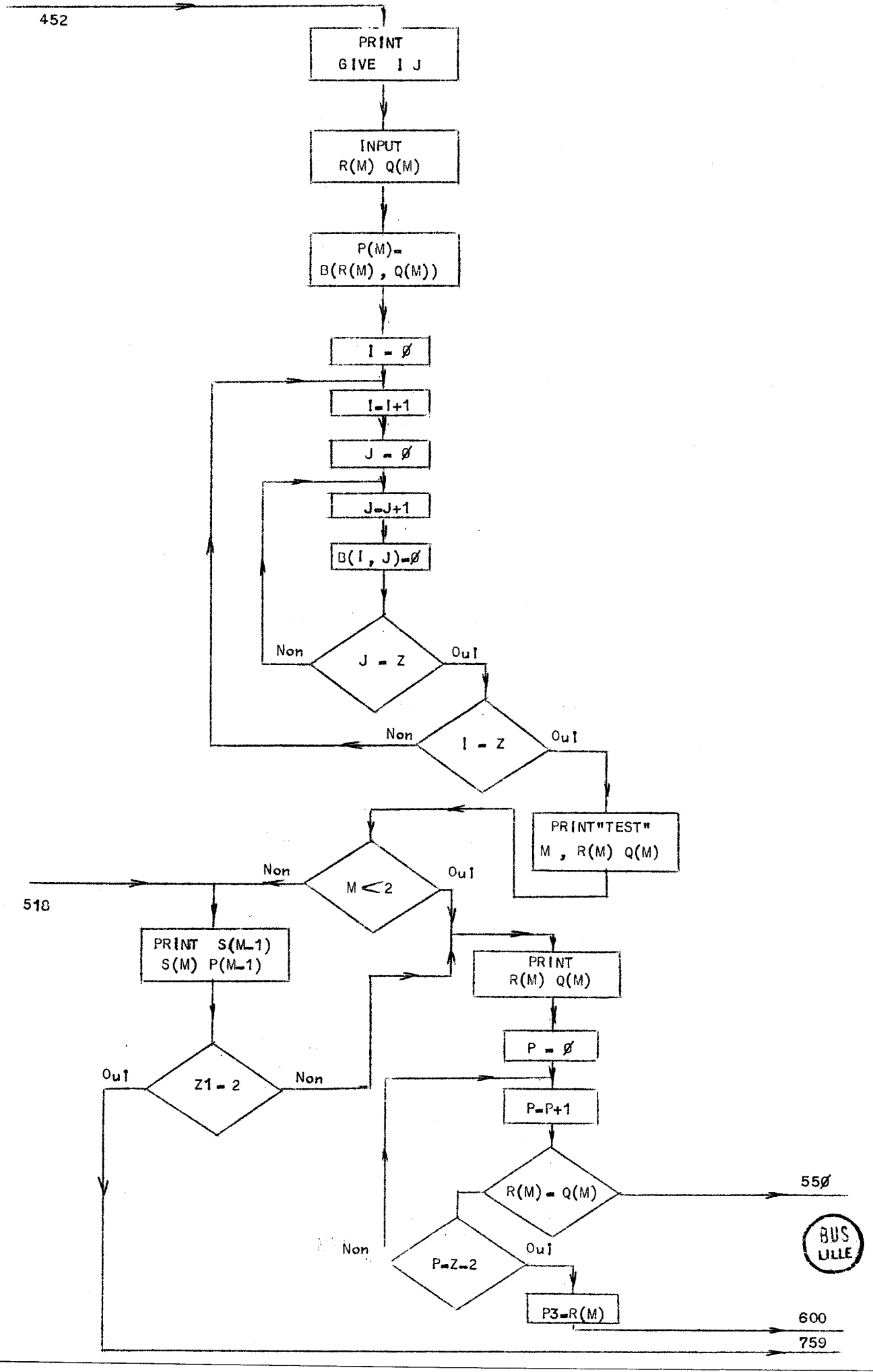

START
DIM DATA
X=1E+38
Z = 9
Z1= 9
P = 0
P=P+1
Q(P) = Ø
P(P)= Ø
Non
P = Z - 2
Oui
I = Ø
I=I+1
J = Ø
J=J+1
READ V
A(I,J) = V
Non
J = Z
Oui
Non
I = Z
Oui
D = Ø
M = Ø
b
M=M+1
S = S(M)
Non
D = 1
Oui
95
90

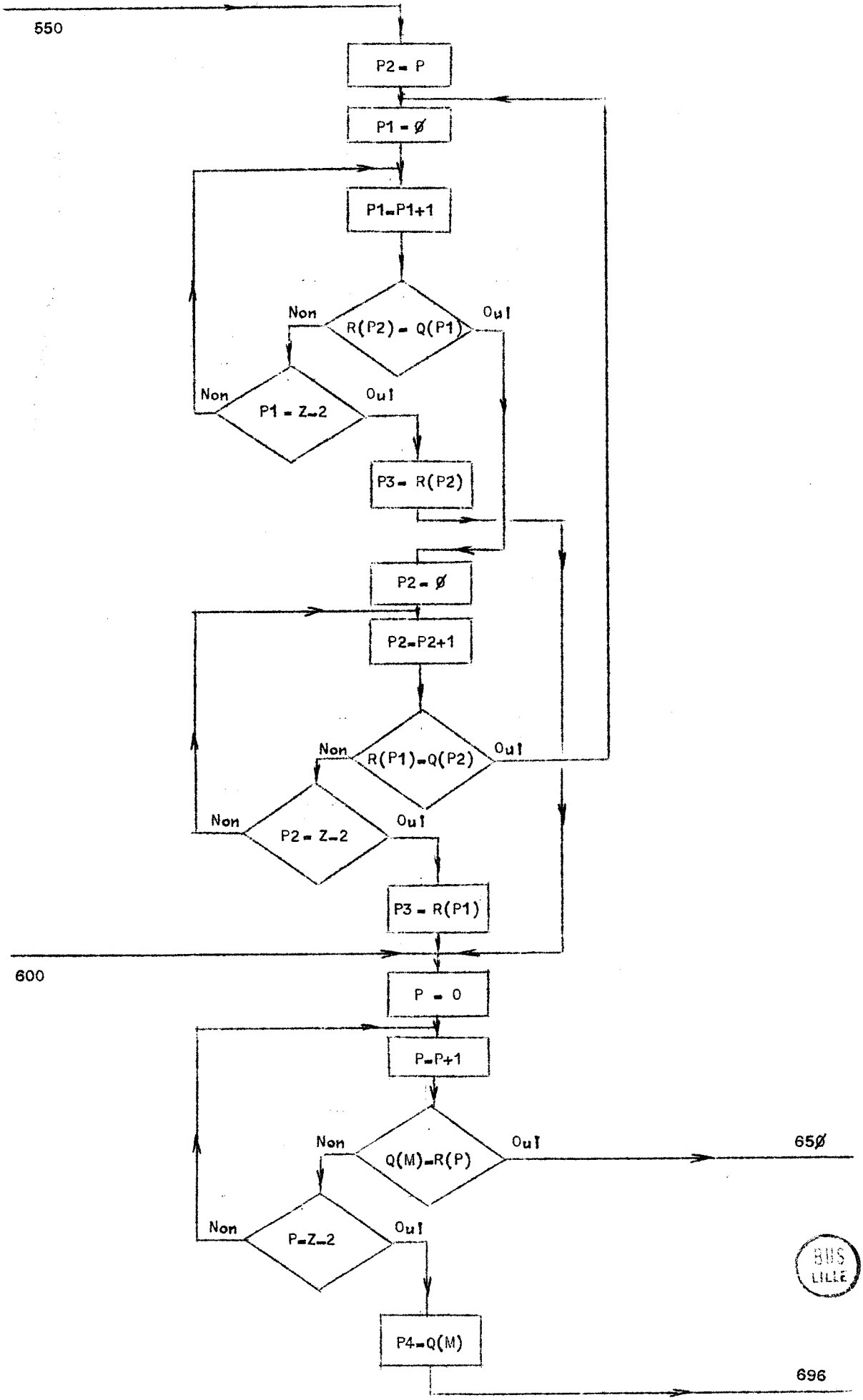

90
E = Ø
95
E=E+1
D = Ø
Non
Oui
M=M+2
S = S(M)
I = Ø
I=I+1
P = Ø
P=P+1
R(P) = I
Non
Oui
P = Z - 2
Non
Oui
J = Ø
J=J+1
P = Ø
P=P+1
Q(P) = J
Non
Oui
P=Z-2
Non
Oui
A(I,J) = X
Non
Oui
PRINT A(I , J)
PRINT X
J = Z
Non
Oui
I = Z
Non
Oui
PRINT S
E = 2
Non
Oui
400
200
BUS LILLE

