Le problème d'agencement des ressources à l'intérieur des cellules des systèmes de production

Thomas Hamann

N° d'ordre :

ACADEMIE DE NANCY-METZ

UNIVERSITE DE METZ, FACULTE DES SCIENCES
L'UFR MATHEMATIQUE, INFORMATIQUE, MECANIQUE, AUTOMATIQUE

THESE

présentée à

L'UNIVERSITE DE METZ

pour l'obtention du titre de

DOCTEUR

Spécialité :

AUTOMATIQUE

par

Thomas HAMANN

Sujet de la thèse :

LE PROBLEME D'AGENCEMENT DES RESSOURCES A L'INTERIEUR DES CELLULES DES SYSTEMES DE PRODUCTION

Soutenue le 14 Décembre 1992 devant le Jury composé de :

Rapporteurs :	Mr B. MUTEL Mr J. FAVREL
Examinateurs :	Mme M. LUMBRERAS Mr C. LAURENT Mr J.-M. PROTH Mr F. VERNADAT Mr G. WENNER

N° d'ordre :

ACADEMIE DE NANCY-METZ

UNIVERSITE DE METZ, FACULTE DES SCIENCES
L'UFR MATHEMATIQUE, INFORMATIQUE, MECANIQUE, AUTOMATIQUE

THESE

présentée à

L'UNIVERSITE DE METZ

pour l'obtention du titre de

DOCTEUR

Spécialité :

AUTOMATIQUE

par

Thomas HAMANN

Sujet de la thèse :

LE PROBLEME D'AGENCEMENT DES RESSOURCES A L'INTERIEUR DES CELLULES DES SYSTEMES DE PRODUCTION

Soutenue le 14 Décembre 1992 devant le Jury composé de :

Rapporteurs : Mr B. MUTEL
Mr J. FAVREL

Examinateurs : Mme M. LUMBRERAS
Mr C. LAURENT
Mr J.-M. PROTH
Mr F. VERNADAT
Mr G. WENNER

LE PROBLEME D'AGENCEMENT DES RESSOURCES A L'INTERIEUR DES CELLULES DES SYSTEMES DE PRODUCTION

Thèse de doctorat de Thomas HAMANN
soutenue le 14 décembre 1992
à l'Université de Metz

Sujet :

"Le problème de l'agencement des ressources à l'intérieur des cellules des systèmes de production"

Résumé :

Dans cette thèse, nous nous intéressons à l'agencement des ressources à l'intérieur des cellules dans un système de production automatisé. La solution présentée se structure en deux étapes.

La première étape consiste à déterminer le système de transport et le (ou les) type(s) d'agencement de base possible(s) de la cellule. Ces procédures de choix sont basées sur les caractéristiques (i) des produits, (ii) de leurs gammes de fabrication, (iii) des machines, et (iv) de la cellule. Un système expert a été développé pour aider à ces choix.

La seconde étape consiste à optimiser l'agencement (c'est-à-dire à minimiser le trafic entre les machines de la cellule) compte tenu des choix précédents et des contraintes fonctionnelles et physiques et tout particulièrement des distances réelles entre les machines. Cette seconde étape demande la résolution de problèmes de recherche opérationnelle. Nous avons en particulier utilisé la technique du Recuit Simulé.

Mots-clés :

Agencement intra-cellulaire, Moyens de transport, Configuration de cellules de fabrication, Système expert, Minimisation du trafic, Optimisation.

Thesis of Thomas HAMANN
defended December 14, 1992
at the University of Metz

Title

"The intra-cell layout problem in automated manufacturing systems"

Abstract

The machine layout problem for manufacturing cells (or intra-cell layout problem) of an automated manufacturing system is addressed. The approach presented is divided into two main parts.

The first part concerns the selection of the materials handling system and the possible machine layout type. This procedure is based on the characteristics of: (i) the products (which belong to the same product family), (ii) their manufacturing processes, (iii) the machines, and (iv) the cell. An expert system has been developed for this part.

The second part concerns the physical arrangement of the machines inside the manufacturing cell in order to minimize the intra-cell traffic while respecting the physical constraints (distances between the machines, product constraints, technological constraints, user preference, etc.). This part has been solved using Operations Research algorithms and especially the simulated annealing approach. The originality of the approach is to consider the cell layout, the materials handling system and the real distance followed by parts between the machines for the final arrangement of machines.

Keywords

Intra-cell layout problem, Materials handling systems, Layout type, Expert system, Minimize the intra-cell traffic, Operations Research

REMERCIEMENTS

Cette thèse a été réalisée au sein du projet SAGEP de l'INRIA-Lorraine, où j'ai pu bénéficier d'un encadrement efficace et d'une ambiance chaleureuse.

Ce travail a été effectué sous la direction scientifique de Monsieur Jean-Marie PROTH, Directeur de Recherche à l'INRIA. Je tiens à lui exprimer toute ma reconnaissance pour m'avoir accueilli dans son équipe, pour la confiance qu'il m'a témoignée et pour l'aide qu'il m'a apportée.

Je tiens à exprimer toute ma reconnaissance à Monsieur Bernard MUTEL, Professeur et Directeur du LRPS à l'Ecole Nationale Supérieure des Arts et Industries de Strasbourg, et Monsieur Joël FAVREL, Professeur et Directeur du GRASP à l'INSA de Lyon pour avoir accepté d'être rapporteurs de cette thèse et membres du jury.

Je remercie vivement les personnalités scientifiques qui se sont intéressées à mon travail et qui ont bien voulu me faire l'honneur de participer à ce jury :

Madame Martine LUMBRERAS, Professeur à l'Univeristé de Metz,

Monsieur Claude LAURENT, Professeur et Directeur du LAEI à l'Université de Metz,

Monsieur Gregor WENNER, Professeur à l'Ecole Supérieure des Techniques et d'Economie de la Sarre (Hochschule für Technik und Wirtschaft des Saarlandes).

Je souhaite exprimer mes plus vifs remerciements à Monsieur François VERNADAT, Chargé de Recherche à l'INRIA, qui m'a suivi tout au long de cette thèse, qui a consacré de nombreuses heures à l'amélioration de ce document et qui a bien voulu me faire l'honneur de participer au jury.

Mes plus vifs remerciements vont aussi à Monsieur Abdelghani SOUILAH, Doctorant au sein du projet SAGEP, pour les discussions fructueuses que nous avons eues, ainsi que pour ses importants travaux de programmation.

Je remercie également mes camarades chercheurs, ingénieur et secrétaires pour leur aide et leur soutien amical.

TABLE DES MATIERES

Introduction 1

Chapitre I Formulation du problème de l'agencement 3
I.1 Problème général d'agencement 3
I.1.1 Présentation générale du problème et définitions 3
I.1.2 Méthodes de résolution 4
I.1.3 Modèles mathématiques et algorithmes de résolution 5
I.2 Problèmes de l'agencement des systèmes de production 13
I.2.1 Introduction 13
I.2.2 Présentation du problème 13
I.2.3 Les objectifs 14
I.2.4 Etat de l'art / Méthodes et logiciels existants 15
I.2.5 Conclusion 21

Chapitre II Nouvelle approche de résolution du problème de l'agencement des systèmes de fabrication 22
II.1 Introduction 22
II.2 Conception des cellules de fabrication 24
II.2.1 La méthode GPM 27
II.2.2 Minimisation du trafic inter-cellules 30
II.2.3 Méthode basée sur le Recuit Simulé 30
II.3 Agencement des ressources à l'intérieur des cellules 31
II.4 Agencement des cellules dans l'atelier de fabrication 32
II.5 Conclusion 35

Chapitre III Nouvelle approche pour l'agencement des ressources à l'intérieur des cellules de fabrication 36
III.1 Introduction 36
III.2 Formulation du problème 36
III.2.1 Quelques définitions 36
III.2.2 Déscription du problème 38
III.2.3 Les données 39
III.2.4 Les moyens de transport et les configurations de base 44
III.2.5 Le critère d'optimisation 44

III.2.6 Les contraintes 45
III.2.7 Un algorithme connu de placement de machines 50
III.2.8 La méthodologie utilisée 52

III.3 Système Expert 55
III.3.1 Introduction 55
III.3.2 SMECI 55
III.3.3 Choix du moyen de transport 58
III.3.3.1 Types de moyens de transport 58
III.3.3.2 Méthode de résolution 59
III.3.3.3 Résultats 62
III.3.4 Choix du type d'agencement de base 62
III.3.4.1 Types d'agencement de base 62
III.3.4.2 Méthode de résolution 64
III.3.4.3 Résultats 67
III.4 Agencement et évaluation 69
III.4.1 Préliminaires 69
III.4.2 Méthode générale de résolution du problème 69
III.4.3 Formulation du problème pour chaque type de configuration de base 74
III.4.4 Discussions 88
III.4.5 Résultats 90

Chapitre IV Présentation des résultats 91

Conclusion Générale 119

Références 121

Annexe A
Recuit Simulé 130

Annexe B
Systèmes experts 134
1.1 Liste des règles 134
1.2 Listings des objets, des prototypes et des règles 163
2. Agencement et évaluation 181
2.1 Listing des programmes 181

INTRODUCTION

INTRODUCTION

Face à l'intensification de la concurrence internationale, l'économie mondiale bascule de plus en plus de l'économie d'échelle vers l'économie de diversification. Dans cette dernière, la production se caractérise par une grande variété de produits et de processus de fabrication, des systèmes de production complexes et un marché fortement fluctuant. Cela engendre de grandes difficultés dans la gestion des systèmes de production et nécessite des stratégies de gestion et de pilotage de plus en plus dynamiques, souples et automatisées.

Cette tendance vers des systèmes de fabrication souples et automatisés, connus sous le nom de **Systèmes Flexibles**, commence à montrer son efficacité et à progresser dans le monde industriel d'aujourd'hui. Elle tend principalement à satisfaire la demande dans les meilleurs délais, aux moindres coûts, et à un niveau de qualité élevé.

Parmi les aspects les plus récents qui caractérisent cette nouvelle révolution industrielle, on trouve ce qu'on appelle la **Production Cellulaire** (Cellular Manufacturing). Un grand nombre de systèmes sont le résultat de l'application de cette philosophie. En effet, de nombreuses études, conduites dans les pays les plus industrialisés, montrent que 80% à 90% de toutes les pièces fabriquées de nos jours sur des machines-outils peuvent être entièrement usinées dans des systèmes de fabrication cellulaires. Ces cellules de fabrication, définies selon des critères liés aux pièces à produire, doivent avoir une grande autonomie de fonctionnement.

Il s'agit donc de regrouper les ressources utiles en cellules, de sorte que les produits passent la plus grande partie, sinon la totalité, de leur temps de fabrication dans la même cellule. Cela permet d'éviter les transports de pièces qui compliquent la gestion, augmentent les en-cours et nuisent à la qualité. On cherche enfin à agencer les ressources dans les cellules, puis les cellules dans l'atelier, de manière à obtenir un trafic inter- et intra-cellulaire aussi faible que possible.

L'objectif de cette thèse est de trouver une approche de résolution du problème d'agencement des ressources à l'intérieur des cellules des systèmes de fabrication, qui tiendrait compte de la plupart des contraintes et caractéristiques physiques existantes.

Le travail que nous présentons dans ce mémoire est organisé en quatre chapitres.

Le chapitre I présente une formulation générale du problème d'agencement. Nous insistons plus particulièrement sur les problèmes d'agencement des systèmes de production et les logiciels existants.

Le chapitre II est consacré à une nouvelle approche de résolution du problème d'agencement des systèmes de production. Cette approche est divisée en trois étapes qui sont les suivantes :

- Conception des cellules de production ;

- Agencement des ressources à l'intérieur des cellules ;

- Agencement des cellules dans un atelier de fabrication.

Dans le chapitre III, nous nous intéressons à une nouvelle approche de résolution du problème d'agencement des ressources à l'intérieur des cellules. Nous développons successivement les points suivants, qui correspondent aux différentes étapes de cette approche :

- Formulation du problème ;

- Choix du moyen de transport ;

- Choix du type fondamental d'agencement ;

- Agencement des ressources à l'intérieur de la cellule et évaluation.

Le dernier chapitre est consacré à la présentation des résultats obtenus.

CHAPITRE I

FORMULATION DU PROBLEME D'AGENCEMENT

I.1 PROBLEME GENERAL D'AGENCEMENT

I.1.1 PRESENTATION GENERALE DU PROBLEME ET DEFINITIONS

Présentation

L'agencement "optimal" des systèmes est une préoccupation importante, non seulement dans le domaine des industries manufacturières, mais aussi dans d'autres domaines, notamment celui de l'urbanisme, où il s'agit d'ajuster les lieux d'habitation, de travail et de prestation de services aux besoins de l'homme, ou encore en électronique pour le placement des composants sur les cartes électroniques. On constate que ce problème est bien couvert dans la littérature. On le retrouve sous plusieurs noms, et cela suivant les spécificités du domaine d'application. Par exemple, on parlera de "Facility Layout", "Facility Allocation", "Layout Design", "Plant Layout", "Machine Layout", "Area Placement", "Circuit Layout", etc ... suivant le domaine d'application.

Du point de vue des domaines d'application, on peut dire que ce problème se présente sous deux formes bien distinctes :

1) "*L'agencement par partitionnement de la surface disponible*" qui se pose essentiellement dans le domaine de l'urbanisme,

2) "*L'agencement par placement des ressources sur la surface disponible*" dont les applications sont très variées :

   - agencement des ressources dans les hôpitaux ;
   - affectation des postes d'incendie dans les bâtiments de service (marchés, banques, bibliothèques, ...), de travail (bureaux, ...) ou d'habitation ;
   - agencement des cellules CMOS ou des macro-cellules dans les circuits VLSI (micro-électronique) ;
   - et, ce qui nous intéresse, **l'agencement des systèmes de fabrication.**

Définition

Par définition, nous dirons que nous avons à faire à un problème d'agencement chaque fois qu'il s'agit d'allouer un emplacement à chacun des objets d'un ensemble donné sur une surface connue ou non a priori, de façon à optimiser un critère donné, ou à trouver un bon compromis entre plusieurs critères.

Le problème de l'agencement de ressources dans les systèmes de fabrication est en général un problème d'agencement dans un espace de dimension 2. Dans la pratique, le problème peut se poser dans des espaces de dimension 1, 2 ou 3.

I.1.2 METHODES DE RESOLUTION

Les problèmes d'agencement pouvant être résolus à l'aide de méthodes exactes et dans un temps raisonnable sont peu nombreux et très spécifiques. Dans la majorité des cas, on utilise des heuristiques. On peut classer les méthodes d'agencement en trois groupes (Heragu S., Kusiak A., 1988) :

- **Méthodes par construction** :

Dans ces méthodes, on part d'un espace vide illimité sur lequel on place les ressources au fur et à mesure que le processus d'agencement se déroule, jusqu'à ce que toutes les ressources soient placées.

- **Méthodes par amélioration** :

Dans celles-ci, on commence par une configuration initiale de l'agencement donnée par l'utilisateur. Cette configuration est modifiée à chaque itération du processus afin d'améliorer la valeur du critère de placement choisi.

- **Méthodes mixtes** :

Ces méthodes mettent en collaboration les deux approches précédentes. Une configuration initiale est obtenue à l'aide d'une méthode par construction. Ensuite, on cherche à améliorer la solution obtenue afin d'aboutir à la configuration donnant une valeur du critère de placement aussi proche de l'optimum que possible, en tenant compte des contraintes éventuelles.

I.1.3 MODELES MATHÉMATIQUES ET ALGORITHMES DE RESOLUTION

Les différents modèles mathématiques utilisés pour la recherche d'agencements peuvent être classés en quatre grandes familles :

- L'***affectation quadratique*** (affectation des ressources à des sites prédéterminés),

- l'approche basée sur la ***théorie des graphes***,

- l'approche basée sur les langages de ***programmation par contraintes***,

- et les ***techniques basées sur l'Intelligence Artificielle***, et notamment les systèmes experts.

1. L'Affectation Quadratique

Un modèle utilisé fréquemment pour la résolution optimale du problème d'agencement est le modèle d'affectation quadratique (quadratic assignment problem, QAP), ainsi nommé parce que la fonction objectif est du second degré et que les contraintes sont linéaires. Koopmans et Beckman (1957) furent les premiers à proposer ce modèle d'affectation quadratique. Le QAP s'applique aux problèmes dans lesquels il s'agit d'imposer des emplacements fixes et d'y affecter les ressources de manière à optimiser un critère quadratique donné en respectant des contraintes linéaires. Sahni et Gonzalez (1976) ont démontré que le QAP est un problème NP-complet.

Ce modèle, sous sa forme la plus simple, s'exprime comme suit :

Soient : f_{ij} le flux entre la ressource i et la ressource j. Ce flux est indépendant de l'emplacement de la ressource,

c_{kl} le coût de transport unitaire de l'emplacement (site) k à l'emplacement l,

x_{ik} une variable binaire, égale à 1 si la ressource i se trouve à l'emplacement k, et à 0 sinon.

L'objectif consiste à minimiser le coût total qui s'écrit :

$$\text{Min} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \sum_{l=1}^{M} f_{ij}\ c_{kl}\ x_{ik}\ x_{jl}$$

sous les contraintes :

$$\sum_{j=1}^{M} x_{ij} = 1,\ i = 1,2,...,N \qquad (1)$$

$$\sum_{i=1}^{N} x_{ij} = 1,\ j = 1,2,...,M \qquad (2)$$

$$x_{ij} \in \{0,1\},\ i = 1,2,...,N;\ j = 1,2,...,M \qquad (3)$$

où N est le nombre de ressources et M le nombre d'emplacements.

Les contraintes (1) et (2) signifient respectivement qu'il ne peut jamais y avoir deux ressources à un même emplacement, et qu'une ressource ne peut jamais être placée à deux emplacements différents à la fois.

Méthodes de résolution

Pour résoudre le problème d'affectation quadratique, plusieurs types d'algorithmes ont été développés (Kusiak A.,Heragu S., 1987) :

- les algorithmes basés sur la méthode "Branch and Bound" (méthode par séparation et évaluation),

- les algorithmes basés sur le principe dit du "Cutting Plane".

Les procédures par séparation et évaluation ("Branch and Bound") explorent implicitement l'ensemble des solutions du problème. Elles effectuent une recherche arborescente partielle grâce à l'utilisation de bornes aux différents noeuds de l'arborescence. Leur utilisation devient délicate lorsque le nombre de ressources à placer augmente.

Les algorithmes dits du "Cutting Plane" ont été développés pour résoudre des problèmes d'optimisation globale multi-extrêmale tels que la minimisation globale d'une fonction concave dans un ensemble convexe et compact. La partie la plus coûteuse en temps de calcul de ces algorithmes concerne le calcul de tous les sommets du polytope P* créé à partir du polytope P (dont l'ensemble des

sommets est connu) par coupure par un plan. Les algorithmes du type "Cutting Plane" présentent les mêmes inconvénients pour les problèmes d'agencement que les algorithmes "Branch and Bound" et n'aboutissent pas à des solutions optimales (Eaves B.C., Zangwill W.I., 1971).

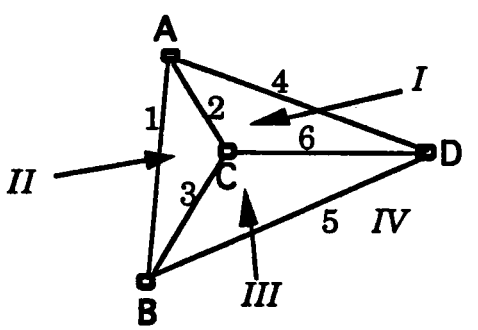
2. Approches basées sur la Théorie des Graphes

Les premières utilisations de la théorie des graphes pour résoudre des problèmes d'agencement ont vu le jour dans les années 1960. Plusieurs travaux ont été menés dans cette voie jusqu'à ce que l'approche atteigne son état actuel (Kusiak A., Heragu S., 1987). Avant de présenter cet état, donnons quelques définitions utiles.

Un *graphe* comporte un ensemble N de sommets. Ces sommets sont reliés par des arcs qui constituent l'ensemble A. Le graphe est noté G = (N, A).

On distingue plusieurs types de graphes. Dans celui qui nous intéresse, les arcs ne sont pas orientés et chacun des sommets est relié à tous les autres par au moins un chemin. Les graphes de ce type sont dits connexes non orientés.

Les surfaces limitées par des arcs et ne contenant aucun sommet à l'intérieur sont appelées *faces* du graphe (voir figure I.1.). La région à l'extérieur du graphe est aussi une face du graphe, mais elle est infinie. De plus, les arcs du graphe peuvent être pondérés.



Sommets : **A, B, C, D**

Arcs : 1, 2, 3, 4, 5, 6

Faces : *I, II, III, IV*

Rq. : IV est la face infinie.

Figure I.1 Graphe connexe non orienté

Deux aspects concernant ce type de graphes vont être exploités dans la suite de l'exposé : la propriété de *planéité du graphe* et le *concept de dualité.*

- Un graphe est *planaire* si ses arcs ne se coupent que sur des sommets (le graphe de la figure I.1.est planaire).

- Un graphe planaire est dit *maximal* s'il contient le nombre maximal d'arcs. Si n est le nombre de sommets d'un graphe planaire :

  - le nombre maximal d'arcs est $a_{max} = 3\,n - 6$,

  - le nombre maximal de faces est $f_{max} = 2\,n - 4$,

  (f_{max} inclut la face infinie).

- A tout graphe planaire G on peut associer un *graphe dual* G* qui est aussi planaire. G* est construit en plaçant un point dans chaque face de G (face infinie inclue) et en joignant, ensuite, tout couple de sommets par une ligne (voir figure I.2). Les points ainsi obtenus sont les sommets de G* et les lignes sont ses arcs. G et G* ont le même nombre d'arcs.

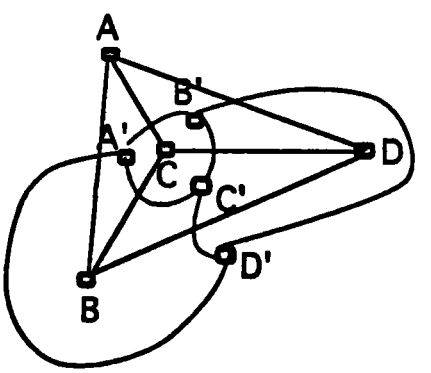


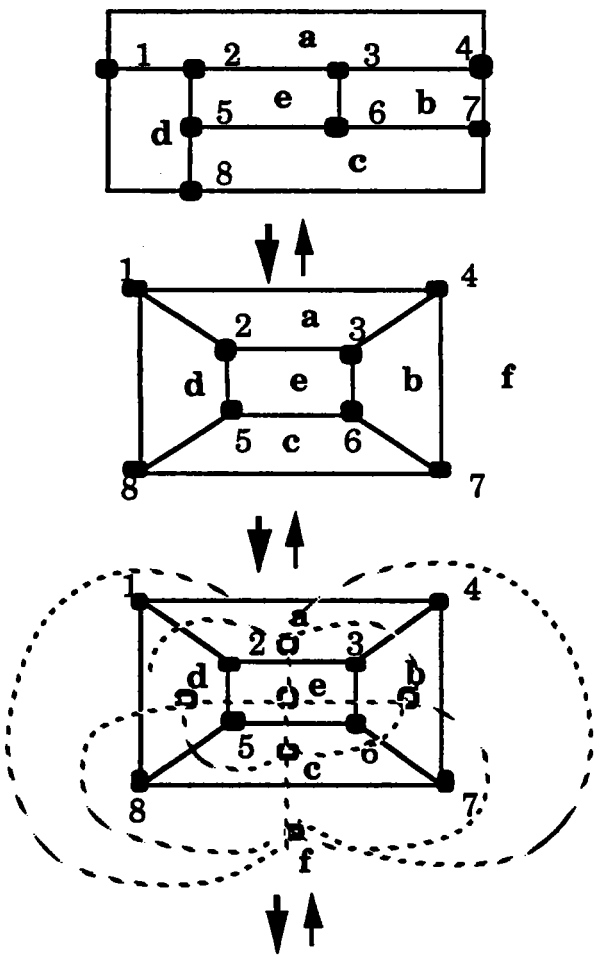
Figure I.2 Graphe dual d'un graphe planaire maximal

Méthodes de résolution

La résolution du problème d'agencement par la théorie des graphes est essentiellement une approche par construction. Compte tenu du modèle sur lequel elle est basée, elle n'est applicable que dans les cas où il s'agit d'agencer une surface donnée en la partagant en blocs ou services adjacents, tout en respectant certaines conditions de proximité entre ces derniers (c'est ce que

nous avons appelé "agencement par partitionnement de la surface disponible" au paragraphe I.1.1). La figure I.3 donne un exemple.

Soit une surface partagée en blocs rectangulaires adjacents. A cette surface est associé un graphe planaire dont les sommets représentent les points d'intersection des côtés des différents blocs et les arcs représentent les côtés appartenant simultanément à deux blocs adjacents et reliant deux sommets. Enfin, à ce graphe planaire (et maximal par construction) est associé son graphe dual qui est aussi planaire et maximal. Les arcs de ce dernier sont pondérés, et les valeurs de pondération représentent les proximités entre les blocs. Ces proximités sont résumées par une matrice (voir figures I.3.a et I.3.b). Le graphe ainsi obtenu est dit Graphe Planaire Pondéré Maximal "GPPM".



Surface divisée en blocs

Graphe associé à la surface (GPPM)

Graphe dual (GPPM*)

Figure I.3.a Processus d'agencement par la théorie des graphes

	a	b	c	d	e
a	-	I	O	U	E
b		-	A	U	I
c			-	E	O
d				-	A
e					-

Matrice des proximités

A : Proximité absolument nécessaire
E : Proximité très importante
I : Proximité importante
O : Proximité d'importance moyenne
U : Proximité sans importance
X : Proximité non souhaitée

Figure I.3.b Processus d'agencement par la théorie des graphes

Pour réaliser l'opération d'agencement, on fait la synthèse de ce qui a été décrit ci-dessus (en suivant les flèches en pointillé dans la figure I.3). En d'autres termes, il faut suivre l'algorithme à trois étapes suivant :

1. A partir de la matrice des proximités, construire le graphe planaire pondéré maximal GPPM,

2. Construire le graphe GPPM* dual de GPPM,

3. Convertir GPPM* en agencement.

Nous ne développons pas ces trois points. Pour plus de détails, voir (Has, 1987).

Cette approche présente plusieurs limitations. Certaines sont communes à toutes les approches qui utilisent la matrice de proximité, entre autres : la difficulté de détermination du premier bloc à placer (problème combinatoire), ou le "phénomène du parapluie" où plusieurs blocs ont tendance à se regrouper autour d'un même bloc. D'autres limitations sont propres à cette approche, telles que : la difficulté de construire le GPPM à partir de la matrice de proximité en s'assurant de sa planéité, ainsi que la difficulté de passer du graphe dual GPPM* à l'agencement.

3. Approches basées sur les langages de Programmation par Contraintes

Les langages de programmation par contraintes permettent d'exprimer aisément un ensemble de contraintes portant sur des variables de plusieurs objets, dont le respect est assuré automatiquement par un mécanisme de propagation de contraintes (Lel, 1988). La résolution du problème d'agencement par les approches basées sur les langages de programmation

par contraintes est essentiellement une approche par construction. Elle n'est applicable que dans les cas où il s'agit d'agencer des ressources sur une surface donnée et tout en respectant certaines contraintes. Programmer devient une tâche déclarative dans la programmation par contraintes. Le programmeur met en place un ensemble de relations (ou contraintes) entre un groupe d'objets, et c'est le système de propagation de contraintes qui se charge de trouver un ensemble de valeurs qui satisfasse ces relations.

4. Techniques basées sur l'Intelligence Artificielle, et notamment les Systèmes Experts

L'Intelligence Artificielle (IA) a sa place dans la résolution de certains problèmes industriels, et en particulier dans la résolution des problèmes d'agencement. Elle intervient chaque fois que les techniques mathématiques se révèlent inadaptées. Cette inadéquation se manifeste souvent lorsqu'il s'agit de traiter des données qualitatives ou symboliques.

Les systèmes à base de connaissances utilisent la connaissance de l'expert humain. Les systèmes experts cherchent à imiter le raisonnement d'un expert humain pour un domaine d'application bien précis. Ce sont des systèmes de raisonnement simples qui s'appuient sur une base de données, une base de connaissances, et un moteur d'inférence (voir figure I.4).

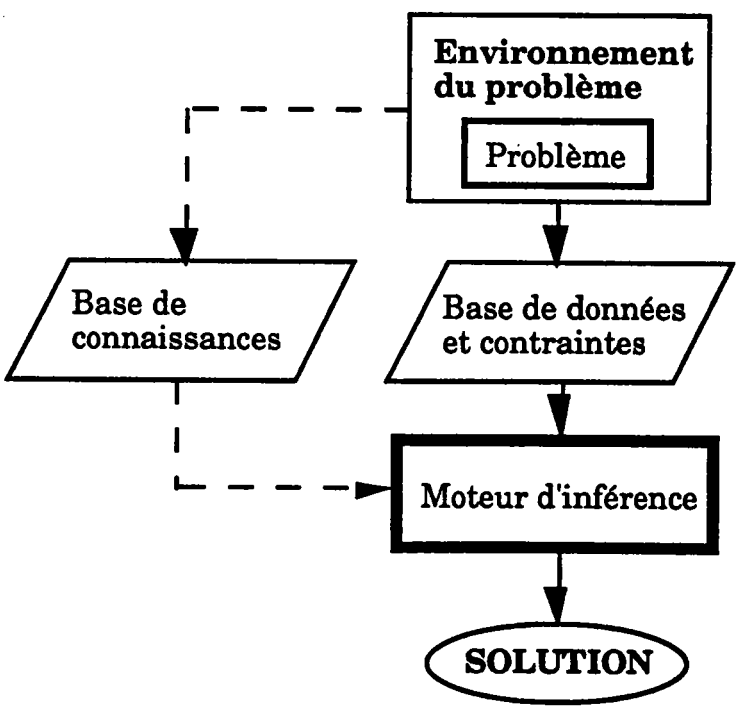


Figure I.4 Schéma simplifié d'un système expert

Ces techniques peuvent être utilisées pour l'agencement de ressources à l'intérieur des cellules de fabrication, comme nous le verrons au Chapitre III. En effet, ce problème fait intervenir un grand nombre de données (numériques et symboliques) utilisées dans des règles heuristiques reflétant une expertise humaine en conception de cellules de fabrication.

I.2 PROBLEMES DE L'AGENCEMENT DES SYSTEMES DE PRODUCTION

I.2.1 INTRODUCTION

La compétition internationale dans le domaine industriel exige l'amélioration de tous les aspects de l'organe de production. Les chercheurs et les ingénieurs sont sensibles aux problèmes d'agencement depuis les années cinquante, mais on observe un net regain d'intérêt pour le domaine depuis le début des années 80. Les raisons en sont multiples : accélération du renouvellement complet ou partiel des systèmes de production, variation de plus en plus importante de la demande, concurrence de plus en plus acharnée, ce qui exige des efforts particuliers, etc ... Un des aspects les plus récents de cette évolution vers l'usine du futur est la recherche d'un agencement optimal des systèmes de fabrication. On a en effet constaté qu'un agencement rationnel conduit à une diminution des cycles de production, une meilleure fiabilité du système, une qualité accrue des produits, une simplification de la gestion, une réponse plus rapide aux fluctuations de la demande, une diminution des encours et du personnel donc, in fine, une réduction des coûts de production. Un bon agencement est donc synonyme d'économie et de compétitivité accrue. C'est surtout, de notre point de vue, une des composantes les plus importantes d'une réponse intelligente aux tendances du monde moderne de la production.

I.2.2 PRESENTATION DU PROBLEME

Le coup d'envoi des travaux sur les problèmes de l'agencement des systèmes de production a été donné par des chercheurs tels que Grant I.W. (1952), Richard M.D. (1955), et El-Rayad T.E., Hollier H.P. (1970). Ces derniers ont proposé une définition de l'agencement des systèmes de production qui est la suivante :

" L'agencement est une procédure visant à obtenir une disposition optimale des postes de travail dans une unité de production ".

Dans la pratique, il existe deux situations dans lesquelles se pose le problème d'agencement :

- Construction d'une nouvelle unité de production,

- Modification d'une unité de production existante.

La construction d'une nouvelle unité de production ne se présente que rarement et son coût de réalisation est relativement élevé. Elle repose sur les données concernant les produits, les ressources, et les moyens de manutention disponibles.

La modification d'une unité de production existante est le cas le plus fréquent. La difficulté du problème dépend des causes qui sont à l'origine de la modification et des objectifs fixés. Un ré-agencement est rendu nécessaire dans les cas suivants :

- lancement de produits nouveaux,

- modification des gammes de fabrication de certains produits,

- ajout de machines dans le système de production,

- retrait de machines du système de production,

- remplacement de certaines machines,

- accroissement de la taille de l'atelier de production,

- changement de certains moyens de transport dans l'atelier,

- instauration de nouvelles normes de sécurité,

- existence de zones de conflits de transport non contrôlées, provoquant une augmentation des en-cours.

Cette liste n'est pas exhaustive.

I.2.3 LES OBJECTIFS

Les critères à optimiser sont les suivants :

- minimiser le trafic entre les postes de travail dont les coûts induits constituent la partie la plus importante du coût d'utilisation du système,

- réduire le temps global de production (i.e., le cycle moyen de production),

- minimiser le coût d'installation du système,

- minimiser les "en-cours" dus aux conflits de routage,

- utiliser l'espace disponible d'une manière optimale.

Certains de ces critères sont principaux, d'autres secondaires. De même, certains sont quantitatifs et d'autres qualitatifs. En outre, il ne sont pas indépendants. Néanmoins, on essaie toujours de les optimiser simultanément, en favorisant les plus importants du point de vue des utilisateurs.

I.2.4 ETAT DE L'ART / METHODES ET LOGICIELS EXISTANTS

Le problème de l'agencement des systèmes de production ne peut pas être résolu globalement à l'aide d'une méthode exacte et en un temps raisonnable. Dans la majorité des cas, on doit utiliser des heuristiques qui donnent des résultats proches de l'optimum en un temps acceptable.

Dans ce qui suit, nous allons présenter quelques logiciels connus.

1. CORELAP (COMPUTERIZED RELATIONSHIP LAYOUT PLANNING)

CORELAP utilise les relations de proximité figurant les liens entre les postes de travail. Ces relations sont, entre autres, basées sur les flux de matières. Il en déduit un taux total de proximité (total closeness rating) pour chaque poste de travail. Le taux indique l'importance des liens qu'a ce poste de travail avec l'ensemble des autres. Par conséquent, CORELAP place en priorité le poste de travail ayant le plus grand taux de proximité sur la surface prévue. Il ajoute ensuite les autres postes de travail dans l'ordre décroissant de leur taux de proximité jusqu'à ce qu'ils soient tous placés.

Une approche de ce type est dite "par construction" (comme expliqué dans le paragraphe I.1.2), car elle fournit une solution en plaçant successivement les postes de travail sur une surface supposée infinie.

CORELAP ne permet pas de prendre en compte des formes imposées (par exemple la forme de la surface d'atelier disponible), ce qui le rend en

particulier impropre aux tâches de réorganisation d'atelier. Un aspect positif du logiciel est sa rapidité d'exécution (Lee R.C., Moore J.M., 1967).

2. CRAFT (COMPUTERIZED RELATIVE ALLOCATION OF FACILITIES TECHNIQUE)

CRAFT utilise les données relatives aux flux de produits entre les postes de travail. Il part d'une configuration initiale admissible (par exemple, la configuration antérieure de l'atelier). Il essaie ensuite des permutations des éléments du système jusqu'à obtenir une configuration qui améliore le critère, et il recommence.

On dit que CRAFT est une approche par "améliorations successives".

CRAFT ne présente pas beaucoup d'intérêt pour notre étude, essentiellement pour deux raisons :

(1) Il ne prend pas en compte les formes des machines (ou des cellules) : elles sont toutes, a priori, considérées comme modifiables ;

(2) Aucun mécanisme n'existe qui permette d'utiliser un emplacement non occupé dans la configuration initiale (Buffa et al., 1964).

3. DISCON (DISPERSION-CONCENTRATION METHOD)

DISCON est une approche entièrement mathématique. Le problème de layout est modélisé comme un problème de programmation mathématique non convexe. Les éléments à placer (machines, moyens de transport ou cellules) sont supposés être des cercles. On cherche à placer ces cercles sur la surface disponible de façon à minimiser le critère. Ce critère est la somme des produits "coût de transport x distance entre les centres des cercles". La contrainte à respecter est le non-chevauchement des cercles. Dans cette méthode, on ne tient pas compte des contraintes physiques telles que la proximité ou l'éloignement entre des éléments à placer, ou encore la taille de la surface disponible.

Le problème est résolu en utilisant un algorithme (appelé algorithme de dispersion-concentration) qui comporte deux étapes. Dans la première étape (étape de dispersion), en utilisant la méthode lagrangienne de type gradient dérivé, de bonnes conditions initiales sont trouvées pour obtenir un minimum local admissible. La solution finale de l'étape de dispersion est une bonne

configuration pour débuter la deuxième étape (étape de concentration). Celle-ci cherche à concentrer des éléments à placer de façon à ce qu'ils ne se chevauchent pas. Cette solution est un minimum local du problème de programmation mathématique.

L'approche précédente a le défaut de ne pas faire référence aux contraintes physiques. En effet, cette approche n'est pas une approche réaliste adaptée au problème de l'agencement des systèmes de production (Drezner Z., 1980).

4. PLOOT (PLANT LAYOUT OPTIMIZATION TOOL)

PLOOT est un produit basé sur le recuit simulé. Il prend en compte la surface disponible de l'atelier et distingue quatre formes différentes d'éléments à placer. Le critère à minimiser est constitué de la somme des produits "flux x distance".

L'algorithme se déroule en deux étapes. Dans la première étape, on détermine l'implantation initiale. Elle est basée sur la constitution de cellules de fabrication minimisant le trafic inter-cellulaire. La deuxième étape réside en l'application du recuit simulé. Ici, il est nécessaire de trouver à chaque étape une configuration voisine de la configuration précédente. On l'obtient soit en permutant deux éléments, soit en déplaçant un élément vers un emplacement libre. Pendant cette permutation, on vérifie toujours si toutes les contraintes physiques sont satisfaites. Si l'une d'entre elles est violée, alors la permutation est annulée et l'on fait un autre essai.

PLOOT utilise la distance de Manhattan, qui ne donne généralement pas la longueur du chemin le plus court entre les éléments à placer en tenant compte des contraintes physiques, mais il permet de tenir compte des formes des éléments avec un minimum de réalisme (Beziat Ph., 1990).

5. SHAPE (SELECTION OF MATERIALS HANDLING EQUIPMENT AND AREA PLACEMENT EVALUATION)

SHAPE propose une méthode par construction. Le critère à minimiser est la somme des couples des éléments des produits "flux x distance".

L'algorithme se déroule en deux étapes. La première consiste à classer les éléments par ordre d'importance. La seconde est l'étape de placement. L'élément en tête dans l'ordre précédent est placé d'abord. L'élément suivant

est placé sur un des quatre côtés de l'ensemble déjà placé et conserve le meilleur placement, et ainsi de suite. Cette approche nécessite parfois un ajustement manuel car elle ne garantit pas des ensembles compacts.

Le logiciel SHAPE ne permet pas de placer des formes définies a priori et il ne tient pas compte de la forme de la surface disponible, ce qui le rend difficile à utiliser dans l'agencement des systèmes de production (Hassan M.M.D. et al.,1986).

6. MACHINE LAYOUT PROBLEMS IN FLEXIBLE MANUFACTURING SYSTEMS

Deux méthodes "par construction" pour l'obtention du layout de systèmes de fabrication flexibles sont proposées par Heragu S. et Kusiak A. (1988). La première méthode, basée sur le flux entre les machines, donne simplement la séquence des machines à placer. L'agencement réel dépend du moyen de transport, de l'espace nécessaire entre les machines, et de leur orientation. Cette méthode est seulement applicable pour les configurations en ligne.

Dans la deuxième méthode, un algorithme d'agencement triangulaire est développé. Il est applicable aux configuration en ligne double et multi-lignes. On construit d'abord, un arbre de recouvrement maximal basé sur les valeurs maximales du flux entre les machines. Dans une deuxième phase, les machines sont classées par ordre décroissant et affectées à des sites pré-spécifiés garantissant l'espace nécessaire entre elles et leur orientation.

D'après les auteurs, la première méthode ne garantit pas de solution optimale pour les problèmes de plus de trois machines. Pour quelques problèmes classiques, la deuxième méthode donne des solutions moins bonnes que celles obtenues par les algorithmes basés sur la méthode d'amélioration (Heragu S., 1988).

7. KBML (A KNOWLEDGE-BASED SYSTEM FOR MACHINE LAYOUT)

KBML est un système hybride basé sur des systèmes experts et des algorithmes d'optimisation combinatoire. Il considère les facteurs qualitatifs aussi bien que quantitatifs, c'est-à-dire qu'il prend en compte, par exemple, différentes configurations de layout fondamental et différents moyens de transport possibles. Deux formulations pour les configurations en simple-ligne et en multi-lignes sont présentées. Leurs systèmes experts peuvent utiliser

n'importe quel algorithme de layout comme, par exemple, celui proposé dans cette thèse (Heragu S., 1990).

8. CLASS (COMPUTERIZED LAYOUT SOLUTIONS USING SIMULATED ANNEALING)

CLASS est un algorithme "par amélioration". Son objectif est de minimiser le flux total de produits entre les ressources. CLASS utilise la méthode du recuit simulé. L'avantage majeur de cette méthode est l'insensibilité de la solution finale à la configuration initiale (Harhalakis G. et al., 1991).

Au delà des logiciels que nous venons de considérer, il existe d'autres algorithmes basés sur la méthode constructive (à l'instar de CORELAP et SHAPE). Ces algorithmes sont les suivants :

1. **HC 66**, Quadratic assignment problem algorithms and the location of indivisible facilities (Hillier F.C., Conners M.M., 1966) ;

2. **ALDEP**, Automated layout design program (Seehof J.M., Evans W.O., 1967) ;

3. **MAT**, Modular allocation technique (Edwards et al., 1970) ;

4. **RMA Comp I**, Computerized layout planning (Muther R., McPherson K., 1970) ;

5. **PLANET**, Plant layout analysis and evaluation technique (Deisenroth M.P., Apple J.M., 1972) ;

6. **LSP**, Layout planning by comuter simulation (Zoller K., Adendorff K., 1972) ;

7. **FATE**, A new construction algorithm for facility layout (Block T.E., 1978) ;

8. **INLAYT**, An interactive approach to computer aided facility layout (O'Brien C., Abdel Barr S.E.Z., 1980) ;

9. **FLAT**, A construction algorithm for the facility layout problem (Heragu S., Kusiak A., 1986) ;

10. **INTALA**, Layoutplanung in der Fabrikplanung (Koch U., 1989) ;

11.**LAPLAS**, Rechnergestützte Layoutplanung von Industriebetrieben (Brandt H.-P., 1989) ;

12. **LAYOS**, Rechnerunterstützte Layoutoptimierung und -simulation (Jetschney W., 1991).

De même, en plus des algorithmes basés sur la méthode "par amélioration" (comme CRAFT et CLASS), on peut trouver les algorithmes suivants :

1. **H63**, Quantitative tools for plant layout analysis (Hillier F.C., 1963) ;

2. **COL**, A computerized model for office layout (Volmann T.E. et al., 1968) ;

3. **Sampling algorithm I**, Techniques for the assignment of facilities to locations (Nugent C.E. et al., 1968) ;

4. **FRAT**, Facilities relative allocation technique (Khalil T.M., 1973) ;

5. **Sampling algorithm II**, The terminal sampling procedure (Hitchings G.G., Cottam M., 1976) ;

6. **COFAD**, An applied model for the facilities design problem (Tompkins J.A., Reed R. Jr., 1976) ;

7. **Revised Hillier algorithm**, Perturbation scheme to improve Hillier's solution to the facilities layout problem (Picone C.J., Wilhelm W.E., 1984).

En plus des méthodes mixtes déjà présentées (par construction et amélioration à l'exemple de DISCON), il existe les algorithmes suivants :

1. Hospital layout as a quadratic assignment problem (Elshafei A.N., 1977) ;

2. A branch-and-bound-based heuristic for solving the QAP (Bazaraa M.S., Kirca O., 1983) ;

3. **FLAC**, A cluster-analytic approach to facility layout (Scriabin M., Vergin R.C., 1985).

I.2.5 CONCLUSION

Dans ce chapitre, nous avons présenté les méthodes qui nous semblent les plus significatives des différents types d'approches existantes pour notre problématique.

Aucun de ces produits ne nous satisfait pleinement, car il leur manque au moins l'une des caractéristiques suivantes :

(i) Les distances entre les ressources à placer ne sont pas les longueurs réelles des chemins.

(ii) Les formes et tailles des ressources ne sont pas prises en compte.

(iii) Les limites physiques de la surface disponible ne sont pas prises en compte.

(iv) Les différents moyens de transport existants avec leurs caractéristiques réelles ne sont pas pris en compte.

(v) Non seulement le flux entre les ressources, mais aussi les contraintes de proximité entre les ressources ainsi que les ressources qui doivent être fixes sur certains emplacements, ne sont pas prises en compte.

Nous proposons plus loin une méthode qui non seulement satisfait les conditions précédentes, mais encore fournit un tracé optimal des chemins empruntés par les ressources de transport.

CHAPITRE II

NOUVELLE APPROCHE DE RESOLUTION DU PROBLEME DE L'AGENCEMENT DES SYSTEMES DE FABRICATION

II.1 INTRODUCTION

Après avoir passé en revue l'état de l'art, nous avons constaté que le problème de l'agencement des systèmes de fabrication n'a pas encore été résolu d'une manière satisfaisante. Une méthode adéquate devrait tenir compte de toutes les caractéristiques de la fabrication, c'est-à-dire celles concernant:

- les produits à fabriquer (type, nombre de produits, dimensions, poids, matériaux, ...) ;

- les machines utilisées (nombre de machines, capacité, taille, fonction, ...) ;

- les gammes de fabrication (séquences d'opérations, horizon de travail, ...) ;

- les systèmes de transport possibles (vitesse, taille, poids transportables, ...) ;

- la surface disponible de l'atelier (géométrie, zones interdites, entrée/sortie, ...) ;

- les ressources utilisées (énergie, moyens auxiliaires, ...).

Nous présentons dans ce qui suit l'approche adoptée au sein du Projet SAGEP de l'INRIA-Lorraine dans le cadre du projet COALA, projet ESPRIT No. 5474. C'est une approche qui vise principalement la conception des ateliers de fabrication en cellules et donne un agencement satisfaisant. Nous retenons trois phases principales dans COALA pour le problème global d'agencement des systèmes de fabrication (voir figure II.1) :

- **La conception des cellules de fabrication,**

- **L'agencement des ressources à l'intérieur des cellules,**

- **L'agencement des cellules dans l'atelier de fabrication.**

La première phase consiste à regrouper les machines en cellules, en optimisant un ou plusieurs critères, et en respectant les contraintes qui s'appliquent aux produits, aux machines et aux gammes de fabrication.

Dans la deuxième phase, on réalise l'agencement intra-cellulaire, c'est-à-dire le choix du moyen de transport, le choix du type d'agencement fondamental (parmi cinq types différents), et la disposition des machines suivant le type d'agencement choisi (construction physique des cellules). Cette phase constitue le thème principal de cette thèse.

La dernière phase est la détermination de l'agencement des cellules sur la surface disponible de l'atelier, en respectant les contraintes physiques et de fonctionnement.

Dans la suite, nous présentons les trois phases en insistant particulièrement sur la deuxième, à laquelle nous consacrons le chapitre III de cette thèse.

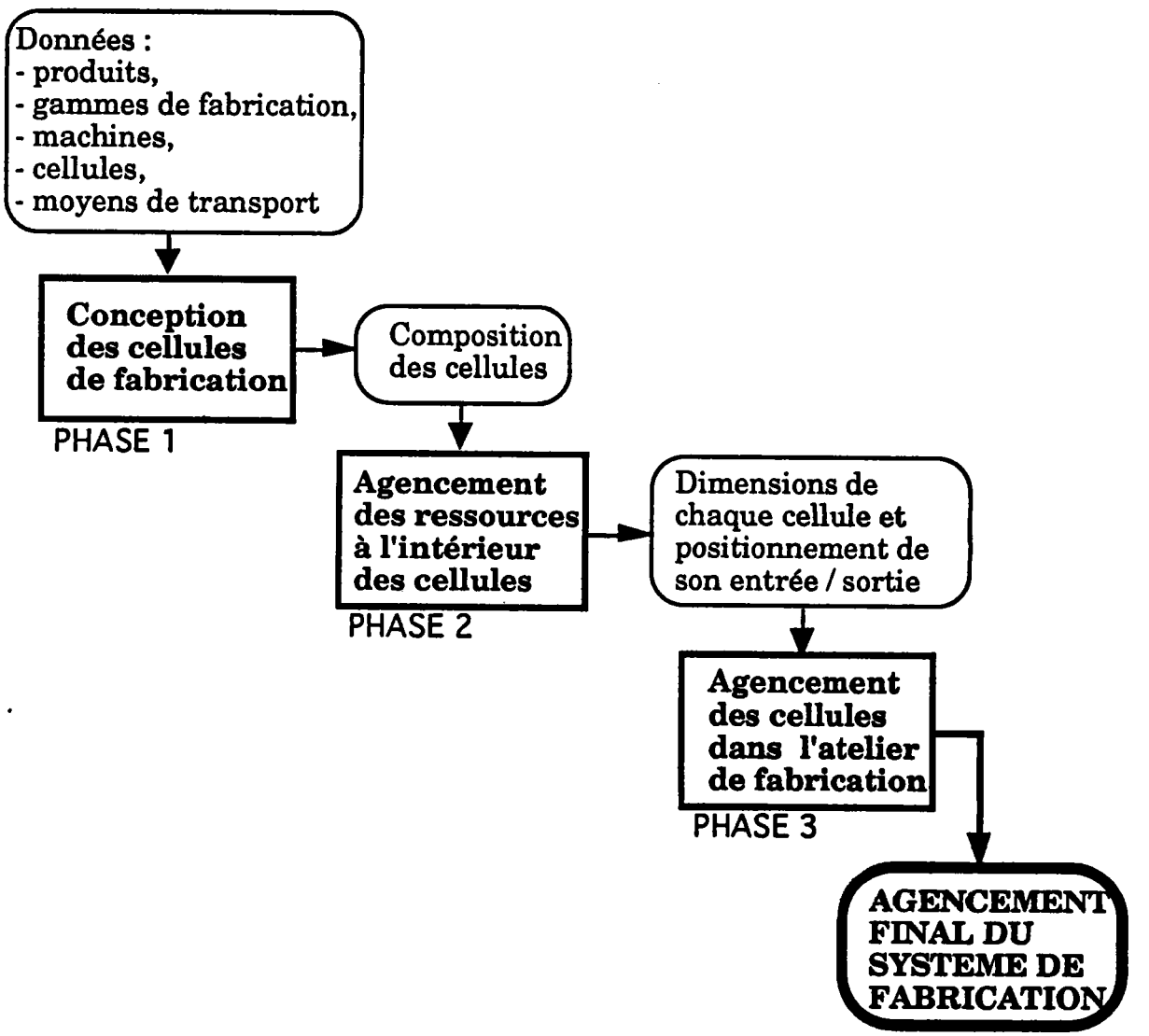


Figure II.1 Les trois phases principales de COALA

II.2 CONCEPTION DES CELLULES DE FABRICATION

Cette première phase consiste à regrouper les machines (ou ressources) du système de fabrication en cellules aussi indépendantes les unes des autres que possible, de façon à minimiser le trafic entre les cellules. On essaie donc de fabriquer chaque produit à l'intérieur d'une seule cellule.