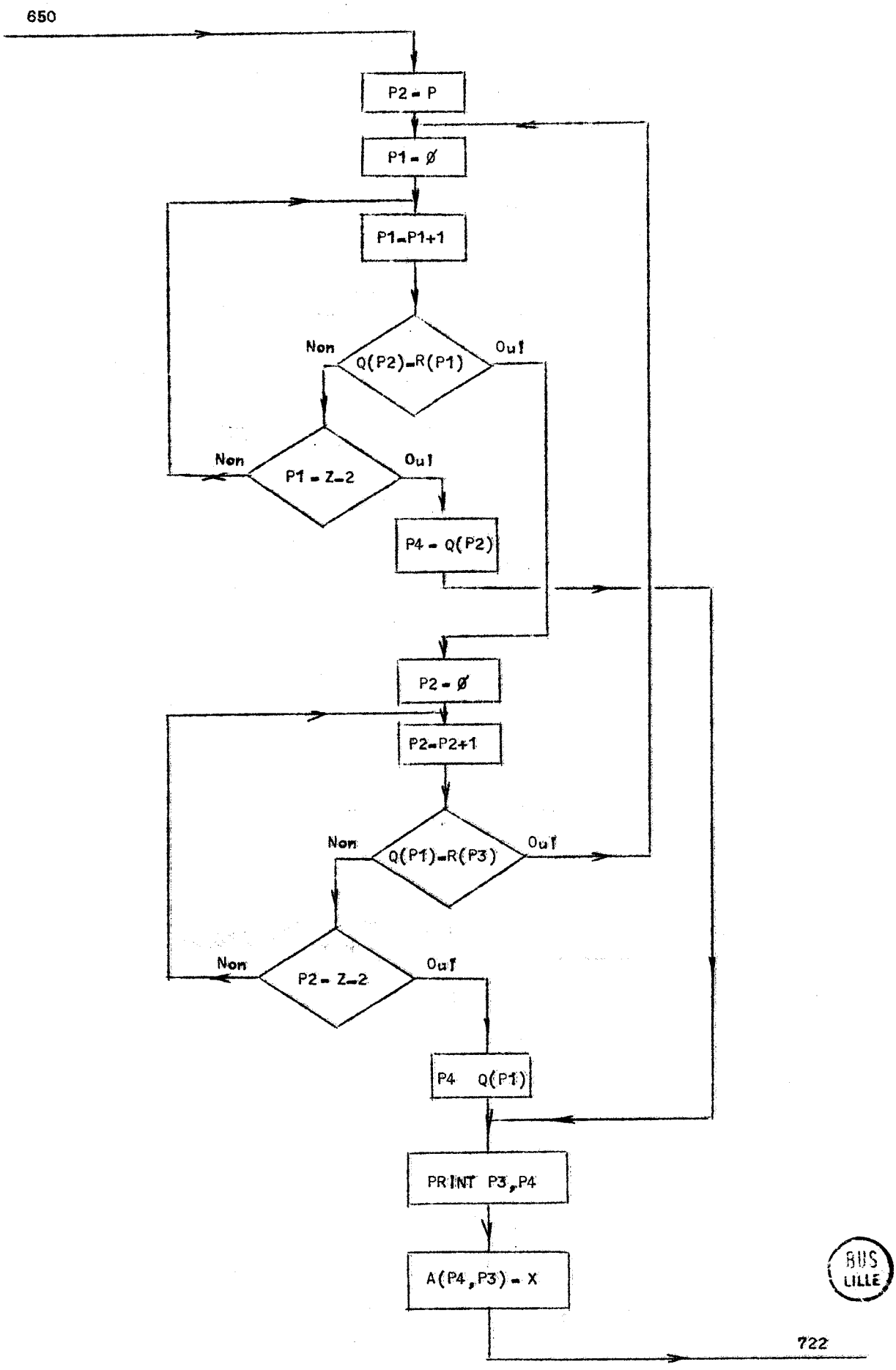

200
I = ø
I=I+1
P = 0
P=P+1
R(P) = I
Non
Oui
P=Z-2
Non
Oui
F=1E+38
J = ø
J=J+1
P = ø
P=P+1
Q(P) = J
Non
Oui
P=Z-2
Non
Oui
A(I,J)<F
Non
Oui
A(I,J) = F
J = Z
Non
Oui
S=S+F
T = ø
T=T+1
A(I,T) = A(I,T) - F
T = Z
Non
Oui
I = Z
Non
Oui
J = ø
J=J+1
P = ø
P=P+1
Q(P) = J
Non
Oui
380
P=Z-2
Non
Oui
F = ∞
315

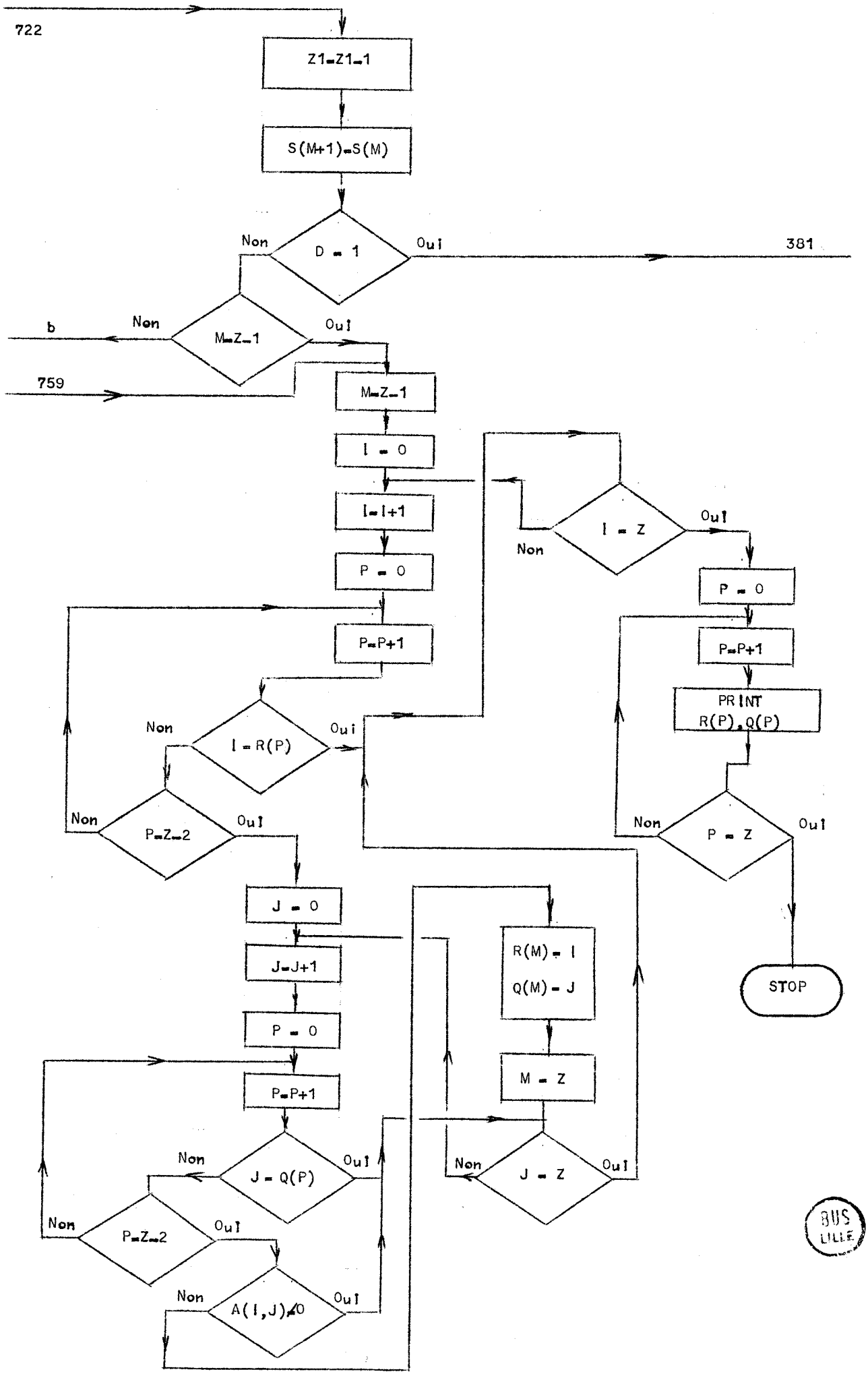

315
I = ø
I=I+1
P = ø
P=P+1
Non
I = R(P)
Oui
Non
P=Z-2
Oui
Non
A(I,J) F
Oui
F = A(I,J)
Non
I = Z
Oui
S=S+F
S(M) = S
T = ø
T=T+1
A(I,J)=A(I,J)-F
Non
T = Z
Oui
380
=1
Non
J = Z
Oui
Non
D1 = ø
Oui
390
I = ø
I=I+1
J = ø
J=J+1
C(I,J)=A(I,J)
Non
J = Z
Oui
Non
I = Z
Oui
D1 = ø
Non
D = ø
Oui
85
95
BUS
LILLE