Cette partition de l'ensemble des ressources s'effectue de façon à optimiser un ou plusieurs critères en tenant compte de diverses contraintes (par exemple, on limite le nombre maximal de machines dans chaque cellule, ou bien on oblige les machines effectuant des travaux salissants à figurer dans la même cellule). Ce problème s'exprime sous forme d'un problème d'optimisation combinatoire sous contraintes, dont la résolution nécessite de préférence l'emploi d'une méthode heuristique.

Pour résoudre ce problème, une formulation mathématique peut être proposée à partir des définitions suivantes :

Soient $M = \{M_1, M_2, ..., M_m\}$ l'ensemble des m machines du système,

$P = \{P_1, P_2, ..., P_N\}$ l'ensemble des N types de produits à fabriquer dans le système de fabrication.

A chaque type de produit P_i (i = 1, 2, ..., N) est associé un routage R_i,

$R_i = \{M_{i,1}, M_{i,2}, ..., M_{i,ki}\}$,

$M_{i,j}$, j = 1, ..., k_i sont les machines utilisées dans le routage R_i,

i est l'indice du produit, (i = 1, 2, ..., N) et

j est l'indice de la machine dans la séquence du routage R_i

(j= 1, 2, ..., k_i), où k_i est le nombre de machines du routage R_i de

Le problème consiste à minimiser le critère suivant :

Pour tout couple de cellules Ci et Cj, ($i \neq j$)

$$\text{Min} \sum_{M_u \in C_i} \sum_{M_v \in C_j} tr(M_u, M_v) \qquad (1)$$

où $tr(M_u, M_v)$ est le trafic entre la machine M_u de la cellule Ci et la machine M_v de la cellule Cj ($u \neq v$).

Le trafic est donné par :

$$tr(M_u, M_v) = \sum_{k=1}^{N} \mu_k n_k$$

μ_k est la quantité de produits du type k transportés à la fois d'une machine à une autre,

n_k est le nombre de passages du moyen de transport entre la machine M_u et la machine M_v pour le produit k sur la période de référence.

Si M_u et M_v appartiennent à la même cellule, alors $tr(M_u, M_v) = 0$.

La fonction objectif (1) décrite ci-dessus est soumise à deux grands types de contraintes, les *contraintes fortes* et les *contraintes molles*.

Les contraintes fortes :

Elles incluent :

i) La limitation de la taille des cellules. Si cette taille n'est pas limitée, les machines se regrouperont dans une même cellule pour donner le trafic inter-cellules minimal, c'est-à-dire un trafic nul. Cette taille limite peut être la surface de la cellule ou le nombre de machines qui la constituent.

ii) La cohabitation entre les machines dans une même cellule. Ces contraintes se réduisent aux deux classes suivantes :

1. Les machines M_u et M_v doivent être dans la même cellule,

2. Les machines M_u et M_v doivent être dans des cellules différentes.

Exemples :

- Les deux machines utilisent le même magasin d'outil,

- Les machines utilisent la même source d'énergie non transportable,

- Un même ouvrier surveille les deux machines,

- ... ,

- Un poste de peinture est toujours mis dans une cellule à part,

- Une machine source de vibrations ne peut pas être mise dans une cellule où se trouvent des machines de grande précision ou de mesure,

- ... .

Les contraintes molles :

Ce sont des contraintes non impératives dont la réalisation n'est que souhaitable.

Elles se réduisent aussi aux deux classes suivantes :

1. Il est souhaitable que les machines M_u et M_v soient dans la même cellule,

2. Il est souhaitable que les machines M_u et M_v ne soient pas dans la même cellule.

Exemples :

- Les deux machines utilisent la même source d'énergie,

- Simplification d'utilisation pour l'opérateur,

- ...,

- La présence des deux machines dans la même cellule exigerait une préparation particulière,

- ... .

Plusieurs méthodes de partitionnement heuristique ont été proposées pour résoudre ce problème combinatoire. Nous allons présenter les principes de trois méthodes qui nous paraissent représentatives des approches présentées dans la littérature.

II.2.1 LA METHODE GPM

Une première méthode a été proposée par Gracia H. et Proth J.M. (1985). Ils considèrent un ensemble de types de produits et un ensemble de machines. Chaque type de produits a une gamme de fabrication, c'est-à-dire une séquence de machines qu'une unité de ce type doit visiter pour être fabriquée, ainsi que les temps de passage sur chacune des machines. L'objectif est de chercher une partition de l'ensemble de types de produits en familles de produits et de l'ensemble de machines en cellules de fabrication de telle sorte que :

a) Une famille de produits corresponde à une cellule de fabrication.

b) Une cellule de fabrication corresponde à une famille de produits.

c) La partition minimise un critère qui est la somme pondérée :

1. du nombre de fois qu'une machine est utilisée pour transformer un produit appartenant à une famille de produits qui ne correspond pas à la cellule à laquelle appartient la machine,

2. du nombre de fois qu'une machine n'est pas utilisée pour transformer un produit appartenant à la famille qui correspond à sa cellule.

Considérons une matrice initiale binaire A à n lignes et m colonnes d'éléments a_{ij} (i = 1, 2, ..., n; j = 1, 2, ..., m) construite comme suit :

$$a_{ij} = \begin{cases} 1 \text{ si Mj figure dans la gamme } P_i \\ 0, \text{ sinon} \end{cases}$$

La méthode GPM cherche à modifier l'ordre des lignes et des colonnes de A de façon à faire apparaître des blocs non empiétants contenant un maximum de 1 et tels qu'il y ait un maximum de 0 à l'extérieur de ces blocs.

La figure II.2 schématise cette situation dans le cas de quatre blocs.

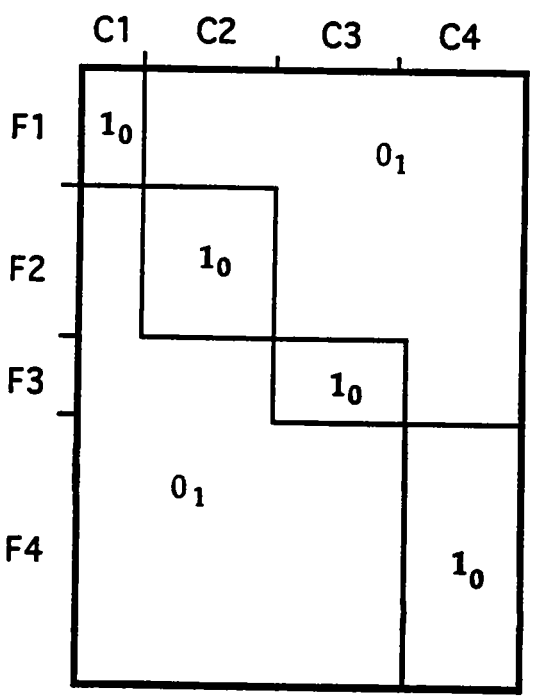


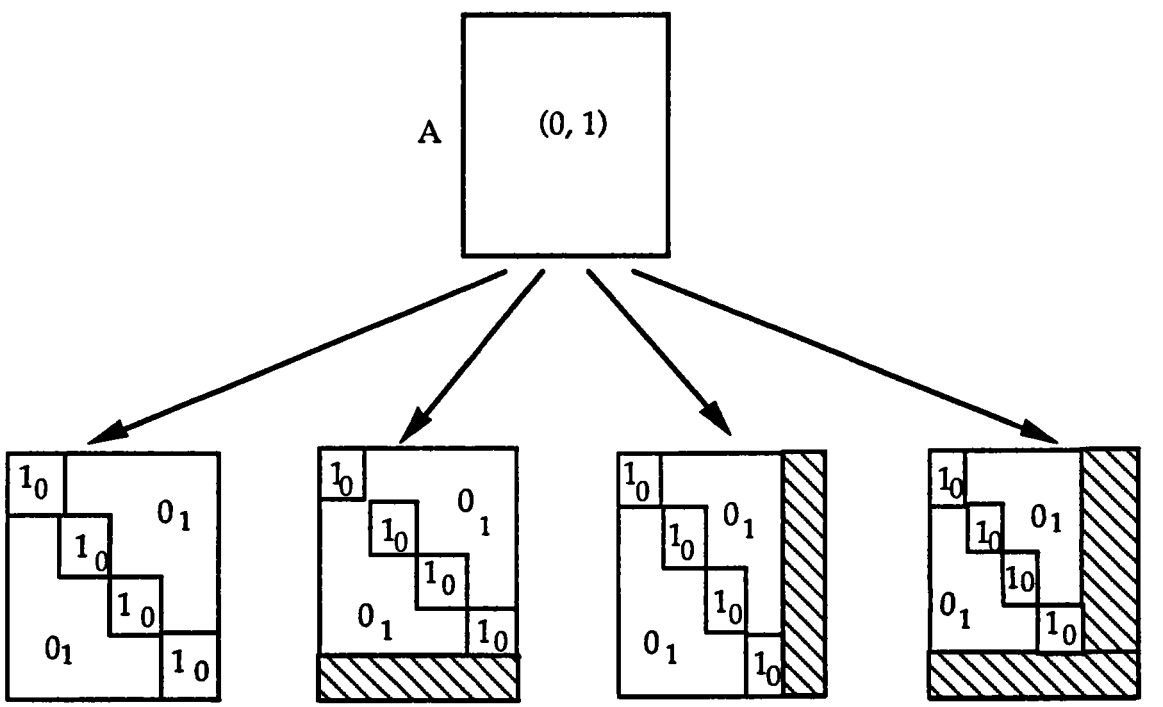
Figure II.2 : Résultat attendu (cas de quatre blocs)

Le symbole 1_0 indique un nombre de 1 aussi important que possible (et donc un nombre de 0 aussi faible que possible). Le symbole 0_1 indique un nombre de 0 aussi important que possible (et donc un nombre de 1 aussi faible que possible).

On cherche un ensemble de cellules $C_1, C_2, \ldots, C_K$ et un ensemble de familles de produits $F_1, F_2, \ldots, F_K$ tels qu'un produit d'un type appartenant à F_k soit fabriqué essentiellement à l'aide des machines de la cellule C_k ($k = 1, 2, \ldots, K$).

La figure II.3 schématise l'ensemble des résultats qu'on peut obtenir à partir d'une matrice initiale binaire A.

Un algorithme heuristique de la méthode GPM est présenté dans (Garcia H., Proth J.M., 1985).



Tous les types de produits et toutes les machines sont classés	Certains types de produits ne sont pas classés	Certaines machines ne sont pas classées	Certains types de produits et certaines machines ne sont pas classés

Figure II.3 : Résultats possibles

II.2.2 MINIMISATION DU TRAFIC INTER-CELLULES

Une deuxième méthode a été proposée par Harhalakis G. et al. (1990a). Cette méthode a pour objectif la minimisation du trafic inter-cellules, qui semble avoir la préférence des entreprises. En entrée, on fournit le nombre de produits qui passent d'une machine à l'autre sur un horizon du travail représentatif de la fabrication moyenne, ainsi que le nombre maximal de machines dans chaque cellule. On normalise le trafic entre deux cellules en divisant le trafic observé par le nombre total de machines dans les cellules. L'algorithme comporte deux phases. La première consiste à créer des cellules minimisant le trafic inter-cellules en respectant la contrainte de taille maximale d'une cellule. La deuxième phase a pour but le raffinement de la configuration obtenue dans la première phase.

II.2.3 METHODE BASEE SUR LE RECUIT SIMULE

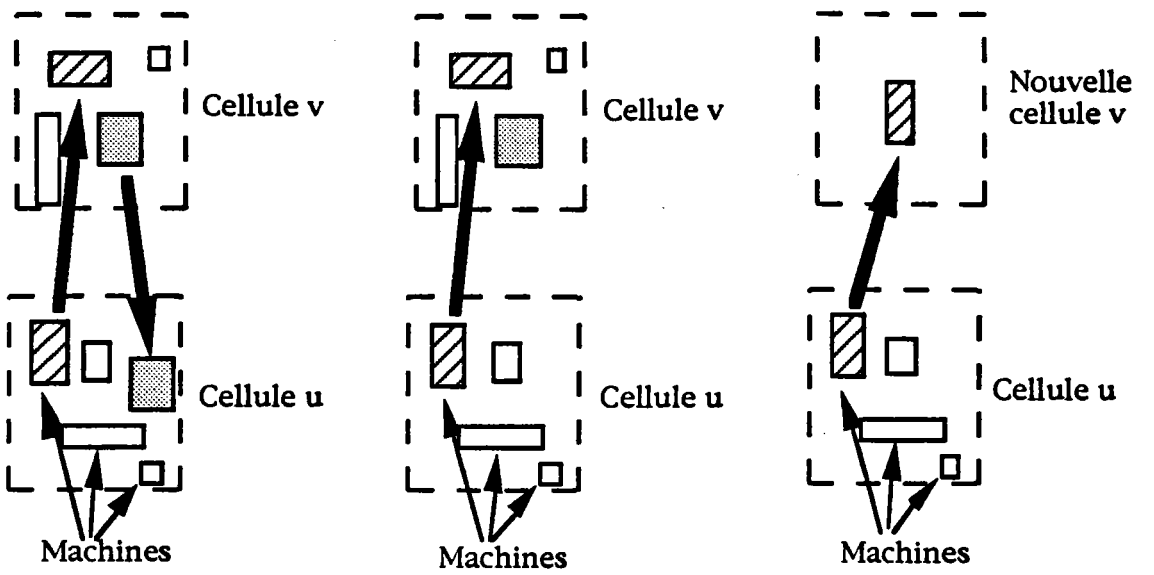
Une troisième méthode, qui est une méthode basée sur le recuit simulé, est celle proposée par Harhalakis G. et al. (1990b). L'objectif est la minimisation du trafic inter-cellules. Le "Recuit Simulé" sera présenté dans le chapitre III. Cette méthode donne aussi de bons résultats pour les exemples de grande taille pour un temps de calcul tout à fait acceptable.

Toute configuration possible de l'ensemble M des machines représente un état du système. Partant d'une configuration initiale, le passage d'un état à un autre s'effectue de la manière suivante : (voir aussi figure II.4)

- permutation de deux machines appartenant à des cellules différentes ;

- mutation d'une machine de sa cellule d'origine vers une autre cellule, à condition de ne pas violer la contrainte de taille ;

- mutation d'une machine de sa cellule d'origine vers une nouvelle cellule créée à cette occasion.



a) permutation de deux machines

b) mutation d'une machine vers une cellule existante

c) mutation d'une machine vers une nouvelle cellule

Figure II.4 Transformations élémentaires lors de la formation des cellules par la méthode du recuit simulé

Il existe aussi des méthode basées sur l'intelligence artificielle, mais nous ne nous intéressons pas à ces méthodes dans cette thèse.

II.3 AGENCEMENT DES RESSOURCES À L'INTERIEUR DES CELLULES

La seconde phase consiste à disposer les ressources à l'intérieur de chaque cellule de fabrication. Elle est divisée en trois étapes successives. Dans la première, on cherche à déterminer les moyens de transport possibles, puis les types d'agencement fondamentaux parmi cinq types de base considérés, et enfin la disposition physique des ressources sur la surface disponible de la cellule.

Nous reviendrons en détail sur cette phase qui est l'objet du chapitre III en présentant une **nouvelle approche de résolution du problème d'agencement des ressources à l'intérieur des cellules.**

II.4 AGENCEMENT DES CELLULES DANS L'ATELIER DE FABRICATION

La dernière phase de la résolution du problème global de l'agencement des systèmes de fabrication, qui a été développée par Souilah A. (1991) du Projet SAGEP de l'INRIA Lorraine, consiste à regrouper et à placer les cellules de fabrication déjà constituées sur la surface disponible, laquelle peut être quelconque, de façon à minimiser un critère lié au trafic et aux distances inter-cellules. Autour de chaque cellule, il a été ajouté une marge qui permet l'accès au moyen de transport. Pour simplifier le problème, chaque cellule est symbolisée par un rectangle ayant un point d'entrée et un point de sortie (voir figure II.5).

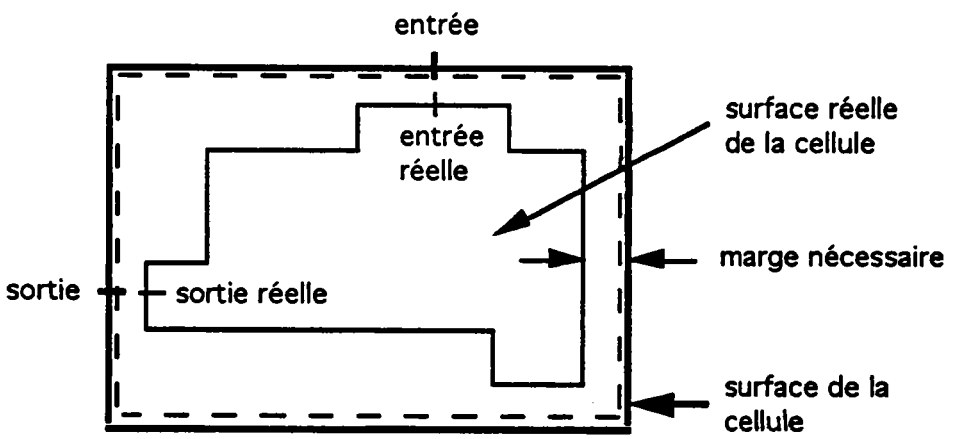


Figure II.5 : Exemple de cellule

La surface peut comporter des zones interdites aux cellules, par exemple des murs, des stockeurs, des zones occupées par des moyens auxiliaires ou induite par l'infrastructure générale de l'atelier, etc. (voir figure II.6).

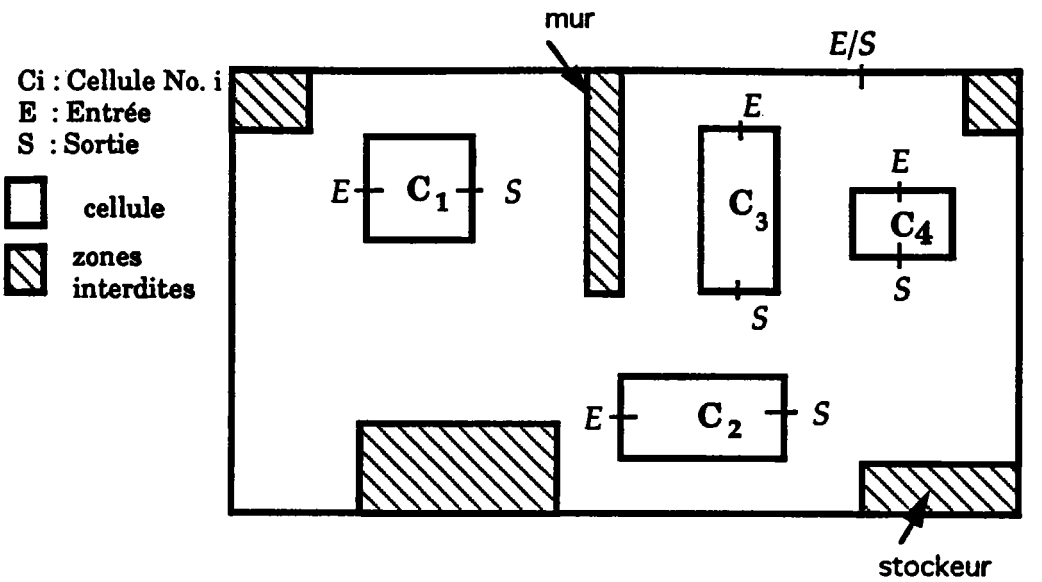


Figure II.6 : Exemple d'une surface d'atelier avec quatre cellules

Le problème consiste à placer les cellules sur la surface de l'atelier de façon à minimiser le critère suivant :

$$\text{Min} \sum_{i,j} \text{trafic}(C_i \rightarrow C_j) \times d_{ij}$$

où :

trafic(C_i -> C_j) est le trafic qui va de la cellule C_i vers C_j et

d_{ij} est la distance entre la sortie de la cellule C_i et l'entrée de C_j, en évitant tous les obstacles, c'est-à-dire la longueur du plus court chemin réel.

L'algorithme utilisé dans les méthodes existantes pour calculer le chemin le plus court entre les cellule est basé sur la théorie des graphes. Dans la suite nous présentons le déroulement de l'algorithme à l'aide de l'exemple de la figure II.7.

L'exemple contient quatre cellules avec leurs entrées et sorties, les zones interdites aux cellules et la surface libre divisée en bandes horizontales parallèles notées, sur notre figure, R_1, R_2, R_3, ..., R_{14}, R_{18}.

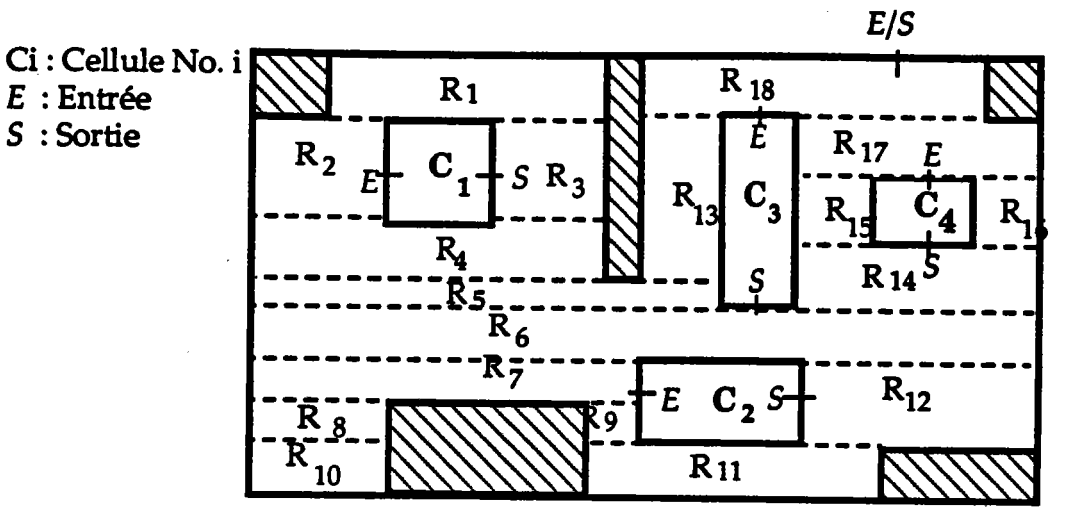


Figure II.7 : Recherche du plus court chemin réel

Certaines bandes sont adjacentes, comme par exemple R_1 et R_2, ou R_4 et R_5, ou R_5 et R_{14}. Partant d'une cellule, il existe une ou plusieurs séquences de bandes à trouver pour atteindre une autre cellule. Chaque séquence est telle que :

- la première bande contient la sortie de la première cellule et la dernière contient l'entrée de la seconde cellule,

- deux bandes qui se suivent dans la séquence sont adjacentes.

Par exemple, pour le couple de cellules (C_2, C_4) de la figure II.7, la séquence (R_{12} -> R_6 -> R_{14} -> R_{15} -> R_{17}) est une telle séquence de bandes à laquelle peut être associée une distance. Le problème à résoudre est donc un problème de recherche de la plus petite distance (ou du plus court chemin) entre deux points. C'est un problème classique d'une complexité de l'ordre du nombre de bandes R_i entre les deux cellules (du nombre de sommets du graphe). L'algorithme de recherche des chemins optimaux (i.e. de longeur minimale) étant connu, on applique l'algorithme du recuit simulé pour trouver une "bonne" disposition des cellules sur la surface disponible. Pour l'application de cet algorithme, il reste à trouver la configuration initiale et la recherche d'une disposition voisine d'une cellule donnée.

Pour la résolution du problème de la configuration initiale, une heuristique qui respecte la contrainte de non-chevauchement mutuel et de non-

chevauchement avec les zones interdites à été proposé par Proth J.M. et Souilah A.(1992).

Pour la recherche d'une configuration voisine de la configuration courante, on se donne deux possibilités d'évolution :

(i) *Permutation de deux cellules.*

Dans ce cas, on choisit deux cellules au hasard. Leurs positions sont définies par rapport à deux axes orthogonaux. Il y a donc, compte tenu de l'existence d'entrées et de sorties, seize positions relatives potentielles pour le couple de cellules. Si toute ces positions violent des contraintes dures, on recommence en choisissant au hasard d'autres inversions, et on s'arrête s'il y a échec plus d'un certain nombre de fois (nombre fixé par l'utilisateur). Si une au moins de ces positions ne viole aucune contrainte dure, on choisit aléatoirement entre les positions admissibles obtenues : ce sera l'état suivant du système.

(ii) *Déplacement d'une cellule.*

On choisit au hasard une cellule et une surface non occupée, et on procède alors comme ci-dessus. Quatre positions sont alors potentiellement possibles.

II.5 CONCLUSION

Ce chapitre a pour objectif de définir le problème d'agencement des systèmes de fabrication dans sa généralité et de présenter les solutions développées dans le cadre du projet ESPRIT No. 5474, COALA (Computer-Aided Manufacturing Layout Design). Les méthodes retenues recouvrent la plupart des approches existant actuellement.

Le problème d'agencement des ressources à l'intérieur des cellules de fabrication est le problème principal de cette thèse, et une méthode de résolution est présentée dans le chapitre suivant.

CHAPITRE III

NOUVELLE APPROCHE POUR L'AGENCEMENT DES RESSOURCES A L'INTERIEUR DES CELLULES DE FABRICATION

III.1 INTRODUCTION

Les produits fabriqués essentiellement à l'intérieur d'une cellule donnée présentent des caractéristiques voisines (i.e. ils appartiennent à la même famille). Pour cette raison, on considère qu'il n'y a qu'un seul type de moyen de transport dans chaque cellule.

Le problème qui se pose est de déterminer d'abord un moyen de transport, puis une configuration de cellule parmi les configurations de base possibles, et enfin un agencement physique des ressources qui optimise cette configuration par rapport à des critères faisant intervenir les déplacements entre les machines de la cellule.

III.2 FORMULATION DU PROBLEME

III.2.1 QUELQUES DEFINITIONS

Dans la suite, nous utiliserons les termes ou expressions suivants :

- Machine : ce terme désigne aussi bien un moyen de fabrication, un poste de travail, une machine de mesure, un poste de nettoyage, un poste de traitement (peinture, thermique, etc.) ou un organe de stockage.

- Site : ce terme désigne un emplacement physique possible pour une machine. Il est définit par un numéro de référence et ses coordonnées x et y, mesurées à partir du point origine de la cellule.

- Flux : le flux entre une machine i et une machine j est le nombre de trajets du moyen de transport entre les machines i et j durant une période donnée.

- Distance entre deux machines : c'est la distance qui sépare les centres des deux machines (c'est la distance qui sera utilisée pour calculer le critère pendant la procédure de placement des machines).

- Distance relative entre deux machines : c'est, suivant le cas, la distance réelle minimale ou maximale à respecter entre deux machines, mesurée à partir du côte de l'enveloppe rectangulaire de chaque machine.

- Indication de distance relative : il s'agit d'un code indiquant si la distance relative entre deux machines est une distance minimale ou maximale (code =

"1" si la distance est maximale, code = "2" si la distance est minimale, code = "0" s'il n'y a pas de contrainte)

- Axe de transport : un axe de transport est une direction de déplacement.

- Distance de transport : c'est l'amplitude d'un déplacement sur un axe de transport.

- Vitesse de transport : c'est la vitesse de déplacement sur un axe de transport.

- Rotation de transport : une rotation de transport est un déplacement angulaire nécessaire pour transporter un produit du système de transport à son emplacement devant la machine.

- Niveau de flexibilité global de la cellule : il caractérise le degré d'adaptation de la cellule face à des changements de son fonctionnement (comme par exemple, l'introduction de variantes de produits dans une famille de produits, la modification locale du chemin emprunté par le moyen de transport, ...). Il est évalué par l'utilisateur et prend les valeurs (important, moyen, faible).

- Niveau de flexibilité du moyen de transport : il caractérise l'aptitude du moyen de transport à s'adapter à des nouvelles configurations de la cellule ou à se prêter à des modifications locales éventuelles du chemin qu'il doit parcourir. Il est évalué par l'utilisateur et prend les valeurs (important, moyen, faible).

- Niveau de sécurité du moyen de transport : il caractérise le risque de détérioration des produits fragiles lors du transport. Il est évalué par l'utilisateur et prend les valeurs (important, moyen, faible).

- Niveau de sécurité des produits : il caractérise la sensibilité des produits aux risque de détérioration lors de leur transport. Il est évalué par l'utilisateur et prend les valeurs (important, moyen, faible).

- Entrée/sortie d'une cellule : la position de l'entrée d'une cellule peut être du même côté que celle de la sortie ou du côté opposé pour les configurations en ligne droite et en ligne double, et sur le pourtour de la cellule pour la configuration en multi-lignes.

III.2.2 DESCRIPTION DU PROBLEME

Pour déterminer le moyen de transport utilisé dans une cellule de fabrication, on doit tout d'abord connaître un certain nombre d'informations relatives aux produits à fabriquer et à manipuler (telles que leur morphologie, leurs dimensions, leur poids). Il est aussi important de savoir si les produits seront ou non montés sur palette (nous parlerons de possibilité de palettisation). On doit également connaître le nombre de machines de la cellule (en effet si on a peu de machines à servir dans un ordre quelconque, on pourra utiliser un robot articulé ; par contre, s'il y a beaucoup de machines avec des produits lourds, on aura peut être intérêt à choisir un pont roulant). Le caractère global du flux de produits (unidirectionnel, bidirectionnel, aléatoire) peut également influencer lè choix du moyen de transport. Cette donnée est fournie par l'utilisateur. De plus, il est important de savoir quels types de déplacements (translations et/ou rotations) sont nécessaires pour positionner les produits devant les machines, dans quelles directions, sur quelles distances et avec quelle précision. Ces aptitudes sont celles du moyen de transport. De plus, un niveau global de flexibilité de la cellule et un niveau de sécurité peuvent être définis par l'utilisateur et pris en compte. Aussi, il peut être souhaitable de prendre en compte le niveau de vibration des machines. Si une ou plusieurs machines provoquent des vibrations importantes, alors on ne peut pas utiliser un robot comme moyen de transport, car celui-ci risque d'être influencé dans ses actions par les vibrations. Dans ce cas, il convient plutôt d'utiliser un pont roulant (robot portique). Finalement, le type de moyen de transport ayant été déterminé, le moyen de transport utilisé dans la cellule est choisi parmi les moyens de transport possibles de ce type, soit arbitrairement par l'utilisateur, soit en comparant les coûts des moyens de transport (ces coûts sont obtenus par addition du coût d'achat, du coût d'installation et d'un coût de maintenance).

Note : Si on voulait tenir compte du coût par unité de distance (convoyeurs, AGVs), il faudrait connaître la position exacte des machines puisqu'elle conditionne les dimensions finales du moyen de transport.

Pour déterminer le type de configuration de base de la cellule le plus approprié, on doit connaître en plus des informations précédentes des informations telles que la longueur, la largeur et éventuellement la hauteur de la cellule (si elles sont données a priori) de même que la longueur, la largeur et éventuellement la hauteur des machines. Egalement, le niveau global de flexibilité de la cellule est important (défini par l'utilisateur).

Pour déterminer le positionnement exact des machines pour une configuration donnée, on doit connaître la distance minimale (ou maximale) à respecter entre certains couples de machines, le flux entre les machines, éventuellement le coût de transport entre les machines. On commence d'abord par affecter des machines à des sites, puis on calcule la position des sites (dans les cas où ces positions n'ont pas été données a priori par l'utilisateur).

III.2.3 LES DONNEES

Les données que nous prenons en compte sont de trois types :

i) les *données de production* concernant les produits, leur fabrication et les moyens de fabrication utilisés. Elles regroupent :

- les types de produits avec leur nomenclature ;
- les gammes de fabrication ;
- les types de machines avec leur description ;
- les types de moyens de transport avec leur description.

ii) les *données géométriques* définissant la forme des objets en présence. Elles regroupent :

- la dimension et la forme des produits, des machines, des moyens de transport et de la cellule ;
- les points d'entrée et de sortie de la cellule, s'ils sont connus a priori ;
- les points singuliers de l'infrastructure (lieux de stockage des déchets, points de distribution d'énergie, etc.).

iii) des *données techniques* concernant le positionnement des produits sur les ressources. Elles regroupent, entre autres :

- la précision de positionnement des produits par rapport au moyen de transport ;
- la vitesse du moyen de transport ;

- le nombre de translations et de rotations du moyen de transport (nécessaires pour positionner la pièce ou le produit sur la machine).

Nous allons, pour chacune d'elles, définir les caractéristiques et le codage utilisé dans le logiciel de résolution. Dans la suite, nous présentons les données avec leur type classées dans l'ordre suivant : caractéristiques des produits, des gammes de fabrication, des machines, des moyens de transport et de la cellule. Certaines de ces données sont optionnelles car elles peuvent ne pas avoir de sens dans certains cas ou ne pas être disponibles.

Caractéristiques des produits :

1. la quantité de produits de chaque type k à fabriquer sur l'horizon choisi, *Q[k]* : entier

2. taille standard des lots de produits à transporter à la fois pour chaque type de produit k, *U[k]* : entier

3. nombre de passages sur les machines (ou positions) dans la gamme de fabrication pour chaque type de produit k, *M[k]* : entier

4. indice de chaque machine pour la position l dans la séquence donnée par la gamme de fabrication de chaque type de produit k, *Ind[l,k]* où l = 1, 2, ..., *M[k]* , k = 1, 2, ..., n, où n est le nombre de produits : entier

5. forme des produits : chaîne de caractères ayant ses valeurs dans {cylindrique, cubique, sphérique} ; optionnelle

6. longueur, largeur, hauteur des produits : 3 réels ; hauteur optionnelle

7. poids des produits : réel

(On prend le poids maximum de tous les produits)

8. possibilité de palettisation des produits : variable booléenne (oui = 1, non = 0)

Caractéristiques des gammes de fabrication :

1. ***F***, matrice de flux, où *f*[i,j] est le flux entre les machines i et j : entiers

2. ***C***, matrice des coûts de transport, où *c*[i,j] est le coût de transport entre les machines i et j : réels ; optionnelle

3. *D*, matrice de distances, où *d*[i,j] est la distance entre les machines i et j : réels

(Initialement inconnue. Elle est calculée pendant la procédure et dépend du type du moyen de transport)

4. *DRI*, matrice d'indications de distance relative, où *dri*[i,j] indique le type de la distance entre les machines i et j ; *dri*[i,j] = 1 (distance maximale), 2 (distance minimale) ou 0 (pas de contrainte)

(Initialement à 0, elle indique s'il y a des positionnements relatifs à respecter entre les machines i et j)

5. *DR*, matrice des distances relatives, où *dr*[i,j] est soit la distance maximale, soit minimale ou inconnue à respecter entre les machines i et j : réels ; optionnelle

6. *P*, matrice des positionnements définitifs des machines sur les sites, où *p*[i,k] est définie comme suit si i est l'indice de machine, i = 1, 2, ..., m et k est l'indice de site, k = 1, 2, ..., l)

$$p[i,k] = \begin{cases} 1, \text{ si la machine i doit être sur le site k} \\ 0, \text{ sinon} \end{cases}$$

7. *S*, vecteur des sites des machines, où *s*[i] indique le numéro de la machine située sur le site i ou 0 si le site est inoccupé. (Initialement inconnu, il indique finalement le numéro de la machine qui se trouve au site i).

8. *CSX*, vecteur des coordonnées des sites selon l'axe des x, où *csx*[i] est l'abscisse du site i à partir du point d'origine choisi pour la cellule.

9. *CSY*, vecteur des coordonnées des sites selon l'axe des y, où *csy*[i] est l'ordonnée du site i à partir du point d'origine choisi pour la cellule.

Caractéristiques des types de machines :

1. longueur, largeur, hauteur des machines : 3 réels ; hauteur optionnelle

2. nombre d'axes de transport nécessaires pour servir les machines en produits : entier égal à 1, 2 ou 3 ; optionnel

3. distances de transport nécessaires sur les axes de transport x, y, et z : 3 réels ; optionnelles

4. vitesses de transport sur les axes x,y, et z : 3 réels ; optionnelles

5. nombre de rotations de transport : entier ; optionnel

6. angles de rotation nécessaire autour des axes x, y et z : 3 réels ; optionnelles

7. vitesses de rotation nécessaire autour des axes x, y et z : 3 réels ; optionnelles

8. précision nécessaire pour le positionnement : réel ; optionnelle

9. niveau de vibration : chaîne de caractères ayant ses valeurs dans {faible, moyen, important} ; optionnel

10. type d'énergie : chaîne de caractères ayant ses valeurs dans {électrique, air sous pression, hydraulique, moteur à combustion} ; optionnel

Caractéristiques des moyens de transport :

1. possibilité de palettisation : variable booléenne (oui = 1, non = 0)

2. nombre maximal de machines que le système de transport peut servir : entier

3. poids maximal transportable : réel

4. longueur, largeur, hauteur maximales des produits à transporter : 3 réels

5. longueur, largeur, hauteur maximales du moyen de transport : 3 réels ; hauteur optionnelle

6. nombre d'axes de transport pour servir les machines en produits : entier

7. distances de transport sur les axes x, y, et z : 3 réels

8. vitesses de transport sur les axes x, y, et z : 3 réels ; optionnelles

9. nombre de rotations de transport : entier ; optionnel

10. angles maximum de rotation autour des axes x, y, et z : 3 réels ; optionnels

11. vitesses de rotation autour des axes x, y, et z : 3 réels ; optionnelles

12. précision pour le positionnement sur les axes x, y, et z : réel ; optionnelle

13. niveau de flexibilité globale du moyen de transport : chaîne de caractères ayant ses valeurs dans {faible, moyen, important} ; optionnel

14. niveau de sécurité : chaîne de caractères ayant ses valeurs dans {faible, moyen, important} ; optionnel

15. niveau de vibration maximal : chaîne de caractères ayant ses valeurs dans {faible, moyen, important} ; optionnel

16. température maximale de l'environnement : entier ; optionnelle

17. type d'énergie : chaîne de caractères ayant ses valeurs dans {électrique, moteur à combustion, air sous pression, hydraulique} ; optionnel

18. coûts : chaîne de caractères ayant ses valeurs dans {petit, moyen, important}, (coûts = coûts d'achat + coûts d'installation + coûts de maintenance sur un certain horizon)

Dans le tableau III.1 suivant se trouvent les caractéristiques de quelques exemples de prototypes du moyen de transport.

MOYENS DE TRANSPORTS / CARACTERISTIQUES	AGV	AGV palette	Robot articulé	Robot palette	Robot poids lourd	Convoyeur	Convoyeur palette	Chariot	Chariot palette	Chariot poids lourd	Chariot palette lourd	Pont roulant 2 dim. léger	Pont roulant 2 dim. moyen	Pont roulant 2 dim. lourd	Pont roulant 3 dim. léger	Pont roulant 3 dim. moyen	Pont roulant 3 dim. lourd	Pont roulant 3 dim. spécial lourd
Possibilité de palettisation	non	oui	non	oui	non	non	oui	non	oui	non	oui	non	non	non	non	non	non	non
Nombre maximal de machines que le système de transport peut servir ∞ = sans limitation	∞	∞	5	5	5	∞	∞	∞	∞	∞	∞	10	10	10	12	15	21	21
Poids maximal transportable [kg]	300	300	10	50	50	300	300	500	500	1000	1000	25	100	300	25	100	300	500
Longueur, largeur, hauteur maximales des produits [m]	2, 1, 1	2, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1	2, 1, 1	2, 1, 1	2, 1, 1	2, 1, 1	2, 1, 1	2, 1, 1	0.5,0.5,0.5	0.5,0.5,0.5	0.5,0.5,0.5	0.5,0.5,0.5	0.5,0.5,0.5	0.5,0.5,0.5	0.5,0.5,0.5
Unidirectionalité du flux des produits	non	non	non	non	non	oui	oui	oui	oui	oui	oui	non	non	non	non	non	non	non
Nombre d'axes de transport pour servir les machines	2	2	3	3	3	2	2	2	2	2	2	2	2	2	3	3	3	3
Distances de transport sur les axes x, y et z [m] ∞ = sans limitation	∞, ∞, 0	∞, ∞, 0	2, 2, 2	2, 2, 2	2, 2, 2	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	8, 0, 3	10, 0, 3	12, 0, 3	6, 6, 3	10, 10, 3	12, 12, 3	12, 12, 3
Vitesses des transports sur les axes x, y et z [m/s]	1, 1, 0	1, 1, 0	2.5,2.5,2.5	0.5,0.5,0.5	0.5,0.5,0.5	1, 1, 0	1, 1, 0	0.5, 0.5,0	0.5, 0.5,0	0.3,0.3, 0	0.25,0.25,0	2.5,0,2.5	1.5,0,1.5	1, 0, 1	2.5,2.5,2.5	1.5,1.5,1.5	1, 1, 1	0.25,0.25,0.25
Nombre de rotations de transport	0	0	3	3	3	0	0	0	0	0	0	3	3	3	3	3	3	3
Angles maximum de rotation autour des axes x, y et z	0, 0, 0	0, 0, 0	360,360,360	360,360,360	360,360,360	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	360,360,360	360,360,360	360,360,360	360,360,360	360,360,360	360,360,360	360,360,360
Vitesses maximales de rotation autour des axes x, y et z [°/s]	0, 0, 0	0, 0, 0	180,180,180	90,90,90	90,90,90	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	180,180,180	120,120,120	90,90,90	180,180,180	120,120,120	90,90,90	60,60,60
Précision du positionnement [mm]	1	2	0.1	1	1	2	2	5	5	10	10	0.1	0.2	0.5	0.1	0.2	0.5	20
Niveau global de flexibilité du moyen de transport	important	important	important	moyen	moyen	petit	petit	important	important	important	important	important	important	important	important	important	important	important
Niveau de sécurité	moyen	moyen	moyen	moyen	moyen	important	important	petit	petit	moyen	moyen	moyen	moyen	moyen	moyen	moyen	moyen	moyen
Température maximale de l'environnement [°C]	50	50	70	70	70	90	90	70	70	70	70	70	70	70	70	70	70	70
Niveau de vibration maximale	moyen	moyen	petit	petit	petit	moyen	moyen	moyen	moyen	moyen	moyen	important	important	important	important	important	important	important
Type d'énergie : E : Electrique, P : Air sous Pression, H : Hydraulique, C : Moteur à combustion	E	E	E, P	E, H	H	E	E	E, C	E, C	E., C.	E, C	E, P	E, H	E, H	E, P	E, H	E, H	E, H
Coûts	moyen	moyen	moyen	moyen	moyen	moyen	moyen	petit	petit	petit	petit	moyen	moyen	moyen	important	important	important	important

Tableau III.1 Exemple des caractéristiques des prototypes du moyen de transport

Caractéristiques de la cellule :

1. nombre de machines dans la cellule, *nb_mach* : entier

2. nombre de type de produits à fabriquer, *nb_prod* : entier

3. longueur, largeur, hauteur maximales de la cellule : 3 réels ; optionnelles

4. position de l'entrée vis à vis de la sortie de la cellule : chaîne de caractères {même coté, en face} ; optionnelle

5. niveau de flexibilité globale de la cellule : chaîne de caractères ayant ses valeurs dans {faible, moyen, important} ; optionnel

6. type d'énergie utilisable : chaîne de caractères ayant ses valeurs dans {électrique, air sous pression, hydraulique} ; optionnel

III.2.4 LES MOYENS DE TRANSPORT ET LES CONFIGURATIONS DE BASE

La procédure de détermination des moyens de transport possibles et de la configuration de base éventuelle est une procédure de sélection basée sur des connaissances très complexes, souvent données par des experts de différents domaines. Cela nous a conduit à développer un système expert pour la résolution de ce problème.

Nous avons considéré les moyens de transport les plus utilisés et les types de configurations de base (standard) les plus fréquents dans l'industrie.

Nous reviendrons sur cet aspect dans le paragraphe III.3.

III.2.5 LE CRITERE D'OPTIMISATION

Le critère d'optimisation à considérer est le suivant (Heragu et Kusiak, 1991) :

$$\text{Min} \quad \sum_{i=1}^{i=n} \sum_{j \neq i} c_{ij} f_{ij} d_{ij}$$

où :

n : nombre de machines

c_{ij} : coût unitaire de transport entre les machines i et j

f_{ij} : trafic entre les machines i et j

d_{ij} : distance entre les machines i et j.

S'il n'y a qu'un type de moyen de transport dans une cellule, alors on peut supposer que le coût c_{ij} associé au moyen de transport entre les machines i et j est égal à un.

Le trafic f_{ij} entre les machines est défini comme le nombre de trajets effectués pour transporter les produits de la machine i à la machine j sur une période donnée.

La distance d_{ij} entre les machines peut être la distance euclidienne, la distance de Manhattan ou la distance réelle. Le choix du type de distance dépend du moyen de transport et du type d'agencement de base considéré. En effet, pour une formation en ligne droite, on utilisera la distance de Manhattan (qui est égale à la distance euclidienne dans ce cas) sur l'axe des x seulement ; pour une formation circulaire, on utilisera la distance réelle (c'est-à-dire la distance qui se calcule par multiplication du rayon du cercle par l'angle formé entre les deux sites des machines et le centre du cercle) ; pour une formation multi-lignes servie par un pont roulant, on utilisera la distance euclidienne; pour une configuration quelconque servie par un AGV, on utilisera la distance réelle. La distance euclidienne et la distance de Manhattan sont définies respectivement comme suit :

$$\text{i)}\quad d_{ij} = ((x_i - x_j)^2 + (y_i - y_j)^2)^{\frac{1}{2}}$$

$$\text{ii)}\quad d_{ij} = \text{abs}(x_i - x_j) + \text{abs}(y_i - y_j)$$

où x_i et y_i sont respectivement l'abscisse et l'ordonnée de la position de la machine i, et abs (u) désigne la valeur absolue de u.

Cet aspect sera repris plus loin.

III.2.6 LES CONTRAINTES

Généralités :

La surface d'un atelier de production peut être divisée en deux types de zones : les *zones interdites* que l'on ne peut pas utiliser pour le placement des ressources (zones réservées, piliers, sources d'alimentation ou d'évacuation, zones de dégagement, etc.); et les *zones libres* où seront placées les ressources.

Lorsqu'on parle de ressources dans un système de fabrication, on parle en principe de trois types de ressources différentes. A savoir :

- les machines ou zones de travail de tout type avec leurs buffers, leurs zones d'outils et zones de sécurité et/ou de manutention associées (c'est le cas le plus courant),

- les systèmes de transport (comme définis préalablement),

- les zones de stockage (stocks tampons, transtockeurs, etc ...).

Toute cellule de fabrication est donc définie in fine par sa morphologie, sa surface, son moyen de transport, ses machines et la position des machines et du moyen de transport sur cette surface en tenant compte des zones interdites.

Notes :

1. Par la suite, nous parlerons par excès de langage de machines et non de ressources, car dans la plupart des cas, ce sont des machines, des zones de travail et des zones de stockage que nous aurons à placer.

2. Dans tous les cas, la surface d'une machine est approximée par une surface de forme rectangulaire par excès. Ainsi, si la machine a des buffers, des magasins d'outils, des zones de sécurité ou de manutention, ..., associés, alors on suppose que ces zones font partie de la machine et on inscrit le tout dans un rectangle qui va finalement représenter la surface totale de la machine. Avec cette hypothèse on ne limite pas le domaine d'application du problème (voir l'exemple de la figure III.1).

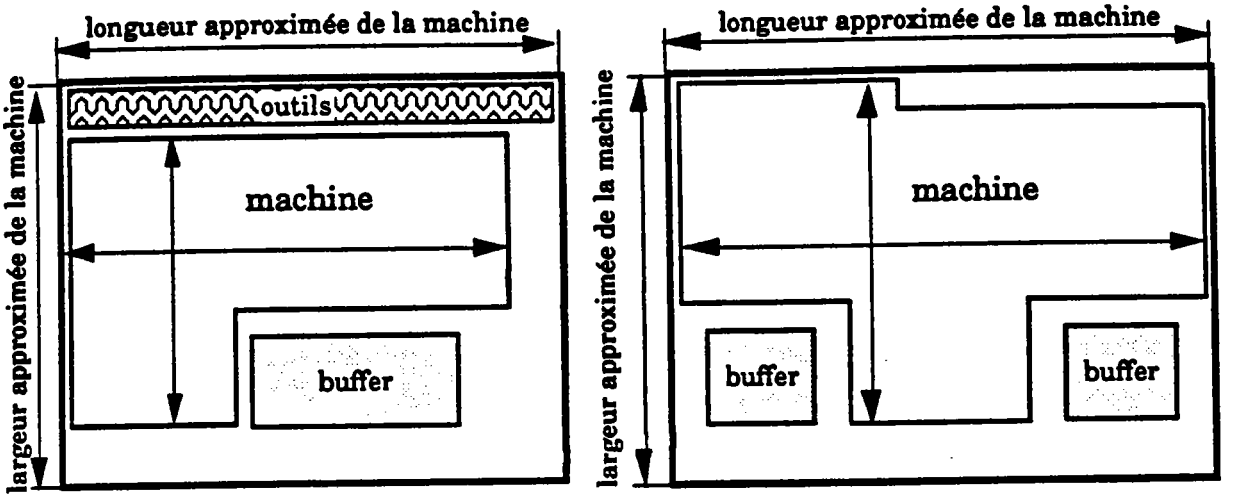


Figure III.1 Deux exemples d'approximation de la forme d'une machine

Contraintes :

Les contraintes ont pour effet d'introduire des limitations dans la recherche d'une solution. Ce sont les suivantes pour ce type de problème :

i) *Contraintes de non chevauchement* entre les machines. Dans la solution finale, il ne doit pas y avoir de chevauchement entre les machines (données par la distance minimale, *dmin* à respecter entre les machines. On l'utilise s'il n'y a pas des contraintes suplémentaires, voir figure III.2).