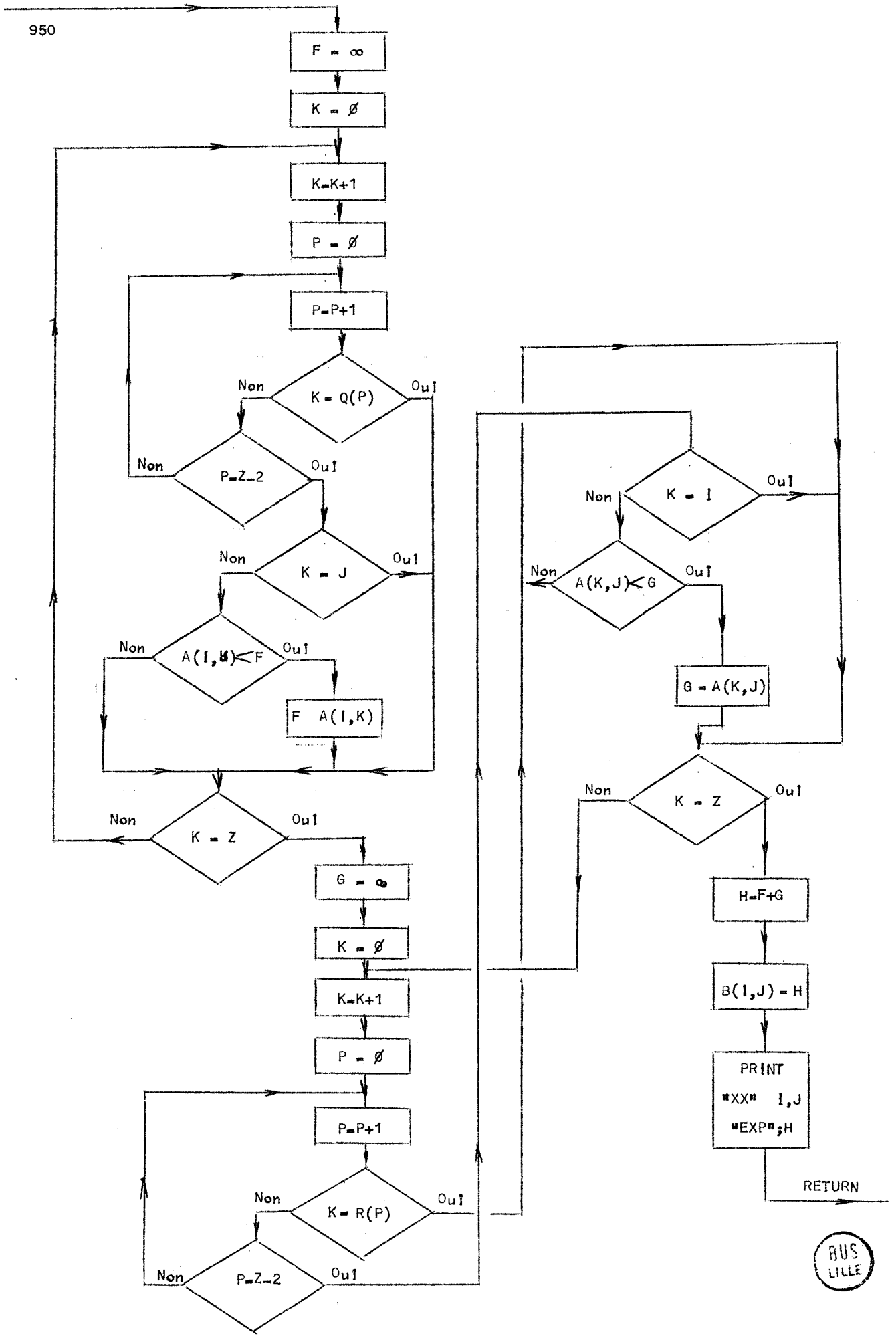

390
I = Ø
I=I+1
J = Ø
J=J+1
D(I,J)=A(I,J)
Non
J = Z
Oui
Non
I = Z
Oui
400
I = 0
I=I+1
P = Ø
P=P+1
Non
I = R(P)
Oui
Non
P=Z-2
Oui
Non
Z1 = 2
Oui
D = Ø
J = Ø
J=J+1
P = Ø
P=P+1
D1 = 1
Non
D = 1
Oui
85
95
Non
J = Q(P)
Oui
Non
P=Z-2
Oui
Non
A(I,J) = Ø
Oui
950
RETURN
Non
J = Z
Oui
Non
I = Z
Oui
P(M) = Ø
452
BUS LILLE


452
PRINT
GIVE I J
INPUT
R(M) Q(M)
P(M)=
B(R(M) , Q(M))
I = Ø
I=I+1
J = Ø
J=J+1
B(I , J)=Ø
Non
J = Z
Oui
Non
I = Z
Oui
PRINT"TEST"
M , R(M) Q(M)
518
Non
M <2
Oui
PRINT S(M-1)
S(M) P(M-1)
PRINT
R(M) Q(M)
P = Ø
Oui
Z1 = 2
Non
P=P+1
R(M) = Q(M)
550
Non
P=Z-2
Oui
P3=R(M)
600
759
BUS
LILLE


550
P2 = P
P1 = ∅
P1=P1+1
Non
R(P2) = Q(P1)
Oui
Non
P1 = Z-2
Oui
P3 = R(P2)
P2 = ∅
P2=P2+1
Non
R(P1)=Q(P2)
Oui
Non
P2 = Z-2
Oui
P3 = R(P1)
600
P = 0
P=P+1
Non
Q(M)=R(P)
Oui
65∅
Non
P=Z-2
Oui
P4=Q(M)
696


650
P2 = P
P1 = Ø
P1=P1+1
Non
Q(P2)=R(P1)
Oui
Non
P1 = Z-2
Oui
P4 = Q(P2)
P2 = Ø
P2=P2+1
Non
Q(P1)=R(P3)
Oui
Non
P2 = Z-2
Oui
P4 Q(P1)
PRINT P3,P4
A(P4,P3) = X
722


722
Z1=Z1-1
S(M+1)=S(M)
D = 1
Non
Oui
381
b
Non
M=Z-1
Oui
759
M=Z-1
I = 0
I=I+1
P = 0
P=P+1
Non
I = R(P)
Oui
Non
P=Z-2
Oui
J = 0
J=J+1
P = 0
P=P+1
Non
J = Q(P)
Oui
Non
P=Z-2
Oui
Non
A(I,J)≠0
Oui
I = Z
Non
Oui
P = 0
P=P+1
PRINT
R(P), Q(P)
Non
P = Z
Oui
STOP
R(M) = I
Q(M) = J
M = Z
Non
J = Z
Oui


950
F = ∞
K = ∅
K=K+1
P = ∅
P=P+1
K = Q(P)
Non
Oui
P=Z-2
K = J
A(I,K)<F
F A(I,K)
K = Z
G = ∞
K = ∅
K=K+1
P = ∅
P=P+1
K = R(P)
K = I
A(K,J)<G
G = A(K,J)
H=F+G
B(I,J) = H
PRINT
"XX" I,J
"EXP";H
RETURN