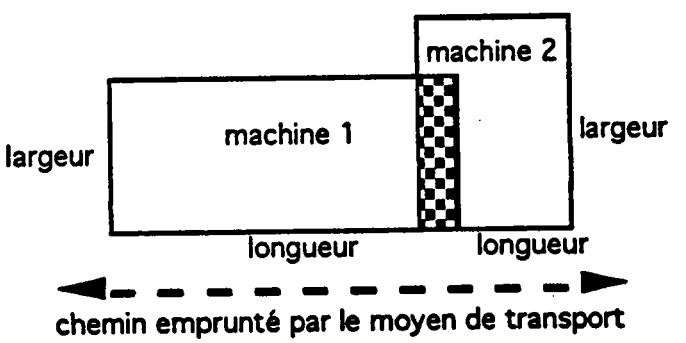


Figure III.2 Exemple de chevauchement entre les machines dans une configuration linéaire

ii) *Contraintes d'éloignement* entre les machines. Certaines machines doivent se trouver à une distance minimale l'une de l'autre pour diverses raisons techniques (sécurité, problèmes de vibration, dégagement d'odeurs ou de poussière, etc.). La figure III.3A donne un exemple pour

deux machines; $distmin_{12}$ indique la distance minimale à respecter entre les machines 1 et 2.

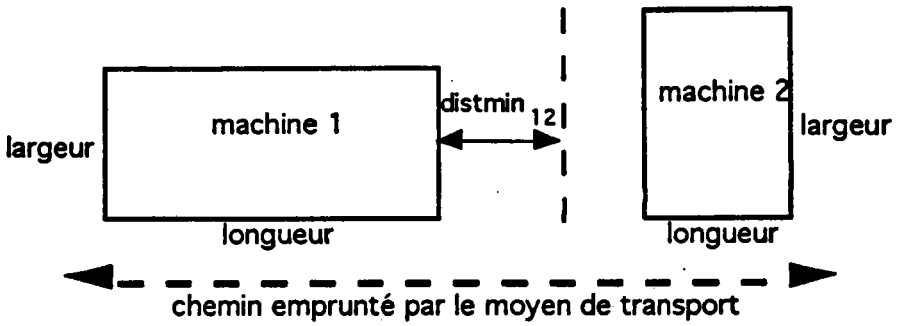


Figure III.3A Exemple d'éloignement entre les machines

iii) *Contraintes de rapprochement* entre les machines. Certaines machines doivent se trouver à une distance maximale l'une de l'autre pour diverses raisons techniques (qualité de produits, fiabilité, nécessité pour le processus de fabrication, etc.). La figure III.3B donne un exemple pour deux machines; $distmax_{12}$ indique la distance maximale à respecter entre les machines 1 et 2.

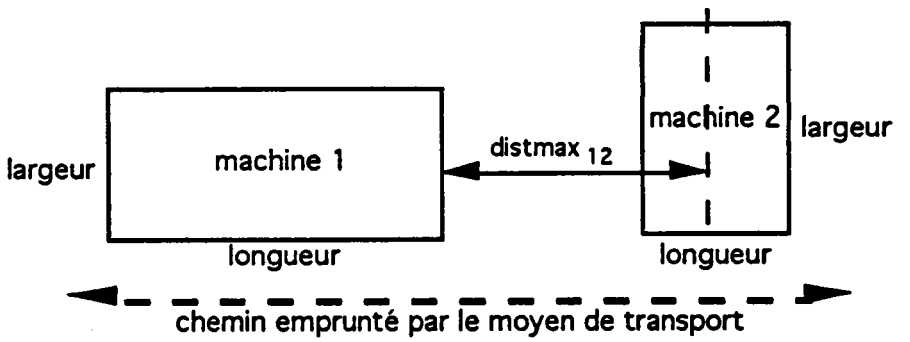


Figure III.3B Exemple de rapprochement entre les machines

iv) *Contraintes d'affectation* de machines à des sites prédéfinis. Dans un atelier de production, on peut être amené à fixer un emplacement précis à une machine. Cet emplacement devra être respecté lors de l'implantation, même au risque d'augmenter le coût d'exploitation de l'atelier (voir figure III.4).

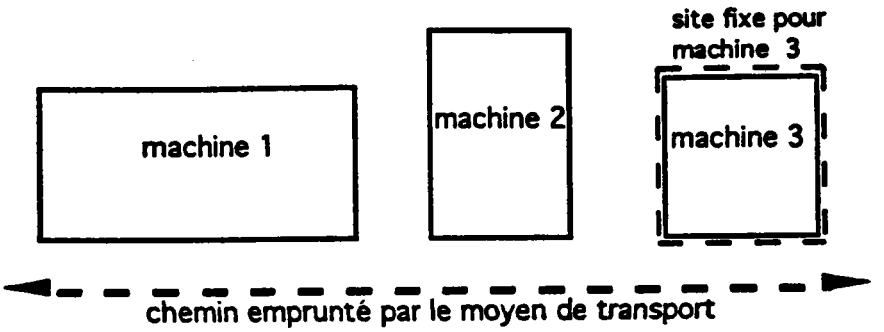
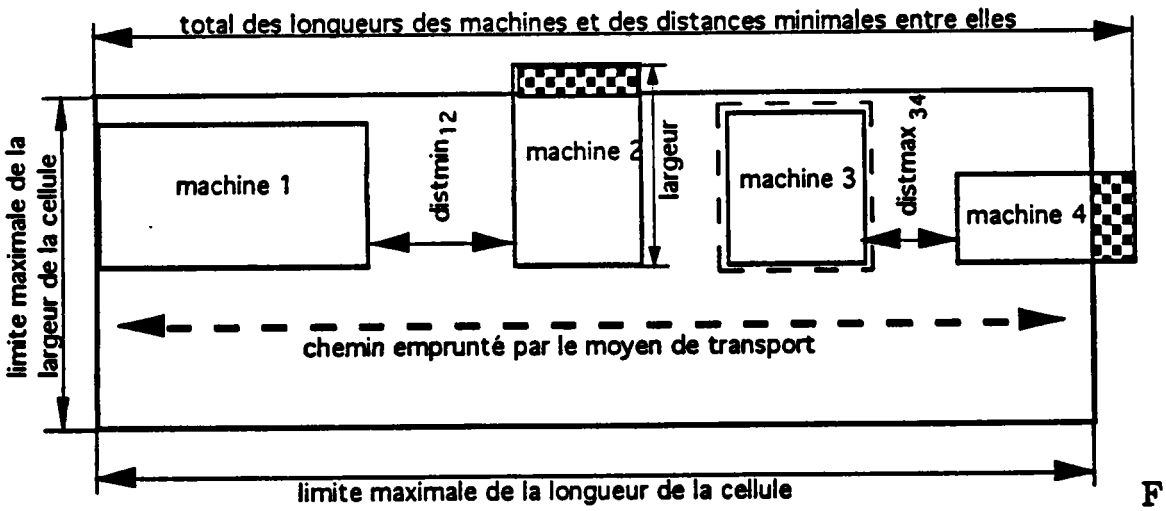


Figure III.4 Exemple d'affectation d'une machine à un site prédéfini

v) *Contraintes sur les dimensions de la cellule.* (Ce problème se pose uniquement lorsque les surfaces des cellules sont limitées). La longueur et la largeur des machines ajoutées à la distance qui les séparent après l'affectation aux sites ne doivent respectivement pas dépasser les limites maximales de la longueur et de la largeur de la surface pour la cellule (voir figure III.5). Si cette contrainte n'est pas respectée, alors il faut augmenter les dimensions de l'atelier ou sélectioner une autre configuration de base.



F

igure III.5 Exemple de contraintes relatives aux dimensions de la cellule

vi) *Contraintes sur l'orientation des machines.* On définit la longueur de chaque machine comme étant le côté par lequel entrent les produits (voir figure III.6). Il se peut ainsi que ce que nous nommons largeur d'une machine soit plus longue que sa longueur d'après cette définition. C'est une convention que nous acceptons dans la suite.

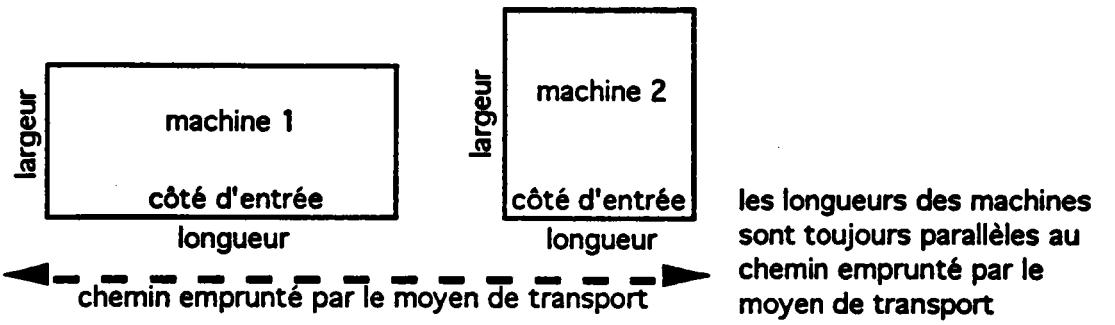


Figure III.6 Exemple d'orientation des machines par rapport au chemin emprunté par le moyen de transport

III.2.7 UN ALGORITHME CONNU DE PLACEMENT DE MACHINES

Un algorithme pour résoudre le problème de placement des machines, développé par Heragu et Kusiak (1988a), est un algorithme ressemblant à l'algorithme dit de "Maximum Spanning Tree". Il est basé sur une approche du type méthode constructive et est applicable pour les configurations en ligne droite et en formation circulaire. Cet algorithme conduit à une méthodologie en deux étapes :

1. Dans la première étape, on détermine la séquence de placement des machines sur la ligne (basée sur le flux de produits).

2. Dans la deuxième étape, on détermine le positionnement exact des machines suivant l'ordre donné par la première étape.

Le critère d'optimisation est la minimisation du flux de produits entre les machines.

Algorithme utilisé pour l'étape 1 :

Etape 0. A partir de la matrice du flux de produits $[f_{ij}]$, calculer

$$f_{i^*j^*} = \max \{ f_{ij} : i = 1, 2, ..., n, j = 1, 2, ..., n \}.$$

Connecter i^*, j^* et les noter dans la solution partielle. Faire $f_{i^*j^*} = f_{j^*i^*} = -\infty$.

Etape 1. Calculer

$f_{p^*q^*} = \max \{ f_{i^*k}, f_{j^*l} : k = 1, 2, ..., n ; l = 1, 2, ..., n\}$ et

(i) connecter q^* to p^* et rajouter q^* à la solution partielle ;

(ii) effacer la ligne p^* et la colonne p^* de $[f_{ij}]$;

(iii) si $p^* = i^*$, alors faire $i^* = q^*$; sinon, faire $j^* = q^*$.

Etape 2. Recommencer à l'étape 1 jusqu'à ce que la solution finale soit trouvée (i.e., jusqu'à ce que toutes les machines soient inclues dans la solution).

où : $[f_{ij}]$ est le flux de produits entre les machines i et j et n est le nombre de machines.

Critère utilisé pour l'étape 2 :

Le positionnement exact des machines dépend du moyen de transport, de la distance nécessaire entre les machines et de leur orientation. Si un robot est utilisé comme moyen de transport, alors, en partant de la séquence obtenue par l'algorithme de l'étape 1, les machines sont agencées de manière circulaire avec un rayon déterminé par l'élongation maximale du bras du robot, et le robot sera positionné au centre du cercle. Si un AGV ou un convoyeur sont utilisés comme moyen de transport, alors les machines sont positionnées en ligne droite. Pour déterminer l'orientation des machines, des facteurs comme la forme de la machine et le côté de chargement/déchargement sont à considérer. La distance relative à considérer entre les machines est donnée par les experts.

Pour la formulation et la résolution du problème, on peut utiliser les algorithmes décrits dans la section III.4.2 suivant le moyen de transport choisi.

A cause de sa simplicité, cette approche peut permettre de résoudre rapidement des problèmes peu contraints dans le cas de configurations linéaires. Cependant, les problèmes de choix du moyen de transport et de l'orientation des machines sont négligés. De plus, l'algorithme n'est pas facilement généralisable aux autres types de configurations de base. Enfin, l'algorithme garantit seulement une solution optimale pour le problème avec moins de quatre machines par cellule (Heragu et Kusiak, 1988a).

Dans la suite, nous allons présenter une méthode plus globale qui prend en compte à la fois la distance et le flux entre les machines, et qui est applicable aux différents types de configurations que nous considérons. D'autres contraintes, comme les sites fixes pour certaines machines, ou le positionnement des entrées/sorties de la cellule, sont également prises en compte. Cette méthode peut encore être utilisée si l'on choisit le critère (coût x distance x flux).

III.2.8 LA METHODOLOGIE UTILISEE

La méthode que nous avons développée pour l'agencement des machines à l'intérieur des cellules se déroule comme suit (Figure III.7) :

1. A partir des caractéristiques des produits, des machines, des gammes de fabrication à traiter, et des moyens de transport existants sur le marché, on cherche une liste de tous les moyens de transport possibles. Pour cette étape, on a développé un système expert (Système Expert I).

2. Ensuite, on donne la main à l'utilisateur pour qu'il sélectionne un moyen de transport dans cette liste de moyens de transport possibles. S'il ne fait aucun choix, alors le système va choisir le moyen de transport le plus économique. Si le système expert ne trouve pas de moyen de transport possible, alors il donne à l'utilisateur les raisons de cet échec. L'utilisateur peut alors modifier les données pour que le problème ait une solution.

3. A partir du moyen de transport sélectionné et de toutes les données sur les produits, les machines et les gammes de fabrication, on cherche les types d'agencement de cellules possibles parmi cinq types d'agencement de base considérés et caractérisés en base de données (voir paragraphe III.3.3). Pour cela, nous avons également développé un système expert (Système Expert 2).

4. On donne à nouveau la main à l'utilisateur pour qu'il fasse le choix d'une configuration de base parmi la liste des configurations proposées. S'il ne fait aucun choix, alors le système choisit la configuration la plus simple parmi les configurations possibles. Si le système expert ne trouve pas de configuration possible, alors il donne à l'utilisateur les raisons de cet échec. L'utilisateur peut encore une fois modifier les données responsables de cette situation, ou sélectionner un autre moyen de transport pour qu'il puisse y avoir une ou plusieurs configurations possibles. Si aucune configuration n'est possible, alors

l'utilisateur peut proposer une configuration quelconque. Dans ce cas, l'utilisateur doit rentrer toutes les coordonnées des sites possibles pour placer les machines avec la contrainte suivante : le nombre de sites ≥ le nombre des machines à placer. Ensuite, en fonction du moyen de transport, la distance entre les sites sera calculée.

5. A ce stade, le moyen de transport et le type de configuration de base ont été choisis. Il reste maintenant à placer physiquement les machines en accord avec cette configuration de façon à minimiser les trajets des produits à l'intérieur de la cellule. Ce problème est un problème d'optimisation combinatoire. Pour la résolution de ce problème, nous utilisons la méthode du Recuit Simulé. Le résultat final donne le moyen de transport et l'agencement détaillé des machines. L'utilisateur peut intervenir et peut soit accepter ce résultat, soit relancer le processus à différents niveaux.

La méthodologie générale de la résolution du problème d'agencement des ressources à l'intérieur des cellules de fabrication est résumée par la figure III.7.

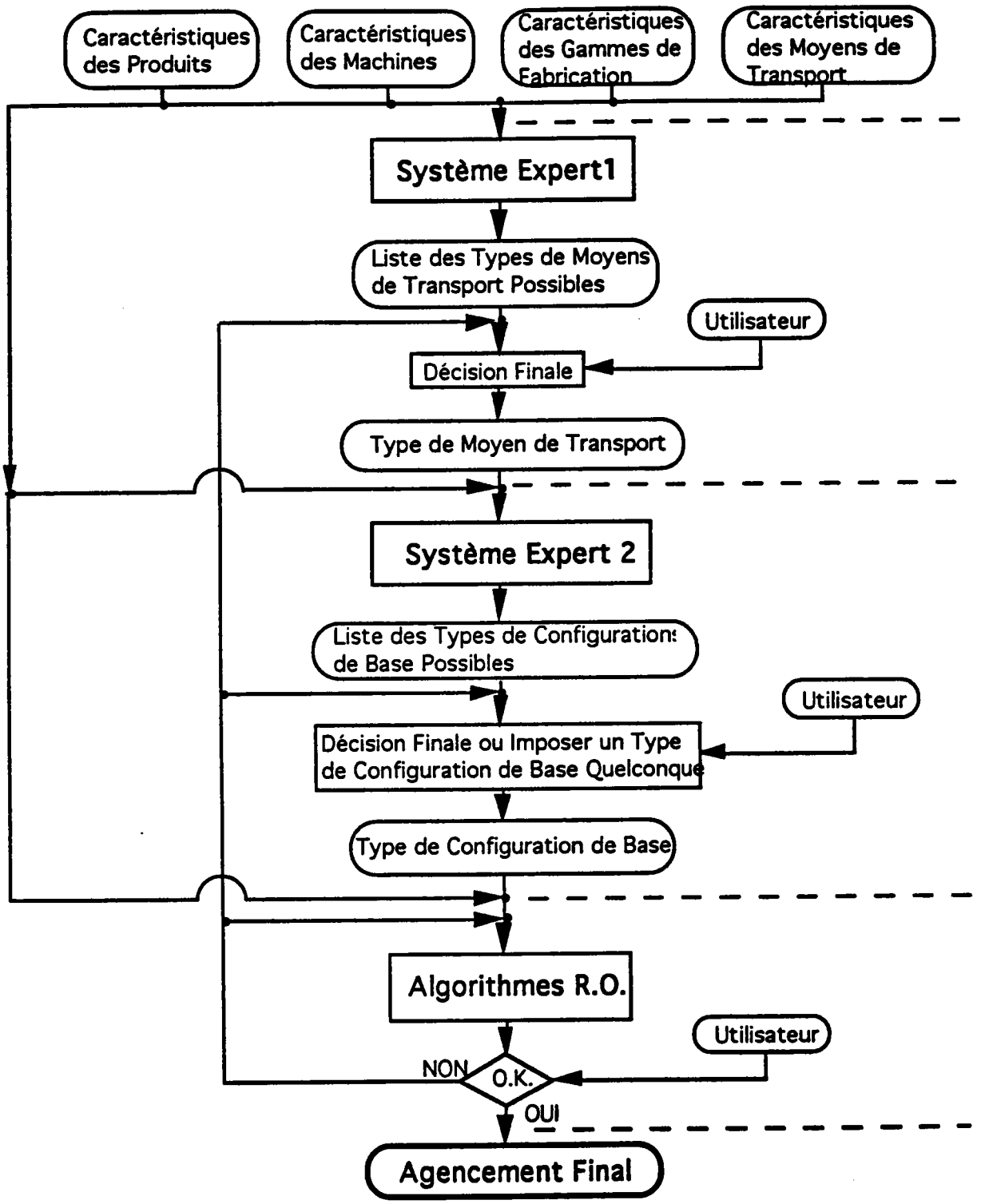


Figure III.7 Méthodologie générale de résolution du problème d'agencement des ressources à l'intérieur des cellules de fabrication

III.3 SYSTEME EXPERT

III.3.1 INTRODUCTION

Un système expert cherche à reproduire l'expertise humaine utilisée pour résoudre un problème particulier. Pour cela, il utilise des connaissances de base ou *faits*, stockés dans une base de données et décrivant l'état initial du problème. Il utilise également une *base de connaissances* qui contient l'expertise nécessaire pour résoudre le problème. Généralement, cette expertise est stockée sous forme de règles de la forme SI <condition> ALORS <action>, appelées *règles de production* et définies au moyen de la logique des propositions ou des prédicats. D'autres formes de descriptions existent comme les réseaux sémantiques. Enfin, il utilise un *moteur d'inférence* qui fournit un mécanisme de base pour l'utilisation conjointe de la base de données et de la base de connaissances pour dérouler un raisonnement au sujet de problème et aboutir à une conclusion.

Le problème de sélection des moyens de transport possibles et des configurations de base à utiliser pour l'agencement des systèmes de fabrication est un problème de ce type, pour lequel il est nécessaire de faire appel à l'expertise de spécialistes. C'est la raison pour laquelle nous avons décidé de développer un système expert, ou plus précisément un système à base de connaissances, pour la résolution du problème de sélection des moyens de transport possibles et pour le choix des configurations de base à utiliser pour l'agencement de la cellule.

Pour développer ce système expert, nous avons utilisé le logiciel de développement "SMECI" d'ILOG S.A., France.

III.3.2 SMECI

SMECI est un environnement de programmation spécialement conçu pour réaliser des systèmes à base de connaissances dans des domaines variés, tels que la conception, la simulation, le diagnostic de systèmes complexes ou la planification. SMECI constitue un "*environnement*" de développement d'applications permettant de représenter divers types de "*connaissances*", et de

mettre en oeuvre différentes formes de "*raisonnements*" (ou moteurs d'inférence) exploitant les connaissances. Les points 1,2, et 3 suivants sont extraits de la documentation SMECI (ILOG S.A., 1991).

1. L'environnement

L'environnement de développement de SMECI se caractérise par les éléments suivants :

- une représentation à base de *catégories*[1], de *prototypes* et d'*objets*, dans laquelle les catégories permettent de définir la *structure* des objets manipulés, les prototypes regroupent les connaissances à l'intérieur d'une catégorie, et les objets[2] modélisent une description particulière d'un problème à résoudre ;

- des *règles* de production d'ordre 1, organisées en *bases de règles*, elles-mêmes associées à des *tâches*. L'ensemble des bases de règles forme une base de connaissances. Cette organisation permet de structurer les connaissances et simplifie leur mise en oeuvre ;

- un moteur d'inférence gérant un agenda de tâches. Les tâches modélisent les composantes élémentaires du raisonnement. L'agenda détermine dynamiquement l'exécution des tâches ;

- une gestion de *mondes multiples*. Le raisonnement en SMECI produit un *arbre d'états*, dans lequel chaque *état* donne une vue instantanée de l'ensemble des objets modifiés lors de la résolution du problème par l'application. L'utilisateur peut en outre définir une *fonction d'évaluation* d'état, qui sera utilisée par le moteur d'inférence : lors de chaque cycle, le moteur poursuit son raisonnement sur l'état jugé le meilleur par la fonction d'évaluation ;

- un mécanisme de *valeurs actives* et de *démons*, qui permet de programmer des actions "réflexes" déclenchées dynamiquement par les modifications sur les objets ;

---

[1] Dans la terminologie habituelle des environnements orientés objets, une catégorie est appelée *classe d'objets*.

[2] Les objets sont des instances de catégories.

- un environnement de développement graphique multi-fenêtres comprenant des éditeurs dédiés et facilitant les tâches de programmation et de maintenance des *bases de connaissances* ;

- des utilitaires de stockage des connaissances sur fichier, ainsi que de tout ou partie des résultats d'une session ;

- en standard, l'environnement de développement d'interfaces graphiques *AIDA* ;

- la possibilité de communiquer avec des logiciels écrits en FORTRAN ou en C, et donc de réutiliser des programmes existants.

2. Les connaissances

Trois types de connaissances sont à distinguer dans un système développé avec SMECI :

- les connaissances factuelles (ou faits), représentées sous forme d'objets (catégories, prototypes) ;

- les connaissances procédurales, comprenant les *méthodes*, les *démons*, les *valeurs actives* et les *fonctions* LISP ;

- les connaissances déclaratives, énoncées sous la forme de règles.

Chaque représentation est adaptée à un type de connaissances particulier. La formalisation des connaissances suppose le plus souvent la recherche d'un style de représentation "idéal" en fonction de critères tels que l'efficacité, la lisibilité, l'accessibilité, la simplicité, etc... La variété des formes de connaissances énoncées ci-dessus tente de répondre à la diversité de ces critères.

3. Le raisonnement

En SMECI, le raisonnement peut être décrit sous forme d'une succession de traitements élémentaires à partir de la base de connaissances. Cette décomposition se fait en définissant des tâches qui matérialisent les étapes du raisonnement de manière expressive.

Dans le même esprit, un gestionnaire de raisonnement synthétise les informations utiles à l'initialisation et au suivi du raisonnement. En particulier, il référence l'arbre des états du système.

Nous nous sommes limités à une courte introduction au logiciel de développement des systèmes à base de connaissances SMECI. Pour des informations plus détaillées, nous renvoyons le lecteur à la documentation du logiciel SMECI (ILOG S.A., 1991).

III.3.3 CHOIX DU MOYEN DE TRANSPORT

III.3.3.1 Types de moyens de transport

Les cellules de fabrication sont composées de groupes de machines dédiés à des familles de produits. Chaque famille de produits contient des types de produits voisins, c'est-à-dire ayant à peu près les mêmes caractéristiques et qui utilisent les mêmes machines. Aussi, le nombre des machines par cellule est souvent petit (inférieur à 10). Pour ces raisons, on considère qu'il n'y a qu'un seul type de moyen de transport dans chaque cellule pour manipuler tous les produits. Comme moyens de transport, nous avons considéré les moyens de transport standards qui sont utilisés dans l'industrie (voir figure III.8) : robots, ponts roulants, chariots filo-guidés (automated guided vehicles ou AGV), convoyeurs et chariots manuels.

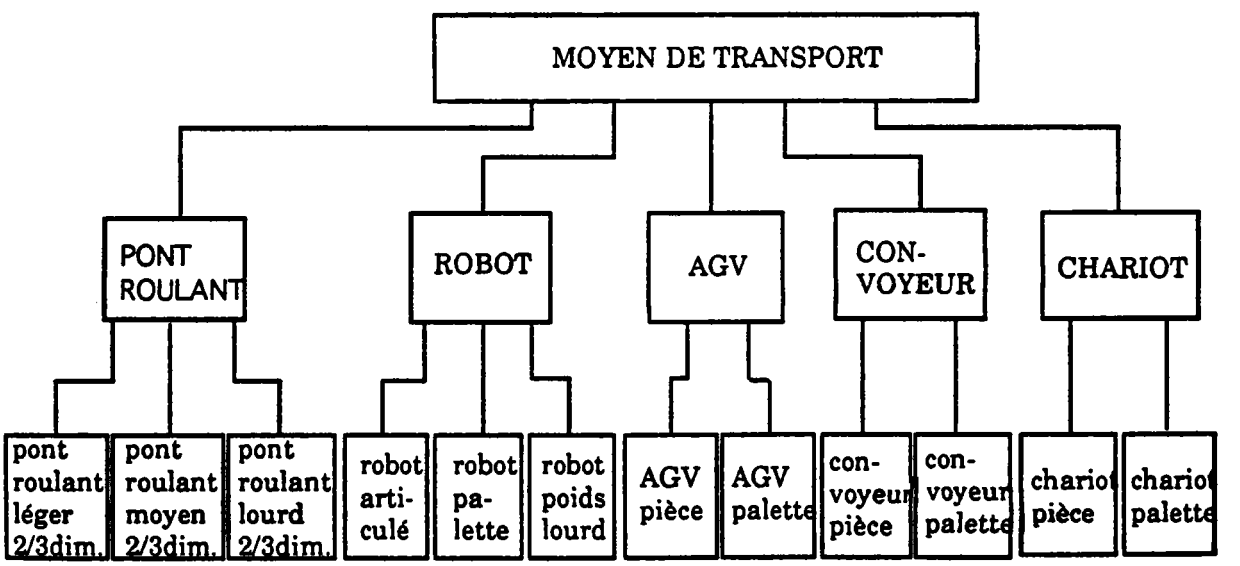


Figure III.8 Moyens de transport considérés

Les robots (voir figure III.9) peuvent encore être classés en robots articulés, robots palettes (qui sont capables de manipuler des palettes) et en robots poids lourds (pour transporter les produits lourds). Les ponts roulants (voir figure III.10) peuvent être classés en ponts roulants à deux et trois dimensions et

encore en poids léger, moyen et lourd. Les véhicules filo-guidés (voir figure III.11), les convoyeurs (voir figure III.12) et les chariots (voir figure III.13) peuvent transporter soit des pièces, soit des palettes et sont différenciés en fonction des différents poids et moyens de traction.

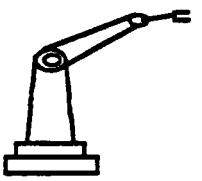


Figure III.9 Robot

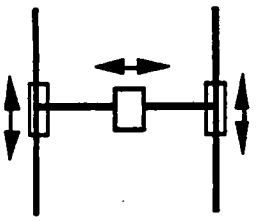


Figure III.10 Pont roulant

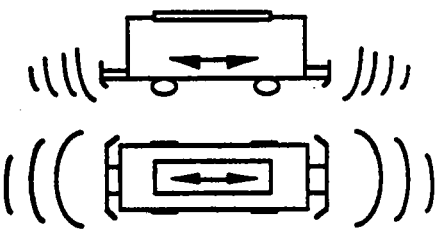


Figure III.11 AGV

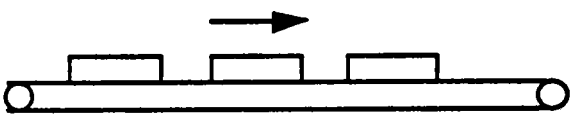


Figure III.12 Convoyeur à bande ou à chaîne, etc.

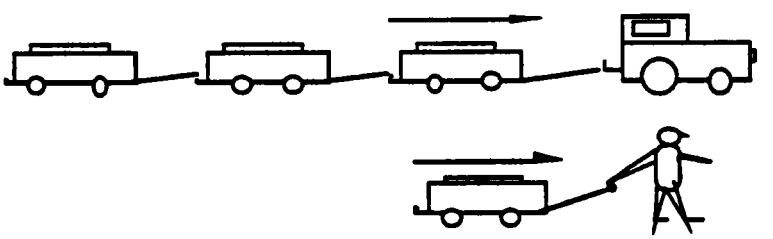


Figure III.13 Chariot manuel ou train

III.3.3.2 Méthode de résolution

Pour la résolution du problème du choix du moyen de transport, nous avons créé une base de données et une base de connaissances.

La base de données contient les données de base nécessaires pour résoudre le problème. Les données nécessaires sont les caractéristiques des produits, de

leurs gammes de fabrication, des machines et de la cellule. Elles sont données par l'utilisateur et sont stockées sous forme d'objets dans le système.

La base de connaissances se compose de règles de déduction pour résoudre le problème de sélection du moyen de transport. Les règles sont basées sur les caractéristiques de tous les moyens de transport considérés (voir au-dessus). La figure III.14 représente une partie de la hiérarchie des caractéristiques qui sont considérées dans les règles du système expert pour le choix du moyen de transport. L'arbre de règles commence par considérer la possibilité de palettiser ou non les produits. Ensuite, le nombre de machines dans une cellule devient un facteur discriminant. Si ce nombre est, par exemple, plus petit ou égal à cinq, alors un robot peut a priori être considéré comme un moyen de transport possible. Le poids maximal des produits est aussi une caractéristique importante qui peut être considérée à ce stade du raisonnement. Par exemple, les produits lourds (> 500 kg) ont besoin d'un convoyeur ou d'un pont roulant lourd. Ensuite, le nombre de translations de positionnement pour placer les produits sur les machines devient un facteur à prendre en compte. Finalement, des contraintes supplémentaires (non représentées dans la figure III.14) sont utilisées pour sélectionner le type de moyen de transport convenable.

Toutes les caractéristiques à prendre en compte pour sélectionner un Robot Articulé sous la forme d'une règle de production qui peut être codée en SMECI sont illustrées après la figure III.14.

Il est possible que le système expert trouve plusieurs types de moyen de transport possibles. Dans ce cas, soit l'utilisateur décide d'en sélectionner un, soit il demande au système de choisir le moyen de transport le plus économique.

Il est aussi possible que le système expert ne trouve aucun type de moyen de transport possible. Dans ce cas, le système en donne l'explication. L'utilisateur peut alors modifier certaines données de la base de données qui conduisaient éventuellement à des contradictions. Ensuite, le système va de nouveau essayer de trouver des types de moyen de transport possibles. On itère jusqu'à ce qu'au moins un type de moyen de transport puisse être choisi, ou que l'utilisateur impose une solution.

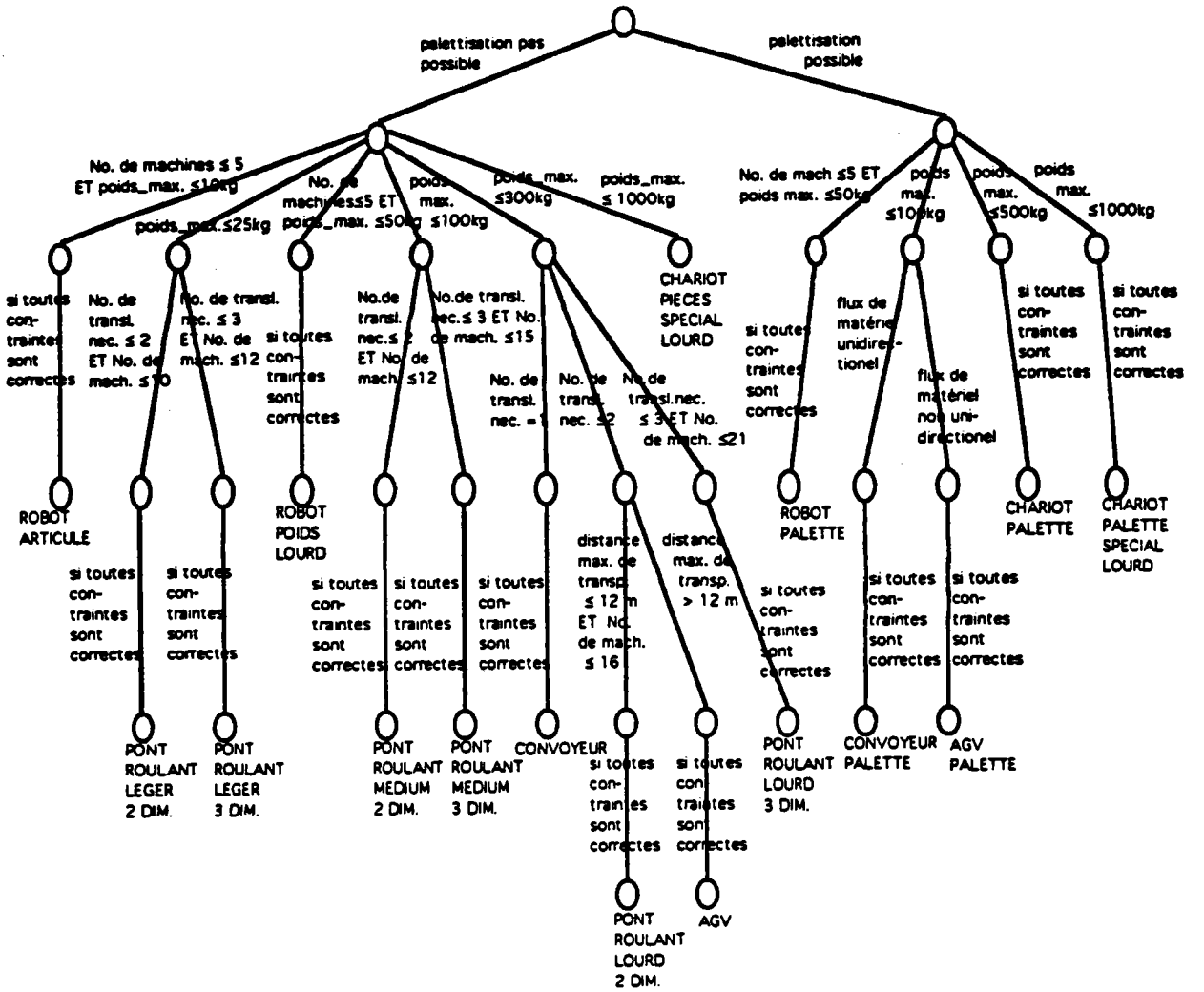


Figure III.14 Exemple des règles de choix du type de moyen de transport

Règles pour choisir un Robot Articulé

Si le nombre de machines dans la cellule ≤ à 5

et les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation = '**faux**')

et le poids maximal des produits ≤ au poids maximal des produits qu'un robot articulé peut transporter (10 kg)

et la longueur maximale des produits ≤ à la longueur maximale des produits qu'un robot articulé peut transporter (1 m)

et la largeur maximale des produits ≤ à la largeur maximale des produits qu'un robot articulé peut transporter (1 m)

et la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un robot articulé peut transporter (1 m)

et la vitesse de transport nécessaire pour les produits ≤ à la vitesse de transport d'un robot articulé (2,5 m/s)

et la précision maximale du positionnement des produits sur les machines ≤ à la précision de positionnement d'un robot articulé (0,1 mm)
et le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un robot articulé (important)
et le flux de matériel n'est pas unidirectionnel
et le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un robot articulé (3)
et la distance de transport nécessaire (l'amplitude de déplacement) sur les axes ≤ la distance maximale de transport d'un robot articulé (2m)
et le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un robot articulé (3)
et le niveau de flexibilité du système de fabrication ≤ au niveau de flexibilité d'un robot articulé (important)
et la température maximale de l'environnement ≤ à la température d'environnement maximale pour un robot articulé (70°C)
et le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un robot articulé (petit)
et le type d'énergie disponible = au type d'énergie d'un robot articulé (électrique)

Alors utiliser **Robot Articulé** comme moyen de transport.

III.3.3.3 Résultat

Le résultat obtenu est le ou les moyen(s) de transport sélectionné(s). Toutes ses caractéristiques sont conformes aux données des produits, de leurs gammes de fabrication, des machines et de la cellule.

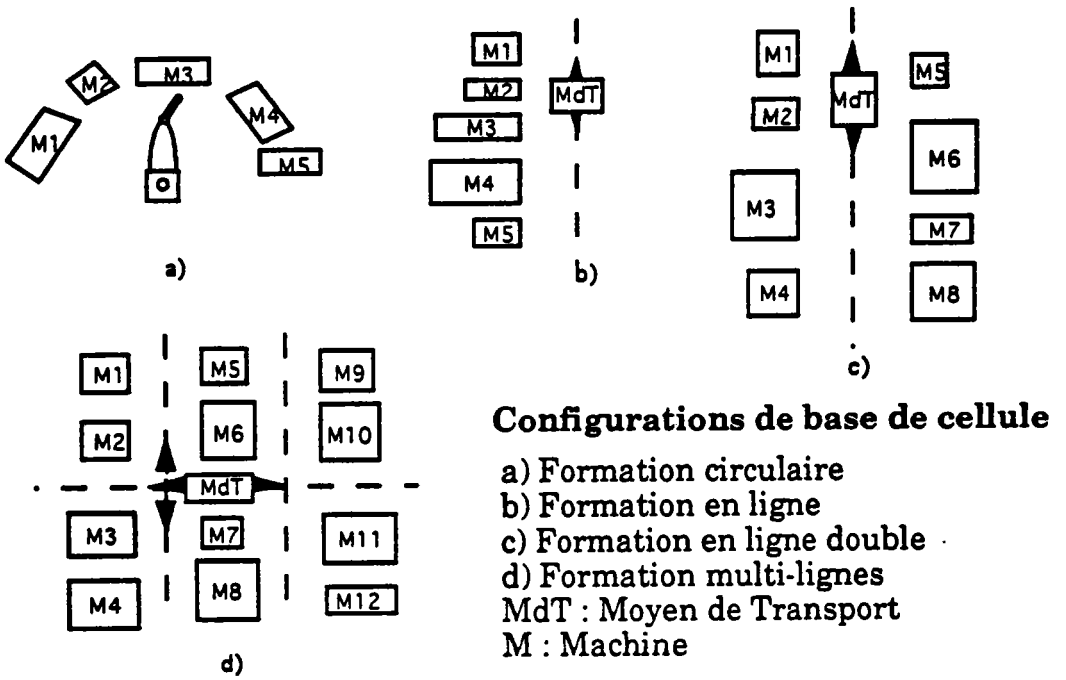
La procédure continue avec toutes les caractéristiques du moyen de transport sélectionné pour chercher les types d'agencement de bas à utiliser. C'est l'objet du paragraphe III.3.4.

III.3.4 CHOIX DU TYPE D'AGENCEMENT DE BASE

III.3.4.1 Types d'agencement de base

Dans les systèmes de fabrication organisés en cellules, on distingue principalement quatre types d'agencements de base (voir figure III.15) :

- configuration circulaire,
- configuration en ligne droite,
- configuration en ligne double et
- configuration multi-lignes.



Configurations de base de cellule

a) Formation circulaire
b) Formation en ligne
c) Formation en ligne double
d) Formation multi-lignes
MdT : Moyen de Transport
M : Machine

Figure III.15 Les configurations de base de cellule

Dans certains cas, il n'est pas possible de classer une configuration au moyen des quatre classes principales de base. Dans ce cas il est nécessaire d'introduire la possibilité d'une *configuration quelconque*. Cette configuration quelconque doit être donnée par l'utilisateur avec toutes les coordonnées des points possibles pour les sites des ressources (voir figure III.16).

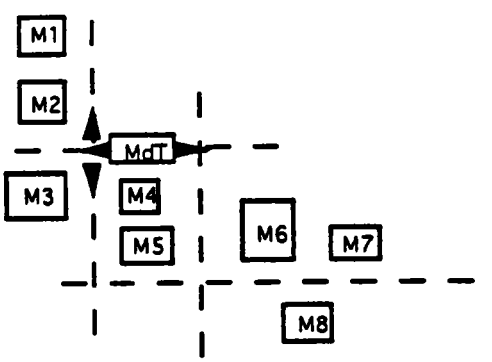


Figure III.16 Exemple de configuration quelconque de cellule

III3.4.2 Méthode de résolution

Pour la résolution du problème du choix du type de configuration de base, nous avons créé une deuxième base de données et une deuxième base de connaissances.

La base de données contient les données nécessaires pour résoudre le problème. Les données nécessaires sont les caractéristiques des machines, les caractéristiques des gammes de fabrication des produits, celles de la cellule et du moyen de transport choisi par la partie précédant du système expert et l'utilisateur. Elles sont stockées sous forme d'objets dans le système.

La base de connaissances se compose de règles de déduction pour résoudre le problème de sélection du type de configuration de base. Le principe appliqué est le même que pour le choix du type de moyen de transport. Les règles sont basées sur les caractéristiques de toutes les configurations de base considérées (voir figureIII.15). La figure III.17 représente une partie de l'arbre des caractéristiques qui sont considérées dans les règles du système expert pour le choix du type de configurations de base. L'arbre de règles commence par considérer le nombre de machines dans la cellule. Ensuite, le niveau de la flexibilité générale du système de fabrication de la cellule devient un facteur discriminant. Si, par exemple, l'utilisateur veut garder une flexibilité importante de la cellule, alors une configuration multi-lignes sera préférable à une configuration en ligne simple ou en ligne double, ou une configuration en ligne double sera préférable à une configuration en ligne simple. Les positions relatives de l'entrée et de la sortie de la cellule, si elles sont connues a priori, peuvent être considérées à ce stade du raisonnement. Ensuite, les dimensions maximales de la cellule (la longueur et la largeur), si elles sont connues a priori, deviennent un facteur à prendre en compte. Finalement, des contraintes supplémentaires (non représentées dans la figure III.17) sont utilisées pour choisir la configuration de base convenable.

Toutes les caractéristiques à prendre en compte pour sélectionner, par exemple, la configuration de base ligne droite, ligne double, multi-lignes et configuration circulaire sous la forme des règles de production, sont illustrées dans la figure III.17. Les règles peuvent être codées en SMECI.

Il est possible que le système expert trouve plusieurs types de configurations de base possibles. Dans ce cas, l'utilisateur doit en choisir une, ou alors le système choisit le type de configuration de base le plus simple, c'est-à-

dire qu'il choisit une configuration en ligne double avant une configuration en multi-lignes, et une configuration en ligne simple avant une configuration en ligne double ou en multi-lignes, parmi les configurations possibles. Si le système expert ne trouve aucune configuration de base possible, alors il en donne l'explication à l'utilisateur. L'utilisateur peut relancer le programme en fonction de ces explications, ou encore introduire une configuration qu'il aura lui-même définie.

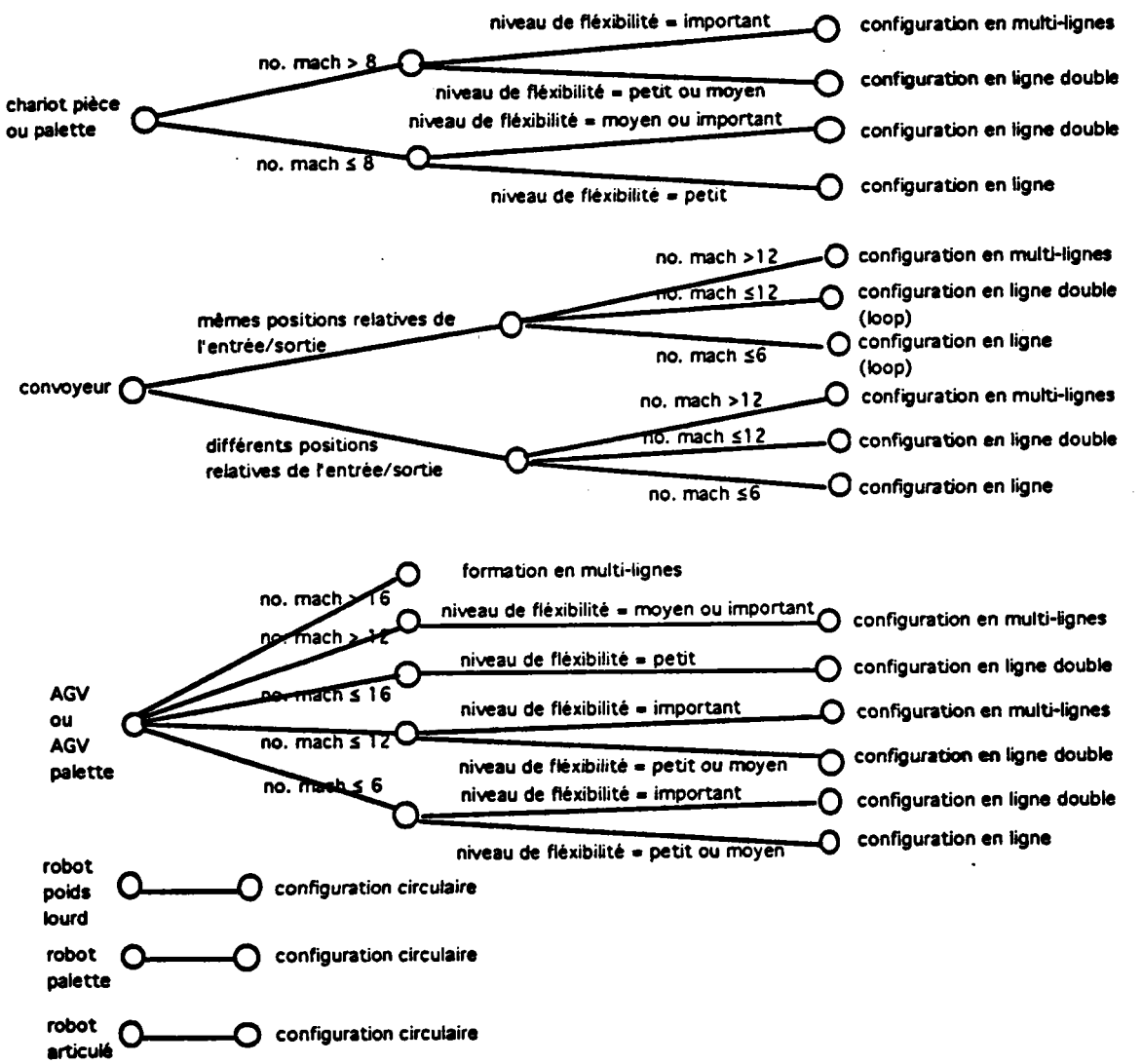


Figure III.17 Exemple des règles de choix de configuration de base

Règles pour choisir la configuration de base en ligne droite

Si le type du moyen de transport choisi est un AGV

et le nombre de machines dans une cellule est ≤ 6

et le flux de matériel n'est pas unidirectionnel
et le niveau de flexibilité du système de fabrication est "petit"
et les longueurs des machines ajoutées aux distances minimales qui les séparent ≤ à la longueur de la cellule (uniquement si les dimensions de la cellule sont limitées)
Alors utiliser la configuration **ligne droite** comme configuration de base pour la cellule.

Règles pour choisir la configuration de base en ligne double

Si le type de moyen de transport choisi est un AGV
et le nombre de machines dans une cellule est ≤ 12
et le flux de matériel n'est pas unidirectionnel
et le niveau de flexibilité du système de fabrication est "faible" ou "moyen"
et la longueur et la largeur des machines ajoutées aux distances minimales qui les séparent ≤ à la longueur et la largeur respectivement de la cellule (uniquement si les dimensions de la cellule sont limitées)
Alors utiliser la configuration **ligne double** comme configuration de base pour la cellule.

Règles pour choisir la configuration de base multi-lignes

Si le type de moyen de transport choisi est un AGV
et le nombre de machines dans une cellule appartiennent à [7,..., 12]
et le niveau de flexibilité du système de fabrication est "important"
et le flux de matériel n'est pas unidirectionnel
Alors utiliser la configuration **multi-lignes** comme configuration de base pour la cellule.

ou :

Si le type du moyen de transport choisi est un AGV
et le nombre de machines dans une cellule est > 12
et le niveau de flexibilité du système de fabrication est "moyen" ou "important"
et le flux de matériel n'est pas unidirectionnel
Alors utiliser la configuration **multi-lignes** comme configuration de base pour la cellule.

ou :

Si le type du moyen de transport choisi est un AGV
et le nombre de machines dans une cellule est > 16

et le flux de matériel n'est pas unidirectionnel
Alors utiliser la configuration **multi-lignes** comme configuration de base pour la cellule.

Règles pour choisir la configuration de base circulaire
Si le type du moyen de transport choisi est un robot
Alors utiliser la configuration **circulaire** comme configuration de base pour la cellule.

III.3.4.3 Résultat

Le résultat obtenu est soit la ou les configuration(s) de base sélectionnée(s), soit une configuration quelconque donnée par l'utilisateur. Dans le cas d'un résultat obtenu par le système expert, toutes les caractéristiques du ou des configurations proposées sont conformes aux données des machines, des gammes de fabrication des produits, celles de la cellule et du moyen de transport choisi par la partie précédante du système expert et des choix de l'utilisateur. La procédure continue avec toutes les caractéristiques du moyen de transport et le type de configuration de base choisi pour chercher l'agencement physique des machines sur cette configuration. C'est l'objet du paragraphe III.4.

III.4 AGENCEMENT ET EVALUATION

III.4.1 Préliminaires

Le problème consiste à déterminer la position physique des machines, compte tenu du moyen de transport et de la configuration de base choisie à l'intérieur de la cellule, de façon à minimiser le flux de produits entre les machines pondéré par les distances. Ce problème est un problème d'optimisation combinatoire.

Pour chaque type de configuration de base, la résolution du problème doit être effectuée un peu différemment, car il y a différentes possibilités de positionnement des machines et les contraintes sont également différentes (voir paragraphe III.4.3). On doit prendre en compte, d'une part, les données globales pour toutes les configurations de base, et d'autre part, les données particulières propres à chaque type de configuration. De plus, la manière de calculer la distance entre les machines est différente.

Pour la résolution de ce problème, nous utilisons la méthode du Recuit Simulé (voir Annexe A). Le résultat final donne l'agencement détaillé des machines dans la cellule, c'est-à-dire les coordonnées exactes des sites, le positionnement des machines sur les sites, ainsi que l'entrée et la sortie de la cellule.