ANNEXE A2

```
8     DIM D(10,10)
10    DIM A(10,10),B(10,10)
11    DIM C(10,10)
12    DIM R(10),Q(10)
13    DIM P(10),S(10)
20    DATA


29    DATA
30    X = 1E + 38
33    Z = 9
34    Z1 = 9
35    FOR P = 1 TO Z - 2
36    R(P) = Ø
37    Q(P) = Ø
38    NEXT P
40    FOR I = 1 TO Z
45    FOR J = 1 TO Z
50    READ V
60    A(I,J) = V
70    NEXT J
80    NEXT I
82    D = Ø
85    FOR M = 1 TO Z - 1
87    S = S(M)
88    IF D = 1 THEN 95
90    E = Ø
95    E = E + 1
100   IF D = Ø THEN 11Ø
101   M = M + 2
102   S = S(M)
110   PRINT  "M",M
111   FOR I = 1 TO Z
113   FOR P = 1 TO Z - 2
115   IF R(P) = I THEN 17Ø
117   NEXT P
120   FOR J = 1 TO Z
123   FOR P = 1 TO Z - 2
```

```
125   IF Q(P) = J THEN 15Ø
127   NEXT P
130   IF A(I,J) = X THEN 145
140   PRINT A(I,J)
142   GOTO 15Ø
145   PRINT "X"
150   NEXT J
160   PRINT
170   NEXT I
175   PRINT S
180   PRINT
190   ON E GOTO 2ØØ,4ØØ
200   FOR I = 1 TO Z
201   FOR P = 1 TO Z - 2
202   IF R(P) = I THEN 28Ø
203   NEXT P
205   F = 1E + 38
210   FOR J = 1 TO Z
211   FOR P = 1 TO Z - 2
212   IF J = Q(P) THEN 23Ø
213   NEXT P
220   IF A(I,J)<F THEN F = A(I,J)
230   NEXT J
235   S = S + F
240   FOR T = 1 TO Z
250   A(I,T) = A(I,T) - F
260   NEXT T
280   NEXT I
300   FOR J = 1 TO Z
301   FOR P = 1 TO Z - 2
302   IF Q(P) = J THEN 38Ø
303   NEXT P
310   F = 1E + 38
315   FOR I = 1 TO Z
316   FOR P = 1 TO Z - 2
317   IF I = R(P) THEN 33Ø
318   NEXT P
320   IF A(I,J)<F THEN F = A(I,J)
330   NEXT I
340   S = S + F
```

```
345     S(M) = S
350     FOR T = 1 TO Z
360     A(T,J) = A(T,J) - F
370     NEXT T
380     NEXT J
381     IF D1 = Ø THEN 39Ø
382     FOR I = 1 TO Z
383     FOR J = 1 TO Z
384     C(I,J) = A(I,J)
385     NEXT J
386     NEXT I
387     D1 = Ø
388     IF D = 1 THEN 85
389     GOTO 95
390     FOR I = 1 TO Z
391     FOR J = 1 TO Z
392     D(I,J) = A(I,J)
393     NEST J
394     NEXT I
395     D1 = 1
396     IF D = 1 THEN 85
398     GOTO 95
400     FOR I = 1 TO Z
401     FOR P = 1 TO Z - 2
402     IF I = R(P) THEN 44Ø
403     NEXT P
404     IF Z1 = 2 THEN 518
405     D = Ø
410     FOR J = 1 TO Z
411     FOR P = 1 TO Z - 2
412     IF J = Q(P) THEN 43Ø
413     NEXT P
420     IF A(I,J) = Ø THEN GOSUB 95Ø
430     NEXT J
440     NEXT I
450     P(M) = Ø
452     PRINT "GIVE I AND J"
```

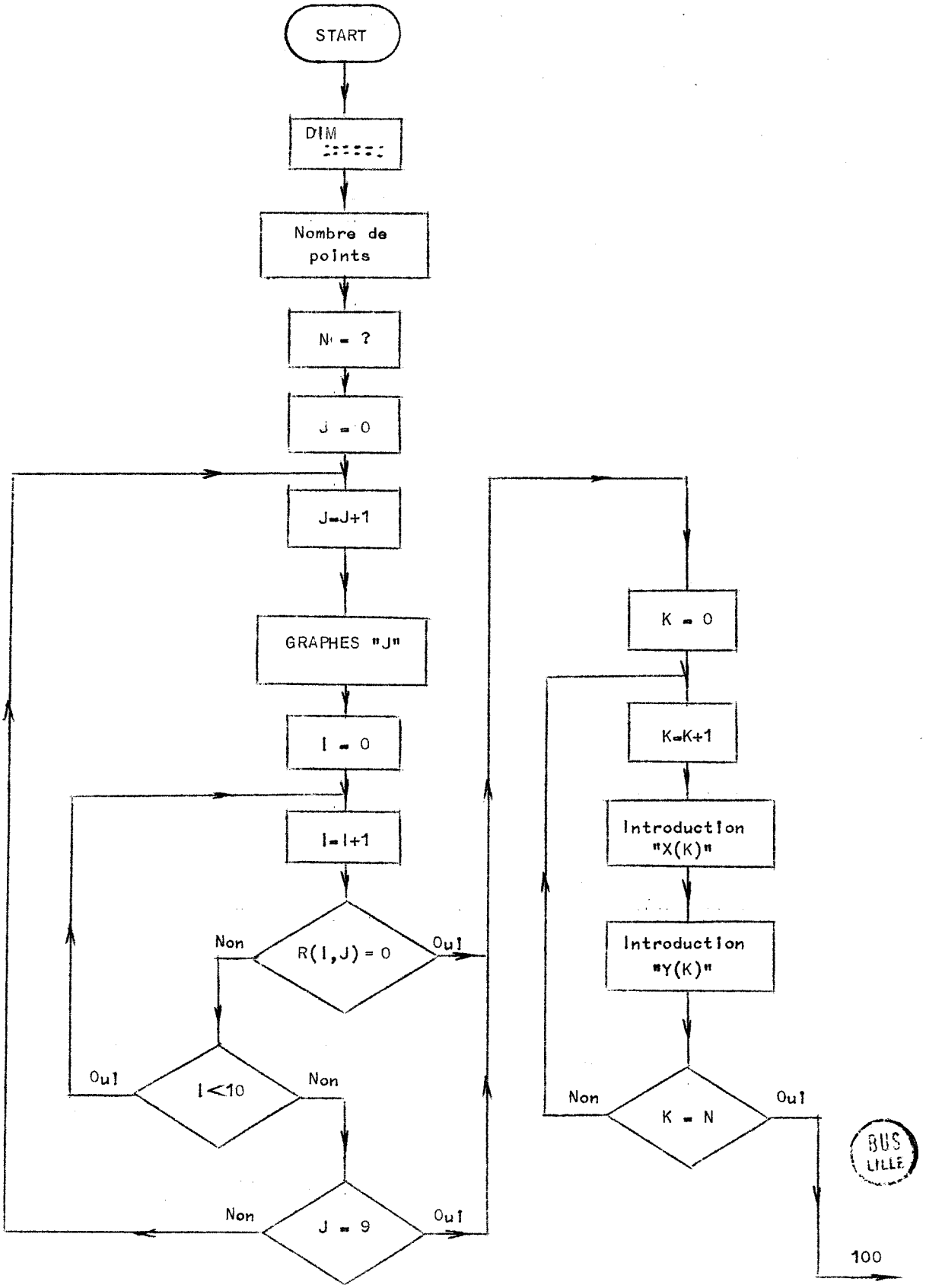
```
453   INPUT R(M),Q(M)
454   P(M) = B(R(M),Q(M))
455   FOR I = 1 TO Z
456   FOR J = 1 TO Z
457   B(I,J) = Ø
458   NEXT J
459   NEXT I
460   GOTO 516
516   PRINT  "TEST" M,R(M),Q(M)
517   IF M<2 THEN 526
518   PRINT S(M - 1),S(M),P(M - 1)
519   IF Z1 = 2 THEN 759
526   PRINT R(M),Q(M)
530   FOR P = 1 TO Z - 2
535   IF R(M) = Q(M) THEN 55Ø
540   NEXT P
545   P3 = R(M)
546   GOTO 6ØØ
550   P2 = P
555   FOR P1 = 1 TO Z - 2
560   IF R(P2) = Q(P1) THEN 58Ø
565   NEXT P1
570   P3 = R(P2)
575   GOTO 6ØØ
580   FOR P2 = 1 TO Z - 2
585   IF R(P) = Q(P) THEN 555
590   NEXT P2
595   P3 = R(P1)
596   GOTO 6ØØ
600   FOR P = 1 TO Z - 2
635   IF Q(M) = R(P) THEN 65Ø
640   NEXT P
645   P4 = Q(M)
646   GOTO 696
650   P2 = P
655   FOR P1 = 1 TO Z - 2
```