III.4.2 Méthode générale de résolution du problème

Hypothèse fondamentale et critère:

Nous supposons qu'il n'y a qu'un type de moyen de transport dans une cellule. Dans ce cas, le coût c_{ij} associé au moyen de transport entre les machines i et j est égal à un. Le critère à optimiser devient le suivant :

$$\text{Min} \quad \sum_{i=1}^{m} \sum_{j \neq i} f_{ij} \, d_{ij}$$

où :

m : nombre de machines

f_{ij} : trafic entre les machines i et j

d_{ij} : distance entre les machines i et j.

La distance d_{ij} entre les machines peut être la distance euclidienne, la distance de Manhattan ou la distance réelle. Le choix du type de distance dépend du type de configuration de base considéré et du moyen de transport. Toutes les distances possibles pour chaque configuration de base suivant le type du moyen de transport vont être présentées dans le paragraphe III.4.3.

Contraintes et données globales :

Les données globales nécessaires pour chaque configuration de base et qui doivent être fournies par l'utilisateur et les contraintes à respecter sont présentées dans le paragraphe III.2.3 et III.2.6 respectivement.

1. La création de la matrice de flux :

La matrice de flux est une matrice carrée F_{mxm} où m est le nombre de machines de la cellule (voir exemple figure III.18). Elle va d'abord être dimensionnée avec le nombre de machines de la cellule et initialisée à zéro. Ensuite, les différentes valeurs f_{ij} de la matrice vont être calculées à partir des informations issues des gammes des produits. f_{ij} est le flux orienté entre les machines. Il se calcule comme suit :

$$f_{ij} = \sum_{k=1}^{n} \left\lceil \frac{Q[k]_{ij}}{U[k]} \right\rceil \quad i \neq j \ ; \ i,j = 1, \ldots, m$$

$f_{ij} = 0$, si $i = j$

où :

Q[k]ij indique la quantité de produits de chaque type k qui passe de la machine i à la machine j sur l'horizon choisi; m est le nombre de machines de la cellule et n est le nombre de type de produits à fabriquer.

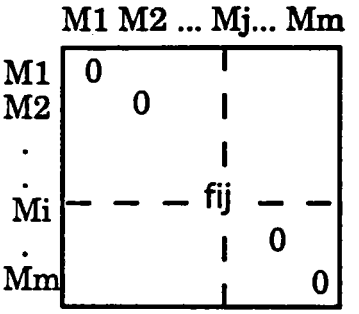


Figure III.18 Matrice de flux

2. Les données relatives aux machines :

i) la longueur des machines de la cellule, *lm[i]* : réel

ii) la largeur des machines de la cellule, *wm[i]* : réel

3. Les données concernant la cellule (si elles sont connues a priori) :

i) la longueur de la cellule, *lc* : réel

ii) la largeur de la cellule, *wc* : réel

La méthode générale de résolution du problème (figure 18.a) :

Après acquisition et traitement des données, le système procède à la construction d'une configuration initiale suivie du calcul de la valeur de critère lui corespondant. L'étape suivante consiste à lancer le processus de Recuit Simulé afin d'obtenir une configuration optimale qui minimise la valeur de la fonction objectif décrite ci-dessus. Le processus du Recuit Simulé se déroule comme suit.

A chaque itération les opérations suivantes sont répétées :

1. Passage de la configuration actuelle à une configuration voisine.

2. Calcul de la valeur de la fonction objectif (appelée coût) pour la nouvelle configuration.

3. Calcul de la différence entre la valeur de la fonction objectif pour la nouvelle configuration et celle de la meilleure configuration trouvée. Si cette

différence est négative, alors la dernière configuration est la meilleure et est automatiquement acceptée. Sinon, on calcule sa probabilité d'acceptation et on utilise la règle d'acceptation de Metropolis (Metropolis et al.,1953) pour la conserver ou la rejeter (voir Annexe A).

A la fin de ce processus, nous obtenons une solution admissible.

Les différentes étapes citées ci-dessus sont décrites dans la suite.

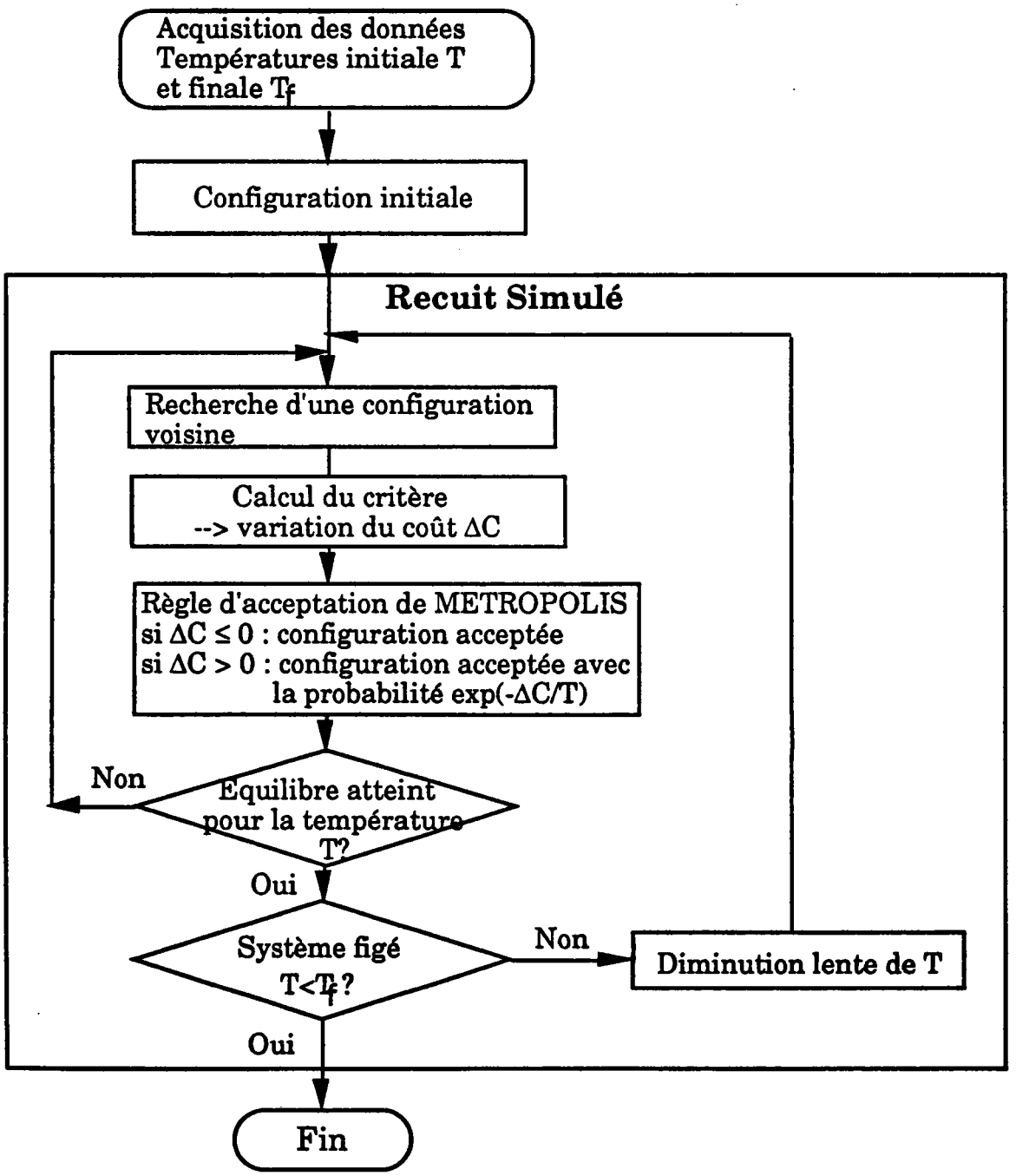


Figure 18.a Méthode générale de résolution du problème

Principales opérations de la méthode générale

1. Configuration initiale

1.1 Affectation des machines sur les sites, en respectant les contraintes de positionnement des machines.

1.1.1 Installation des machines ayant des positions imposées.

1.1.2 Affectation aléatoire ou par l'utilisateur des autres machines.

1.2 Ajustement des positions des machines sur la ou les ligne(s) (en fonction du type de configuration).

1.2.1 Initialisation des positions des machines :

Installation des machines sur des sites équidistants :

Cette équidistance est donnée par :

$$dist_{équi} = \max \{ dmin, \max \{dr[i,j]\} \} \quad i,j = 1, ..., m$$

où : dmin est la distance minimale entre les machines et dr[i,j] la distance relative minimale entre les machines i et j sousmises à une contrainte d'éloignement.

1.2.2 Rapprochement des machines en respectant les contraintes.

Rapprocher les machines de façon à respecter les contraintes de distances relatives minimales entre les machines i et j et la contrainte de non chevauchement entre les machines.

2. Calcul du coût d'une configuration (la valeur de la fonction objectif)

Ce calcul se fait en deux étapes :

2.1 Calcul des distances entre les machines d[i,j].

2.2 Calcul de la valeur de la fonction objectif pour la configuration en question.

3. Réalisation d'une transformation élémentaire

Les transformations élémentaires permettent le passage d'une configuration à une configuration voisine. Elles se font en deux étapes :

3.1. Modification de la séquence des machines sur la ou les ligne(s). Cette modification se fait à l'aide de l'une des modifications suivantes choisie aléatoirement. Le choix d'une transformation élémentaire ainsi que celui de la (ou des) machine(s) à déplacer, aussi de leur(s) nouveau(x) site(s) et du numéro de la ligne en question se fait aléatoirement. Nous présentons ces transformations élémentaires en utilisant l'exemple d'une configuration en ligne. Les transformations sont :

3.1.1 Permutation de deux machines (figure 18.b)

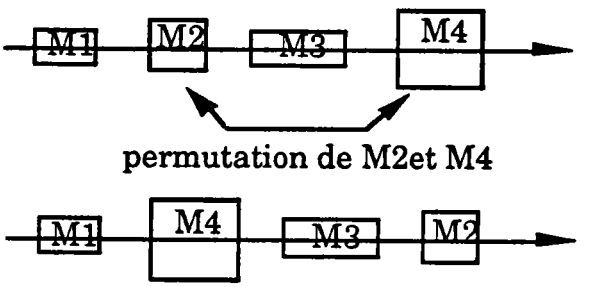


Figure 18.b Permutation de deux machines

3.1.2 Transfert d'une machine vers un site libre (figure 18.c)

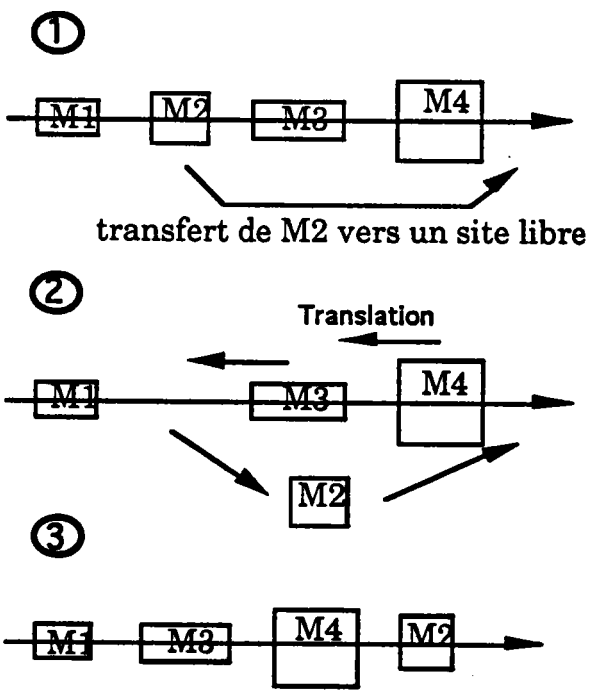


Figure 18.c Transfert d'une machine vers un site libre

3.1.3 Transfert d'une machine vers un site occupé (figure 18.d)

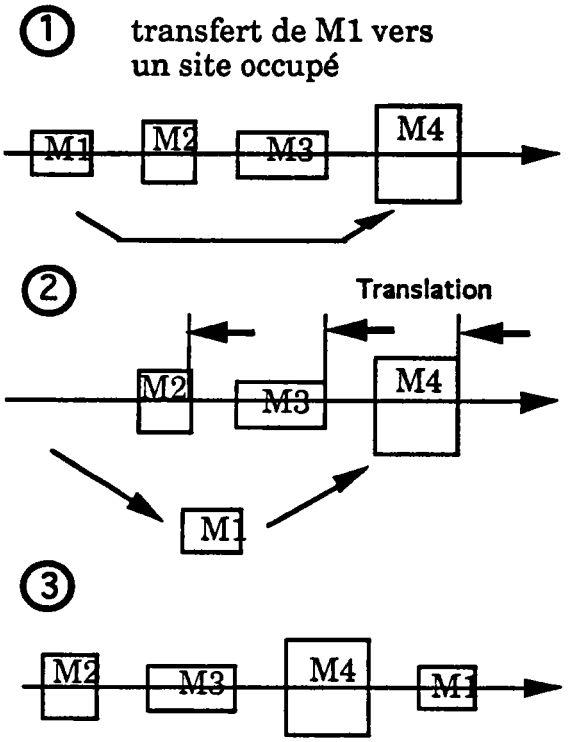


Figure 18.d Transfert d'une machine vers un site occupé

3.2 Ajustement des positions des machines sur la ou les ligne(s).

Cette opération est identique à celle utilisée dans le processus de construction de la configuration initiale (voir section 1.2).

III.4.3 Formulation du problème pour chaque type de configuration de base

1. Configuration en ligne droite (fig. III.19) :

Pour la configuration en ligne droite, la fonction objectif est la suivante :

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times \left| x_i - x_j \right| \qquad (1)$$

Elle est soumise aux contraintes suivantes :

$$| x_i - x_j | \geq (l_i + l_j) / 2 + d\min, \qquad (2)$$

$$x_i \geq 0 \text{ et } x_j \geq 0, \qquad (3)$$

$$i \neq j ; \; i,j = 1, ..., m \qquad (4)$$

où :

x_i est la coordonnée du site de la machine i suivant l'axe de x,

fij= flux de produits entre les machines i et j ;

m = nombre de machines,

dmin est la distance minimale entre les machines i et j,

l_j est la longueur de la machine i.

La fonction objectif (1) a pour but la minimisation de la somme des produits (flux de produits x distance entre les centres des machines), appliqués à tous les couples de machines de la cellule. Elle est soumise aux contraintes de non chevauchement entre les machines (voir (2)), toutes les coordonnées des sites de machines sont positives (3) et deux machines ne partagent pas le même site (4).

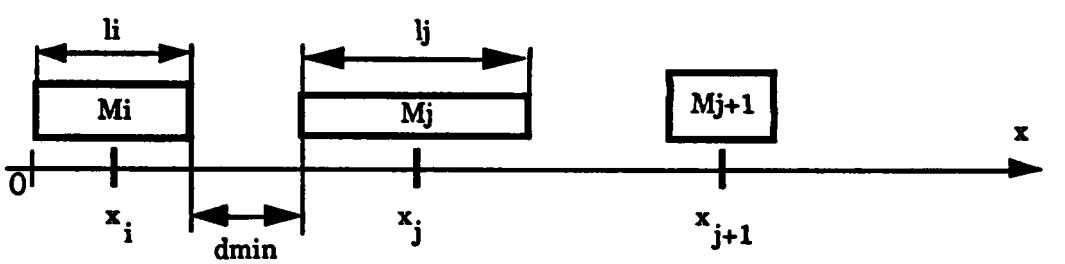


Figure III.19 Configuration en ligne droite

2. Configuration circulaire (fig. III.20) :

Les données suplementaires sont les suivantes :

1. Le rayon R d'action du robot : réel

2. L'angle d'espacement minimal entre les machines i et j : réel (en radian).

La fonction objectif associée à la configuration circulaire est :

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times \left| ß_i - ß_j \right| \qquad (5)$$

Elle est soumise aux contraintes suivantes :

$$| ß_i - ß_j | \geq [\arctan li/(2R) + \arctan lj/(2R)]\pi/180] + \mu_{ij}, \qquad (6)$$

$$ß_i \geq 0 \text{ et } ß_j \geq 0, \qquad (7)$$

$$i \neq j \; ; \; i,j = 1, ..., m \qquad (8)$$

où :

$ß_i$ est l'angle entre le rayon de base et le rayon passant par le site i.

μ_{ij} est l'angle minimal entre les rayons passant par i et j, (tous les angles sont en radians),

R est le rayon du cercle de cette configuration (rayon d'action du robot),

La fonction objectif (5) a pour but la minimisation de la somme des produits (flux de produits x distance entre les centres des machines), appliqués à tous les couples de machines de la cellule. Elle est soumise aux contraintes de non chevauchement entre les machines (6) , toutes les coordonnées des sites de machines sont positives (7) et deux machines ne partagent pas le même site (8).

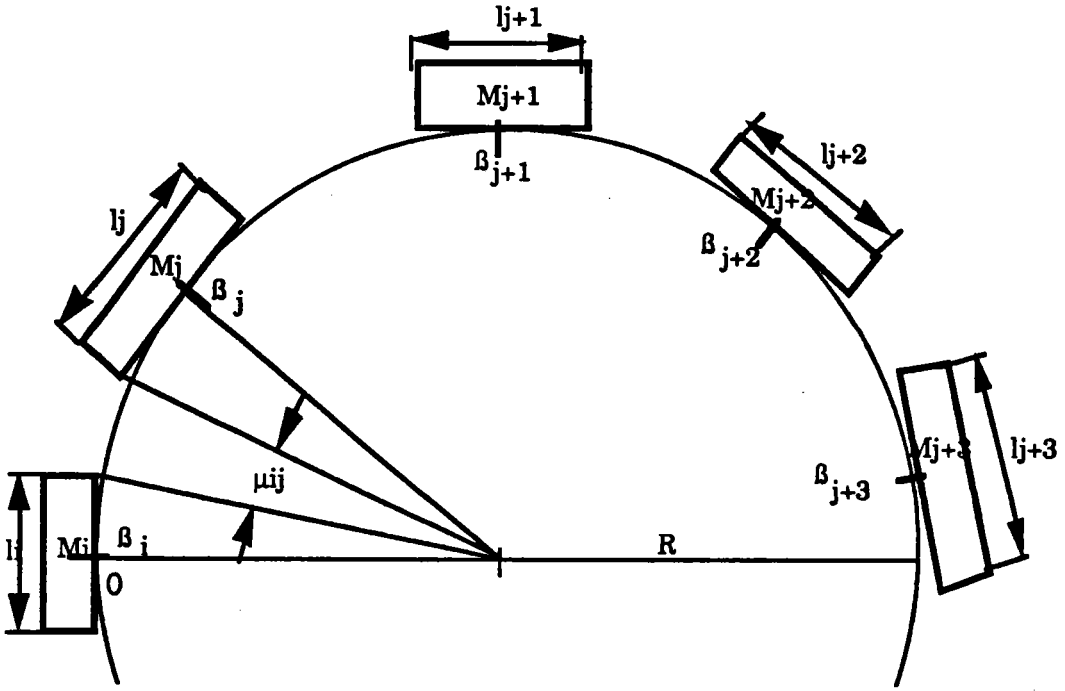


Figure III.20 Configuration circulaire

3. Configuration en ligne double (figure III.21) :

Les données suplémentaires sont les suivants :

1. Le nombre maximal de machines dans une ligne : entier,

2. La largeur du chemin nécessaire pour le moyen de transport : réel,

La fonction objectif est dans ce cas:

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times \left(\left| x_i - x_j \right| + \left| y_i - y_j \right| \right) \qquad (9)$$

Elle est soumise aux contraintes suivantes :

$|x_i - x_j| \geq [(lx_i + lx_j) / 2] + dx_{min}$, pour les machines se trouvant dans une même ligne (10)

$|y_i - y_j| \geq [(ly_i + ly_j) / 2] + dy_{min}$, pour les machines se trouvant dans des lignes différentes (11)

$x_i \geq 0, y_j \geq 0,$ (12)

$i \neq j$; $i,j = 1, ..., m$ (13)

où :

lx_i (resp. ly_i) est la dimension suivant l'axe des x (resp. des y) de la machine i,

dx_{min} est la distance minimale entre deux machines se trouvant sur une même ligne suivant l'axe des x,

dy_{min} est soit la distance minimale entre deux machines se trouvant sur des lignes différentes suivant l'axe des y, soit la largeur du chemin du moyen de transport (dépendant du type du moyen de transport)

La fonction objectif (9) a pour but la minimisation de la somme des produits (flux de produits x distance entre les centres des machines), appliqués à tous les couples de machines de la cellule. Elle est soumise aux contraintes de non chevauchement entre les machines (10) et (11) (où x = 0, y = 0 définit le point d'origine de la ligne double), toutes les coordonnées des sites de machines sont positives (12) et deux machines ne partagent pas le même site (13).

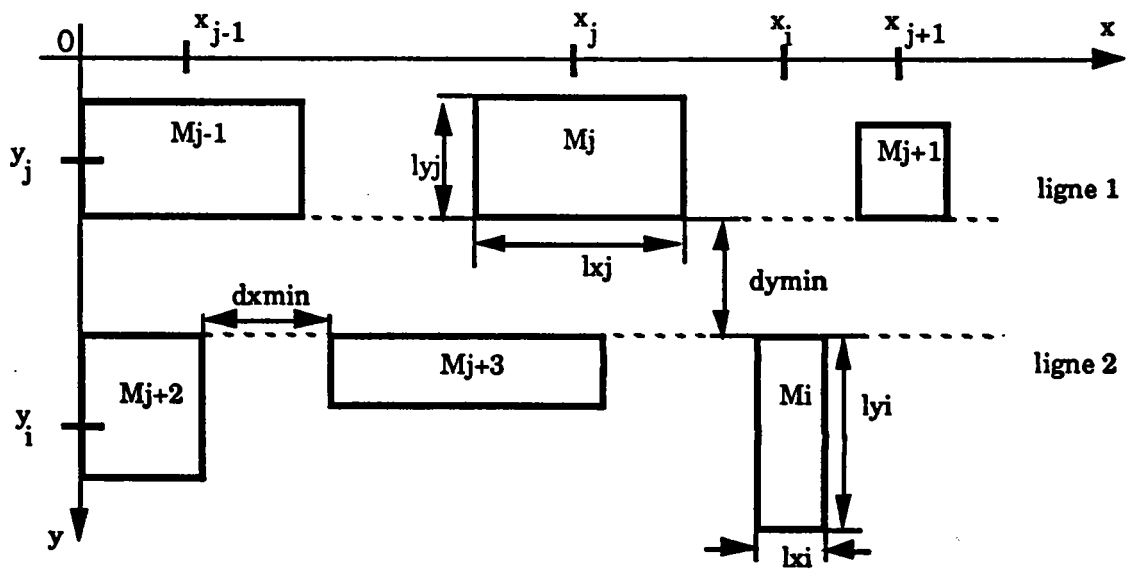


Figure III.22 Configuration en ligne double

4. Configuration multi-lignes :

On peut distinguer trois types de configurations multi-lignes suivant le type du moyen de transport et donc de calcul de la distance entre les machines. Le premier type de configuration multi-lignes est la configuration utilisant un pont roulant comme type du moyen de transport et la distance euclidienne pour calculer la distance entre les machines. Le deuxième type est la configuration multi-lignes utilise des AGVs comme type du moyen de transport, et donc utilise la distance de Manhattan adaptée à une heuristique basée sur la méthode de "branch and bound" pour trouver la distance la plus courte entre les machines. Le troisième type de configuration multi-lignes est la configuration avec convoyeur ou chariot (flux de produits unidirectionnel) comme type du moyen de transport, et donc utilisant la longueur du chemin suivi par le moyen de transport pour calculer la distance entre les machines. Les trois types de configuration multi-lignes sont expliqués plus en détail dans la suite.

a) Configuration multi-lignes avec pont roulant comme type du moyen de transport, et donc utilisant la distance euclidienne pour calculer la distance entre les machines (figure III.22) :

Les données suplementaires sont les suivantes :

1. Le nombre maximal de machines dans une ligne : entier,

2. Le nombre maximal de lignes : entier,

3. La largeur du chemin nécessaire pour le convoyeur ou le chariot : réel,

La fonction objectif est :

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times \left(\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right)^{1/2}$$

Elle est soumise aux contraintes suivantes :

$|x_i - x_j| \geq [(lx_i + lx_j) / 2] + dx_{min}$, pour les machines se trouvant dans une même ligne (14)

$|y_i - y_j| \geq [(ly_i + ly_j) / 2] + dy_{min}$, pour les machines se trouvant dans des lignes différentes (15)

$x_i \geq 0, y_i \geq 0,$ (16)

$i,j = 1, ..., m$ (17)

où :

lx_i (resp. ly_i) est la dimension suivant l'axe des x (resp. des y) de la machine i,

dxmin est la distance minimale entre deux machines se trouvant sur une même ligne suivant l'axe des x,

dymin est la distance minimale entre deux machines se trouvant sur des lignes différentes suivant l'axe des y

La fonction objectif a pour but la minimisation de la somme des produits flux de produits fois distances entre les centres des machines, appliqués à tous les couples de machines de la cellule. Elle est soumise aux contraintes de non chevauchement entre les machines (14) et (15) (où x = 0, y = 0 définit le point d'origine de la configuration multi-lignes), toutes les coordonnées des sites de machines sont positives (16) et deux machines ne partagent pas le même site (17).