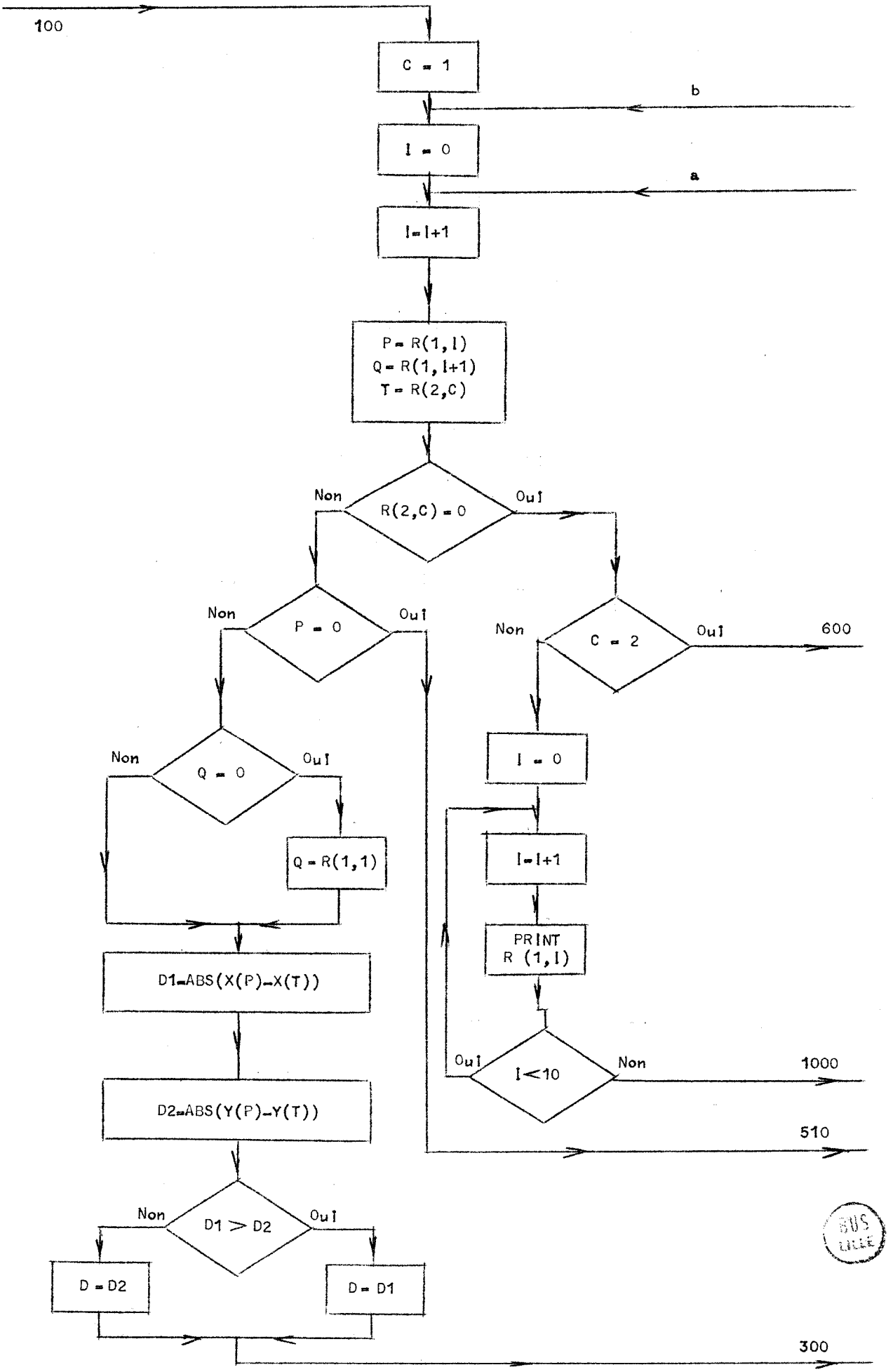
```
660   IF Q(P2) = R(P1) THEN 68Ø
665   NEXT P1
670   P4 = Q(P2)
675   GOTO 696
680   FOR P2 = 1 TO Z - 2
685   IF Q(P1) = R(P2) THEN 655
690   NEXT P2
695   P4 = Q(P1)
696   PRINT P3,P4
697   A(P4,P3) = X
698   GOTO 722
722   Z1 = Z1 - 1
748   S(M + 1) = S(M)
749   IF D = 1 THEN 381
750   NEXT M
759   M = Z - 1
760   FOR I = 1 TO Z
762   FOR P = 1 TO Z - 2
764   IF I = R(P) THEN 78Ø
766   NEXT P
768   FOR J = 1 TO Z
770   FOR P = 1 TO Z - 2
771   IF J = Q(P) THEN 778
772   NEXT P
773   IF A(I,J)<>Ø THEN 778
775   R(M) = I
776   Q(M) = J
777   M = Z
778   NEXT J
780   NEXT I
800   FOR P = 1 TO Z
802   PRINT R(P),Q(P)
804   NEXT P
900   END
950   F = 1E + 38
960   FOR K = 1 TO Z
961   FOR P = 1 TO Z - 2
```

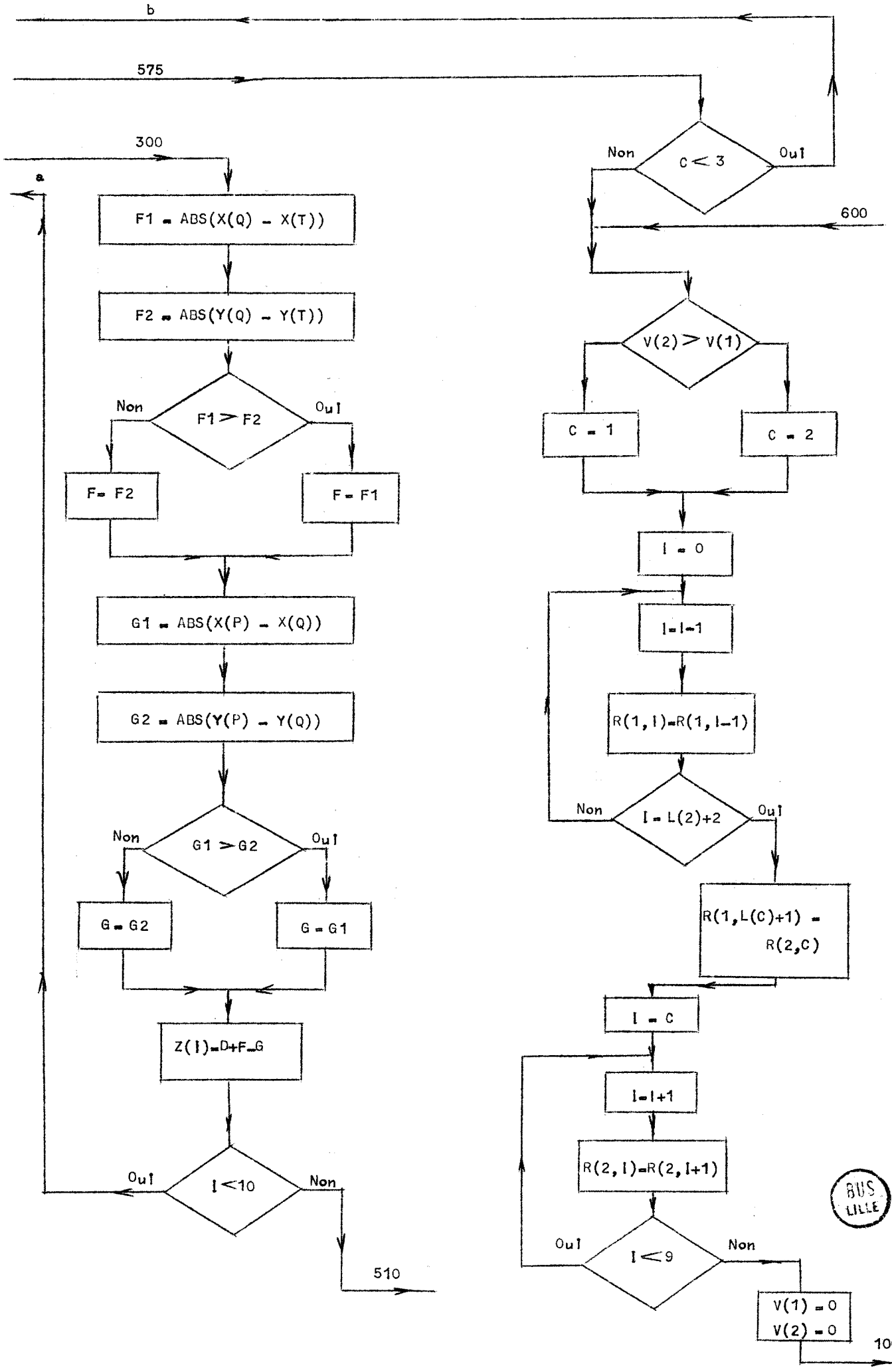
```
963     IF K = Q(P) THEN 98Ø
964     NEXT P
970     IF K = J THEN 98Ø
975     IF A(I,K)< F THEN F = A(I,K)
980     NEXT K
985     G = 1E + 38
990     FOR K = 1 TO Z
991     FOR P = 1 TO Z - 2
993     IF K = R(P) THEN 1Ø1Ø
994     NEXT P
995     IF K = I THEN 1Ø1Ø
1000    IF A(K,J)< G THEN G = A(K,J)
1010    NEXT K
1020    H = F + G
1030    B(I,J) = H
1035    PRINT  "XX" ,I,J," EXP =" ; H
1040    RETURN
```

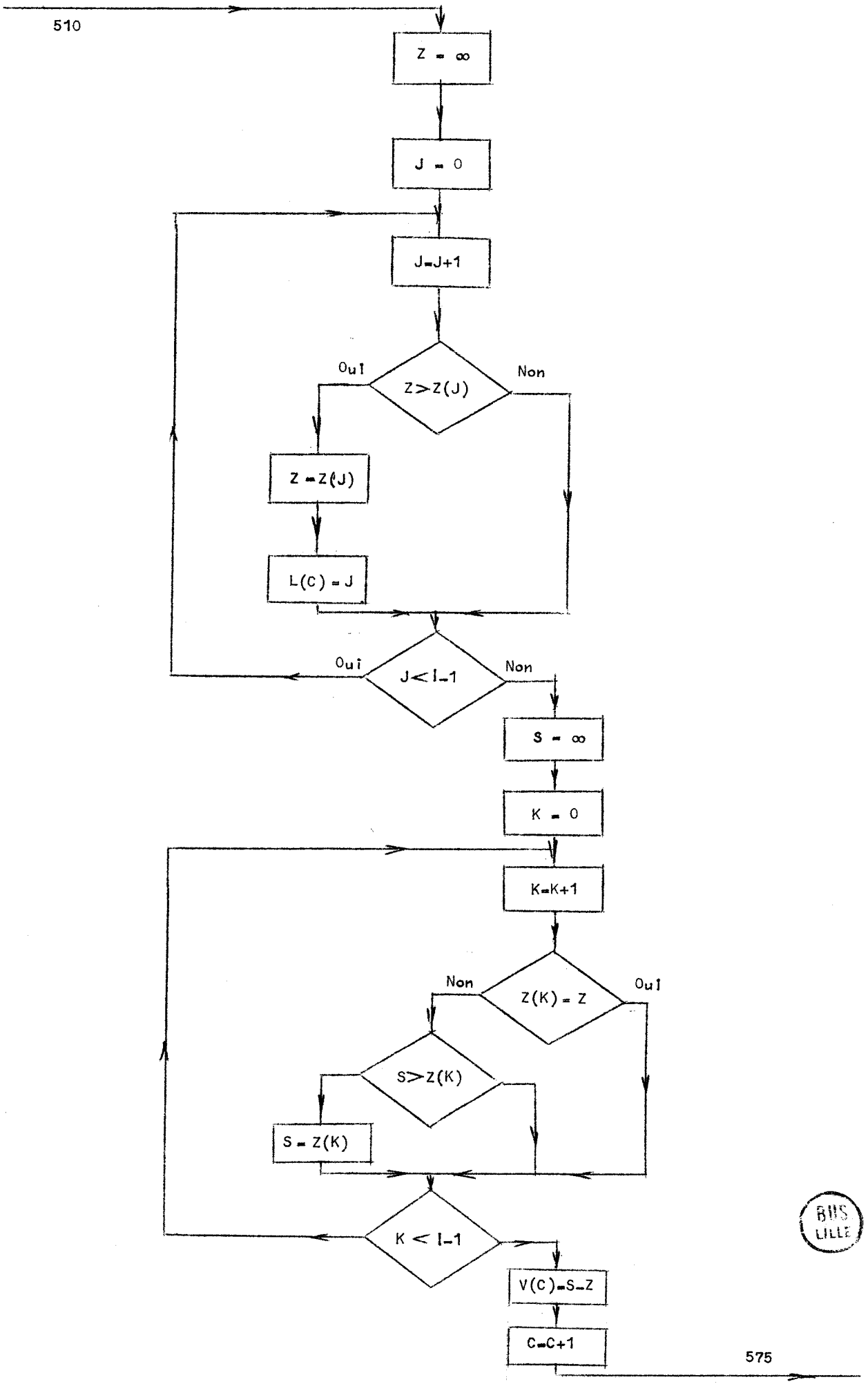
****

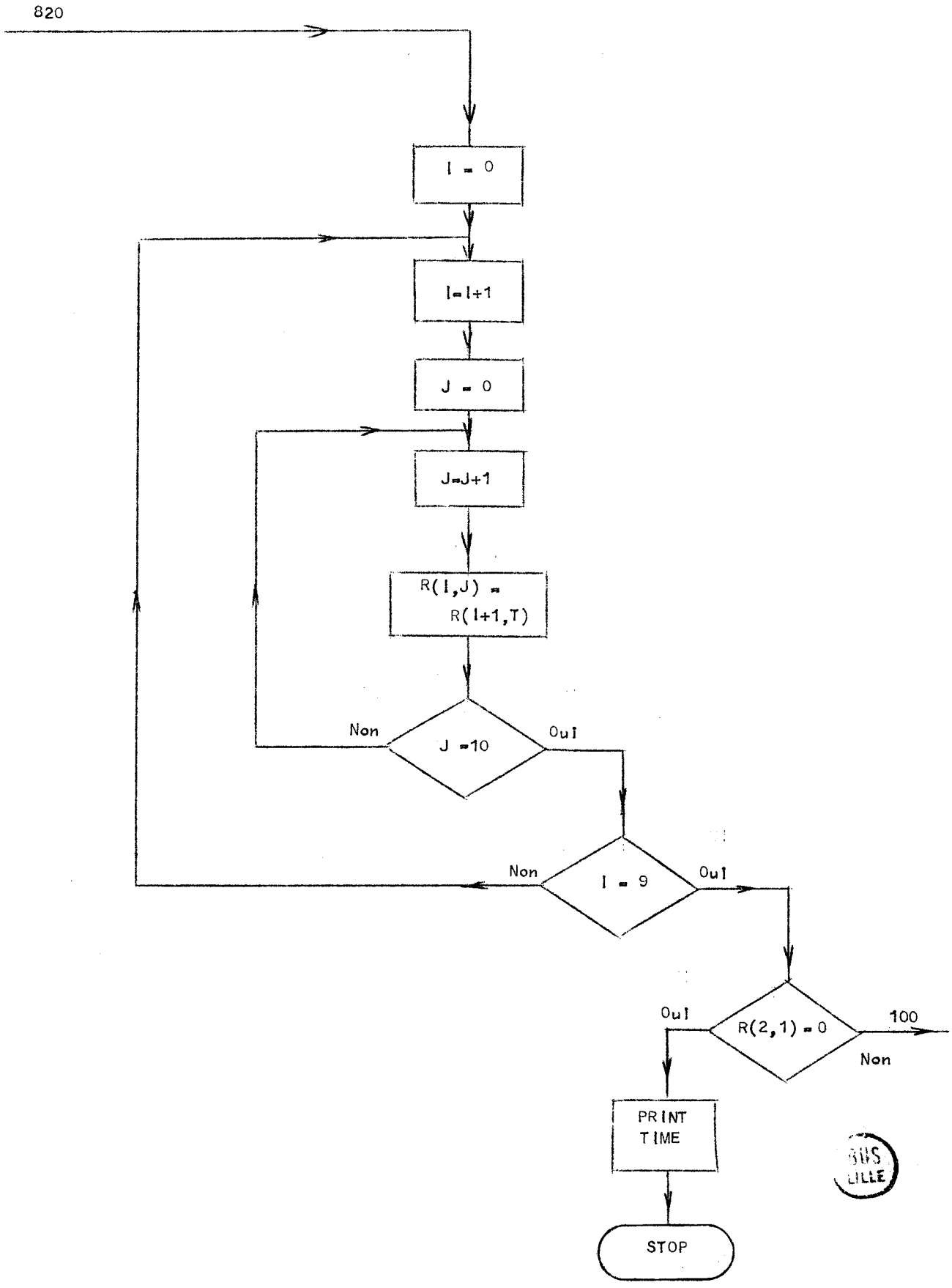
ANNEXE B1

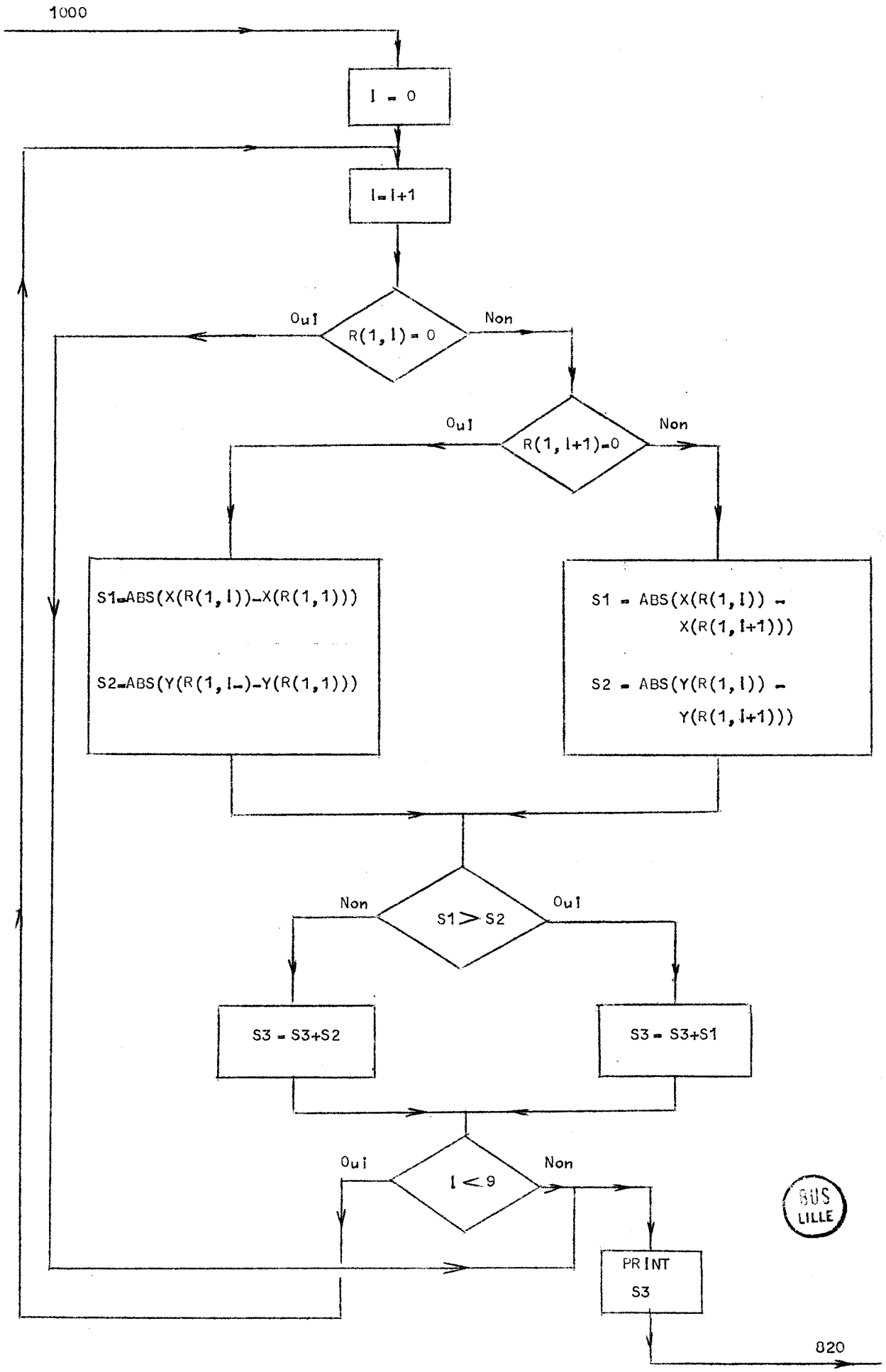

START
DIM
Nombre de points
N = ?
J = 0
J=J+1
GRAPHES "J"
I = 0
I=I+1
R(I,J) = 0
Non
Oui
I<10
Oui
Non
J = 9
Non
Oui
K = 0
K=K+1
Introduction "X(K)"
Introduction "Y(K)"
K = N
Non
Oui
100


100
C = 1
b
I = 0
a
I=I+1
P = R(1,I)
Q = R(1,I+1)
T = R(2,C)
Non
R(2,C) = 0
Oui
Non
P = 0
Oui
Non
C = 2
Oui
600
Non
Q = 0
Oui
Q = R(1,1)
I = 0
I=I+1
PRINT
R (1,I)
Oui
I<10
Non
1000
510
D1=ABS(X(P)-X(T))
D2=ABS(Y(P)-Y(T))
Non
D1 > D2
Oui
D = D2
D = D1
300


b
575
300
a
F1 = ABS(X(Q) - X(T))
F2 = ABS(Y(Q) - Y(T))
Non
F1 > F2
Oui
F= F2
F= F1
G1 = ABS(X(P) - X(Q))
G2 = ABS(Y(P) - Y(Q))
Non
G1 > G2
Oui
G = G2
G = G1
Z(I)=D+F-G
Oui
I <10
Non
510
Non
C < 3
Oui
600
V(2) > V(1)
C = 1
C = 2
I = 0
I=I-1
R(1,I)=R(1,I-1)
Non
I = L(2)+2
Oui
R(1,L(C)+1) =
R(2,C)
I = C
I=I+1
R(2,I)=R(2,I+1)
Oui
I <9
Non
V(1) = 0
V(2) = 0
100
BUS
LILLE


510
Z = ∞
J = 0
J=J+1
Oui
Z>Z(J)
Non
Z = Z(J)
L(C) = J
Oui
J<I-1
Non
S = ∞
K = 0
K=K+1
Non
Z(K) = Z
Oui
S>Z(K)
S = Z(K)
K < I-1
V(C)=S-Z
C=C+1
575


820
I = 0
I=I+1
J = 0
J=J+1
R(I,J) =
R(I+1,T)
Non
J =10
Oui
Non
I = 9
Oui
Oui
R(2,1) = 0
100
Non
PRINT
TIME
STOP


1000
I = 0
I=I+1
Oui
R(1, I) = 0
Non
Oui
R(1, I+1)=0
Non
S1=ABS(X(R(1,I))-X(R(1,1)))
S2=ABS(Y(R(1,I-)-Y(R(1,1)))
S1 = ABS(X(R(1,I)) - X(R(1,I+1)))
S2 = ABS(Y(R(1,I)) - Y(R(1,I+1)))
Non
S1 > S2
Oui
S3 = S3+S2
S3 = S3+S1
Oui
I < 9
Non
PRINT S3
820

ANNEXE B2

```
5     DIM R(10,10),X(10),Y(10),Z(10)L(2)
6     DIM V(2)
20    PRINT "Nombre de points"
21    INPUT N
25    FOR J = 1 TO 9
30    PRINT "GRAPHE" ; J
35    FOR I = 1 TO 10
40    INPUT R(J,I)
41    IF R(J,I) = Ø THEN 55
45    NEXT I
50    NEXT J
55    FOR K = 1 TO N
60    PRINT "GIVE X ("; K ; ")"
65    INPUT X(K)
70    PRINT "GIVE Y(" ; K ; ")"
75    INPUT Y(K)
80    NEXT K
90    TI $ = "000000"
100   C = 1
110   FOR I = 1 TO 10
115   P = R(1,I)
120   Q = R(1,I+1)
125   T = R(2,C)
130   IF R(2,C) = Ø THEN 138
135   GOTO 155
138   IF C = 2 THEN 6ØØ
140   GOTO 8ØØ
155   IF P = Ø THEN 51Ø
160   IF Q = Ø THEN Q = R(1,1)
165   D1 = ABS(X(P) - X(T))
170   D2 = ABS(Y(P) - Y(T))
175   IF D1 > D2 THEN 2ØØ
180   D = D2
190   GOTO 3ØØ
200   D = D1
300   F1 = ABS(X(Q) - X(T))
305   F2 = ABS(Y(Q) - Y(T))
310   IF F1 > F2 THEN 35Ø
```

```
315   F = F2
320   GOTO 4ØØ
350   F = F1
400   G1 = ABS(X(P) - X(Q))
405   G2 = ABS(Y(P) - Y(Q))
410   IF G1 > G2 THEN 45Ø
415   G = G2
420   GOTO 5ØØ
450   G = G1
500   Z(I) = D + F - G
505   NEXT I
510   Z = 1E + 38 (∞)
515   FOR J = 1 TO I - 1
520   IF Z > Z(J) THEN 522
521   GOTO 530
522   Z = Z(J)
525   L(C) = J
530   NEXT J
535   S = 1E + 38
540   FOR K = 1 TO I - 1
545   IF Z(K) = z THEN 555
550   IF S > Z(K) THEN S = Z(K)
555   NEXT K
560   V(C) = S - Z
570   C = C + 1
575   IF C < 3 THEN 11Ø
600   IF V(2) > V(1) THEN 65Ø
610   C = 1
615   GOTO 7ØØ
650  C = 2
700   FOR I = 10 TO L(C) + 2 STEP - 1
705   R(1,I) = R(1,I - 1)
710   NEXT I
715   R(1,L(C) + 1) = R(2,C)
720   FOR I = C TO 9
725   R(2,I) = R(2,I + 1)
730   NEXT I
740   V(1) = Ø
745   V(2) = Ø
750   GOTO 1ØØ
800   FOR I = 1 TO 1Ø
```

```
800     PRINT R(1,I);" ";
810     NEXT I
812     S3 = Ø
815     GOSUB 1ØØØ
820     FOR I = 2 TO 10
825     FOR J = 1 TO 10
830     R(I,J) = R(I + 1,J)
835     NEXT J
840     NEXT I
845     IF R(2,1) = Ø THEN 85Ø
847     GOTO 1ØØ
850     PRINT TI/60; "sec"
860     END
1000    FOR I = 1 TO 9
1002    IF R(1,I) = 0 THEN 1035
1003    IF R(1,I + 1) = Ø THEN 1022
1010    S1 = ABS(X(R(1,I)) - X(R(1,I + 1)))
1020    S2 = ABS(Y(R(1,I)) - Y(R(1,I + 1)))
1021    GOTO 1025
1022    S1 = ABS(X(R(1,I)) - X(R(1,1)))
1023    S2 = ABS(Y(R(1,I)) - Y(R(1,1)))
1025    IF S1 > S2 THEN 1029
1026    S3 = S3 + S2
1027    GOTO 1030
1029    S3 = S3 + S1
1030    NEXT I
1035    PRINT S3
1040    RETURN
```

****
*****
*

REFERENCES

(1) ACKOFF - SASIENI

Fundamentals of Operations Research.
Wiley International Edition (1968)

(2) CHRISTOFIDES N

Graph Theory
An algorithmic approach.
Academic Press (1975)

(3) COOPER - STEINBERG

Introduction to Methods of Optimization
Saunders Company (1970)

(4) COTTLE RW - FRARUP J

Optimization Methods
English University Press Ltd London (1974)

(5) FAURE R

Précis de recherche opérationnelle
Dunod (1978)

(6) GOLDSTEIN E - YOUDINE D
Problèmes particuliers de la programmation linéaire
Editions de Moscou (1973)

(7) HERROELEN W

Heuristische Programmatie
Aurelia Books Leuven (1972)

(8) KAUFMANN

Introduction à la combinatorique en vue des applications
Dunod (1968)

(9) KAUFMANN - LABORDERE

Méthodes et Modèles de la recherche opérationnelle Tome 3
Dunod (1974)

(10) MILLAU C

Mathematical Programming
Wiley and Sons (1970)

(11) MITAL KV

Optimization Methods
Wiley-Eastern (1976)

(12) LEIBOVICI - MOULLE

Recherche opérationnelle Tome 1
Cours ESE (1972)

(13) NEMHAUSER - GARFINKEL

Integer Programming
Wiley and Sons (1972)

(14) ORE O

Les graphes et leurs applications
Collection 21 Sigma Dunod (1970)

(15) PRICE W

Graphs and Networks
London Butterworths (1971)

(16) SCOTT J

Combinatoral Programming
Spatial Analysis and Planning
Methuen and Co London (1971)

(17) TAHA HAMDY

Operations Research an introduction
Collier Mac Millan international edition (1978)

(18) TAHA HAMDY

Integer Programming
Academic Press (1975)

(19) TEN BROEKE

Operationele Research
Samson uitgeverij Brussel (1977)

(20) VAN FRAJER HELMUT EISELT HORST

Operations Research Handbook
De Gruyter (1977)

(21) WAGNER H

Principles of Operation Research
Prentice/Hall international (1972)

(22) WATSON - GANDY - EILON - CHRISTOFIDES

Distribution Management Mathematical Modelling and Spatial analysis
Griffin London (1970)

(23) ZOUTENDIJK G

Mathematical Programming Methods
North - Holland Publishing Company (1976)

(24) BARACHET

Graphic solution of the travelling salesman problem
Op. Research 5 (1957)

(25) BOWMAN - FETTER

Analysis of production and Operations Management
R.D. Irwin Inc Homewood Illinois (1967)

(26) DANTZIG - FULKERSON - JOHNSON

Solution of a large scale travelling salesman problem
Op. Research 2 (1954)

(27) EASTMAN W

Linear programming with pattern constraints
PH D Dissertation Havard (1958)

(28) FLOOD M

The travelling salesman problem
Op. Research 4 (1956)

(29) GILMORE - GOMORY

Sequencing a one state variable machine
A solvable case of the travelling salesman problem
Op. Research 12 (1964)

(30) HELD - KARP

A dynamic programming approach to sequencing problems
SIAM Journal 10 (1962)

(31) KARG - THOMPSON

A heuristic approach to solving travelling salesman problem
Mgmt Sci 10 (1964)

(32) KUEHN A

Heuristic programming: A useful technique .../... for marketing.
Procceding of the 45° National Conference of the American Marketing Association
June 20, 21, 22 (1962)

(33) KRUSKAL J

On the shortest spanning subtree of a graph and the travelling salesman Problem . Proc. AM Math. Soc. 7 (1956) .

(34) LIN S

Computer solutions of the travelling salesman problem
Bell-systems technique J 44 (1965)

(35) LITTLE - MURTY - SWEEWEY

An algorithme for the travelling salesman problem
Op. Research 11 (1963)

(36) RAYMOND.TC.

Heuristic algorithm for the travelling salesman problem
IBM Journal of Research Development July (1969)

(37) REITER.- SHERMAN.

Discrete Optimizing
SIAM Revue 13 (1965)

(38) ROBERTS.- FLORES.

An enginiering approach to the travelling salesman problem
Mgmt Sc. 13 (1966)

(39) SHAPIRO D.

Algorithms for the solution of the optimal cost travelling salesman problem
ScD. Thesis Washington University St Louis (1966)

(40) WYSKIDA - IGNIZIO - WILHELM.

A rationale for heuristic program selection and evaluation
Industrial Engineering 4 (1) January (1972)

****
*****
*