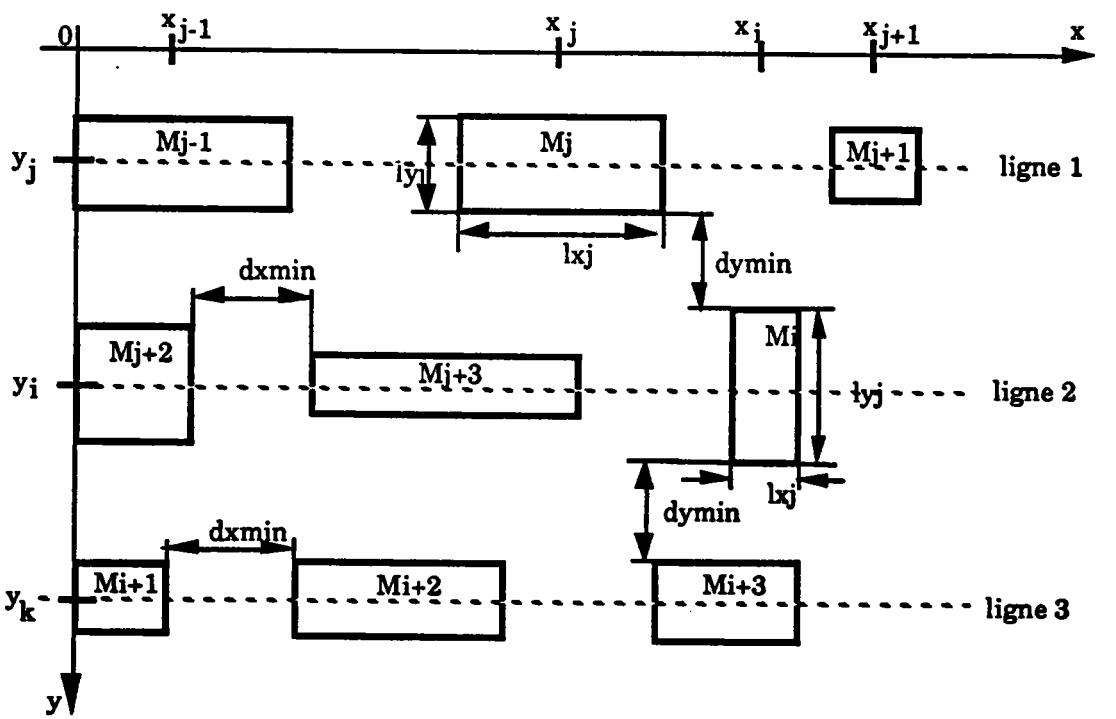


Figure III.22 Configuration multi-lignes

b) Configuration multi-lignes avec AGV comme type du moyen de transport, et donc utilisant la distance de Manhattan adaptée à une heuristique basée sur la méthode de "branch and bound" pour trouver la distance la plus courte entre les machines (figure III.23) :

les données supplémentaires sont les suivantes :

1. nombre maximal de machines dans une ligne : entier,

2. nombre maximal de lignes : entier,

3. largeur du chemin nécessaire pour le AGV : réel,

La fonction objectif est :

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times \left(\left| x_i - x_j \right| + \left| y_i - y_j \right| \right)$$

Elle est soumise aux contraintes suivantes :

$| x_i - x_j | \geq [(lx_i + lx_j) / 2] + d_{MdT}$, pour les machines se trouvant dans une même ligne (18)

$| y_i - y_j | \geq [(ly_i + ly_j) / 2] + d_{MdT}$, pour les machines se trouvant dans des lignes différentes (19)

$x_i \geq 0, y_i \geq 0,$ (20)

$i \neq j$; $i,j = 1, ..., m$ (21)

où :

lx_i (resp. ly_i) est la dimension suivant l'axe des x (resp. des y) de la machine i.

d_{MdT} est la distance minimale entre les machines pour que l'AGV puisse toujours passer entre elles.

La fonction objectif a pour but la minimisation de la somme des produits (flux de produits x distance entre les centres des machines), appliqués à tous les couples de machines de la cellule. Elle est soumise aux contraintes de non chevauchement entre les machines (18) et (19) (où x = 0, y = 0 définit le point

d'origine de la configuration multi-lignes), toutes les coordonnées des sites de machines sont positives (20) et deux machines ne partagent pas le même site (21).

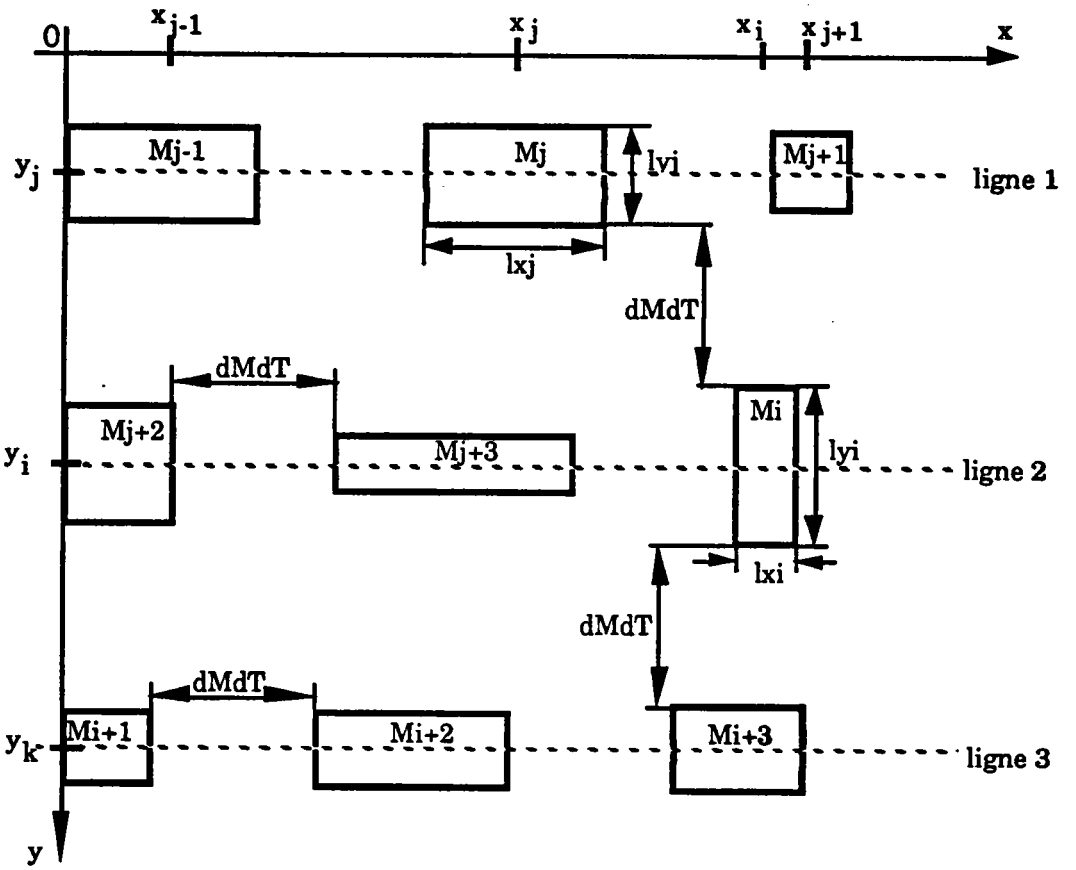


Figure III.23 Configuration multi-lignes avec AGV comme type du moyen de transport

c) Configuration multi-lignes avec convoyeur ou chariot (flux de produits unidirectionnel) comme type du moyen de transport, et donc utilisant la longueur du chemin réel suivi par le moyen de transport pour calculer la distance entre les machines (figure III.23) :

Les données supplémentaires sont les suivantes :

1. nombre maximal de machines dans une ligne : entier,

2. nombre maximal de lignes : entier,

3. largeur du chemin nécessaire pour le convoyeur ou le chariot : réel,

La fonction objectif est :

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times d_{ij}$$

La distance dij entre les machines se calcule comme suit (voir aussi figure III.23) :

$$d_{ij} = \min \{ \Delta y + (x_{S(i)} + lx_i/2) + (x_{S(j)} + lx_j/2) \, , \, \Delta y + 2\, lc - (x_{S(i)} + lx_i/2) - (x_{S(j)} + lx_j/2) + \Delta y_{chemin} \}$$

où : lc est la longueur de la ligne le plus longue suivant l'axe des x,

$$\Delta y = | ki - kj | \, \Delta y_{lignes}$$

ki est le numéro du chemin emprunté par le moyen de transport où se trouve machine i.

Δy_{lignes} est la distance suivant l'axe des y entre les chemins empruntés par le moyen de transport dans les différentes lignes suivant l'axe des x (voir figure III.23). Elle est donnée par :

$$\Delta y_{lignes} = \Delta y_{chemin} + dy_{min} + 2\, l_{max}$$

où : lmax = la largeur de la machine la plus large,

lx_i (resp. lx_j) est la dimension suivant l'axe des x (resp. des y) de la machine i,

dx_{min} est la distance minimale entre deux machines de la même ligne suivant l'axe des x,

dy_{min} est la distance minimale entre deux machine de lignes différentes suivant l'axe des y

Δy_{chemin} est la distance minimale entre deux machines séparées par le chemin du moyen de transport suivant l'axe des y. Elle est donnée par :

Δy_{chemin} = max {dy_{min}, largeur du chemin nécessaire pour le convoyeur ou le chariot }

La fonction objectif est soumise aux contraintes suivantes :

$|x_i - x_j| \geq [(lx_i + lx_j) / 2] + dxmin$, pour les machines se trouvant dans une même ligne (22)

$|y_i - y_j| \geq [(ly_i + ly_j) / 2] + dymin$, pour les machines se trouvant dans des lignes différentes (23)

$x_i \geq 0, y_i \geq 0,$ (24)

$i \neq j ; i,j = 1, ..., m$ (25)

La fonction objectif a pour but la minimisation de la somme des produits (flux de produits x distance entre les centres des machines), appliqués à tous les couples de machines de la cellule. Elle est soumise aux contraintes de non chevauchement entre les machines (22) et (23) (où x = 0, y = 0 définit le point d'origine de la configuration multi-lignes), toutes les coordonnées des sites de machines sont positives (24) et deux machines ne partagent pas le même site (25).

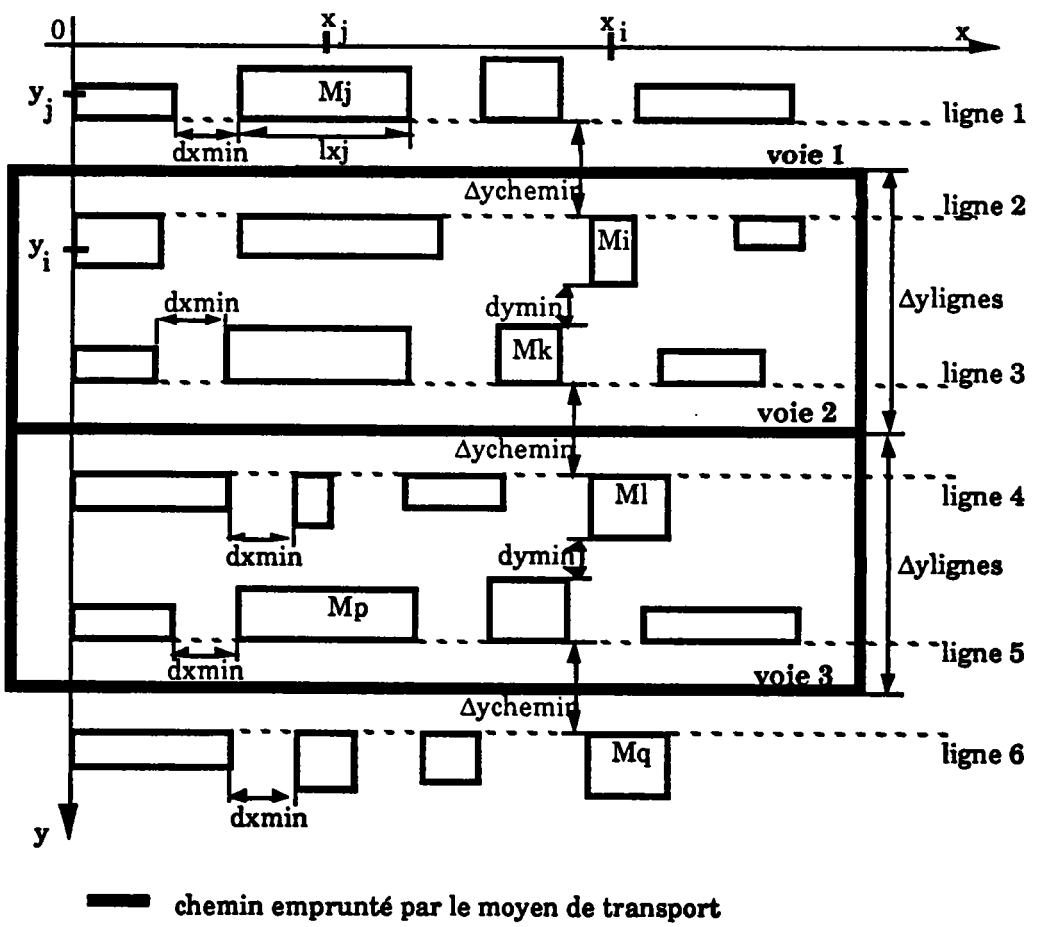


Figure III.23 Configuration multi-lignes avec convoyeur ou chariot comme type du moyen de transport

5. Configuration quelconque :

La configuration quelconque est une configuration fournie par l'utilisateur et consistant en un nombre donné de sites fixes dont on connait les coordonnées (le nombre de sites doit être égal ou supérieur au nombre de machines de la cellule à placer). Dans ce cas, la distance euclidienne est utilisée pour calculer la distance entre les machines (figure III.24).

La fonction objectif est :

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times \left(\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right)^{1/2}$$

Elle est soumise aux contraintes suivantes :

non chevauchement entre les machines

$x_i \geq 0, y_i \geq 0,$ (28)

$i \neq j$; $i,j = 1, ..., m$ (29)

où :

lx_i (resp. ly_i) est la dimension suivant l'axe des x (resp. des y) de la machine i et dmin est la distance minimale entre les machines,

La fonction objectif a pour but la minimisation de la somme des produits (flux de produits x distance entre les centres des machines), appliqués à tous les couples de machines de la cellule. Elle est soumise aux les contraintes de non chevauchement entre les machines, toutes les coordonnées des sites de machines sont positives (28) et deux machines ne partagent pas le même site (29).

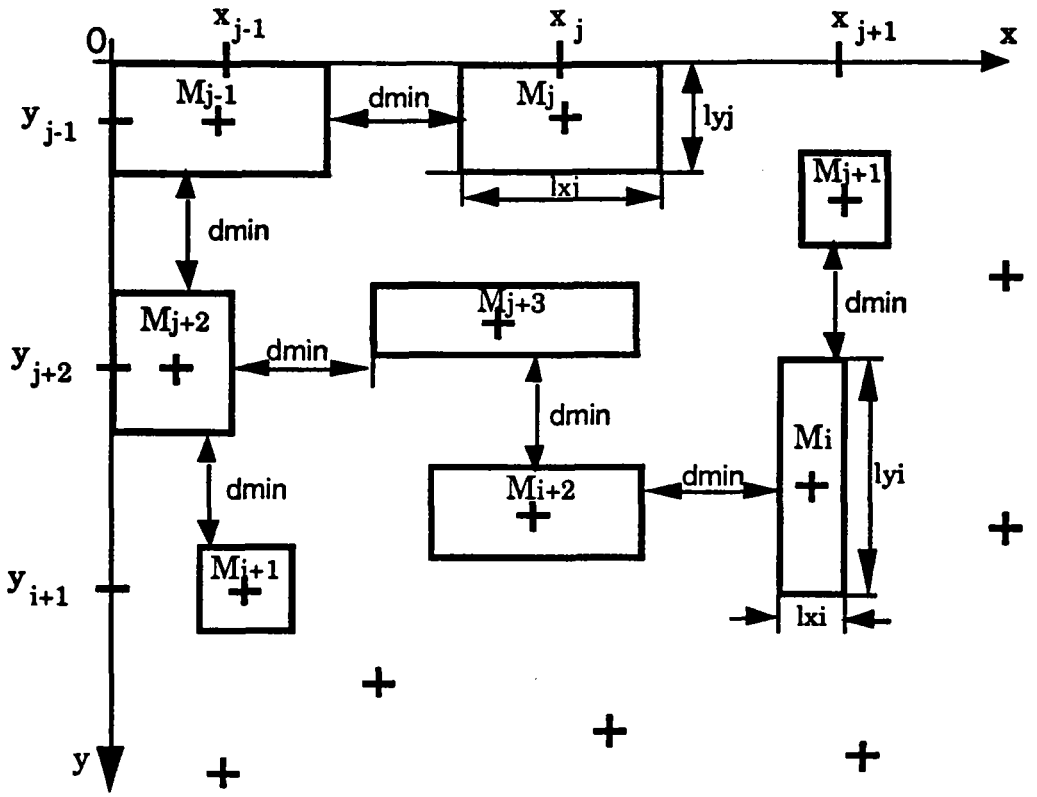


Figure III.24 Configuration quelconque

III.4.4 Discussions

1. Positionnement de l'entrée et de la sortie de la cellule

Le positionnement de l'entrée et de la sortie peut se faire de deux manière différentes :

La première consistes à considérer l'entrée et la sortie de la cellule comme des machines de chargement et de déchargement de produits dans la cellule. Dans ce cas, on ajoute à la matrice de flux, les flux entre l'entrée et les machines et ceux entre les machines et la sortie. Cette matrice est calculée en ajoutant au début des routages des produits la machine de chargement (l'entrée de la cellule), et à la fin de leurs routages la machine de déchargement (la sortie de la cellule). Nous déterminons ainsi simultanément les positions des machines et celles de l'entrée et de la sortie.

La deuxième manière permet d'aboutir à la même solution en deux étapes successives :

1. Dans la première étape, nous cherchons la configuration optimale des machines, en ne tenant compte que des flux entre les machines et en ignorant

les flux de produits entre l'entrée de la cellule et les machines et les flux entre les machines et la sortie.

2. Dans la deuxième étape nous déterminons les positions de l'entrée E et de la sortie S de la cellule. Cela est réalisé de la manière suivante :

Nous allons trouver les positions de l'entrée et de la sortie sur le pourtour de la cellule qui minimisent le critère suivant :

$$\text{Min} \sum_{i=1}^{m} \left[f(E,i) \times d(E,i) + f(i,S) \times d(i,S) \right]$$

où :

f(E,i) est le flux entre l'entrée E et la machine i.

f(i,S) est le flux entre machine i et la sortie S.

d(E,i) est la distance parcourue par le moyen de transport entre l'entrée de la cellule et la machine i.

d(i,S) est la distance parcourue par le moyen de transport entre la machine i et la sortie de la cellule.

L'ensemble des positions que peuvent prendre ces deux points, l'entrée et la sortie, est défini en fonction du type de configuration et du moyen de transport utilisé.

Exemples :

1. Cas d'une configuration en ligne droite et d'un convoyeur ou d'un chariot:

L'ensemble des positions possibles de l'entrée et de la sortie contient seulement deux points. Le points de début de ligne et le point de fin de ligne.

2. Cas d'une configuration en ligne droite et d'un AGV :

L'ensemble des positions possibles de l'entrée et de la sortie est formé de tous les points de la ligne.

3. Cas d'une configuration en ligne double et d'un AGV :

L'ensemble des positions possibles de l'entrée et de la sortie contient seulement deux points. Le point de début de ligne et le point de fin de ligne.

4. Cas d'une configuration multi-ligne et d'un pont roulant :

L'ensemble des positions possibles de l'entrée et de la sortie est formé des points situés sur le pourtour de la cellule.

2. Avantages et inconvénients de l'approche du Recuit Simulé

L'approche du Recuit Simulé est une approche qui cherche à éviter les minima locaux d'un problème d'optimisation combinatoire. C'est une approche extrêmement rapide qui permet de traiter des problèmes de grande taille par rapport aux méthodes exactes. Un autre avantage de la méthode du Recuit Simulé est la possibilité d'obtenir plusieurs solutions différentes sous différentes conditions de lancement, c'est-à-dire à partir de solutions admissibles initiales différentes.

Le plus grand désavandage de l'approche est le fait de ne pas garantir l'obtention de la solution optimale. Le principal inconvénient est la difficulté pour déterminer les paramètres de contrôle convenables pour chaque problème différent traité.

III.4.5 Résultats

L'approche de résolution du problème d'agencement des machines que nous avons présentée dans le paragraphe III.4 est une approche qui est adaptée à chaque cas de configuration de base considérée avec le moyen de transport associé. Aussi, une configuration quelconque introduite par l'utilisateur peut être prise en compte. Un avantage majeur de cette approche est le fait de prendre en compte les distances réelles pendant tous les calculs. Le résultat obtenu est la configuration finale qui donne l'agencement détaillé des machines sur les sites de la cellule, c'est-à-dire les coordonnées exactes des sites, le positionnement des machines sur les sites, ainsi que l'entrée et la sortie de la cellule.

IV. PRESENTATION DES RESULTATS

INTRODUCTION

Dans ce chapitre, nous allons montrer des résultats numériques obtenus pour deux exemples de cellules de production pour lesquels on applique les différentes configurations de base en utilisant les moyens de transport possibles.

EXEMPLE 1 :

1.1 Détermination d'une configuration en ligne droite et AGV, convoyeur, chariot, ou pont roulant comme moyen de transport

1.2 Détermination d'une configuration circulaire et robot comme moyen de transport

```
  EXEMPLE1  AGENCEMENT-INTRA-CELLULAIRE
*****************************************

Nombre_de_types_de_produit_a_fabriquer_: 5
Nombre_de_machines_dans_la_cellule_: 4

-----------------------------------------

Produit_1_:
Quantite_a_produire_: 1000
Taille_du_lot_: 10
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 1

Indice_de_la_machine_2_du_routage_: 3

Produit_2_:
Quantite_a_produire_: 500
Taille_du_lot_: 5
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 4

Indice_de_la_machine_2_du_routage_: 2

Indice_de_la_machine_3_du_routage_: 4

Indice_de_la_machine_4_du_routage_: 1

Produit_3_:
Quantite_a_produire_: 1500
Taille_du_lot_: 10
Nombre_de_machines_du_routage_: 3

Indice_de_la_machine_1_du_routage_: 3

Indice_de_la_machine_2_du_routage_: 1

Indice_de_la_machine_3_du_routage_: 4

Produit_4_:
Quantite_a_produire_: 2000
Taille_du_lot_: 20
Nombre_de_machines_du_routage_: 3

Indice_de_la_machine_1_du_routage_: 2

Indice_de_la_machine_2_du_routage_: 3

Indice_de_la_machine_3_du_routage_4_: 1

Produit_5_:
Quantite_a_produire_: 100
Taille_du_lot_: 1
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 4

Indice_de_la_machine_2_du_routage_: 1
```



```
*****************************************

MATRICE DES FLUX INTER-MACHINES
-------------------------------

          M1          M2          M3          M4
M1    0.000000    0.000000  100.000000  150.000000
M2    0.000000    0.000000  100.000000  100.000000
M3  250.000000    0.000000    0.000000    0.000000
M4  200.000000  100.000000    0.000000    0.000000
```

MACHINES				PRODUITS					
				numéro	1	2	3	4	5
				Quantité	1000	500	1500	2000	100
Indice Mach.	Long. [m]	Larg. [m]	Poids [t]	Taille lot	10	5	10	20	1
1	5	2	4.5	Gamme	1	4	2	3	2
2	1	1	0.5			2		1	
3	2.5	2	1.2		2		1	2	
4	2.5	3	1.9			1, 3	3		1

Tableau IV.1 Exemple 1 : 5 produits à fabriquer
4 machines utilisées

1.1 Détermination d'une configuration en ligne droite et AGV, convoyeur, chariot, ou pont roulant comme moyen de transport

EXEMPLE 1 : CONFIGURATION EN LIGNE DROITE SANS CONTRAINTES
****

La distance minimale entre les machines est : dist_min = 1.5

LA CONFIGURATION INITIALE

La sequence des machines et les coordonnees de leurs sites sont :

[4 <0.00>] [2 <4.00>] [3 <6.50>] [1 <10.50>]

LA CONFIGURATION FINALE

Le nombre total d'iterations est : 505

Le temps de traitement est : 53 sec. CPU

Le cout de la configuration initiale est : 3775.000000

Le cout de la configuration finale est : 2550.000000

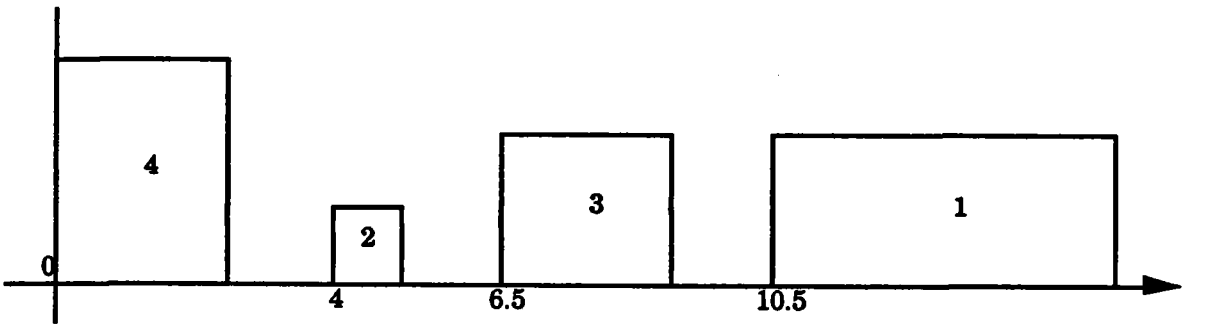
Le taux d'amelioration est de : 32.45 pour-cent

La sequence des machines et les coordonnees de leurs sites sont :

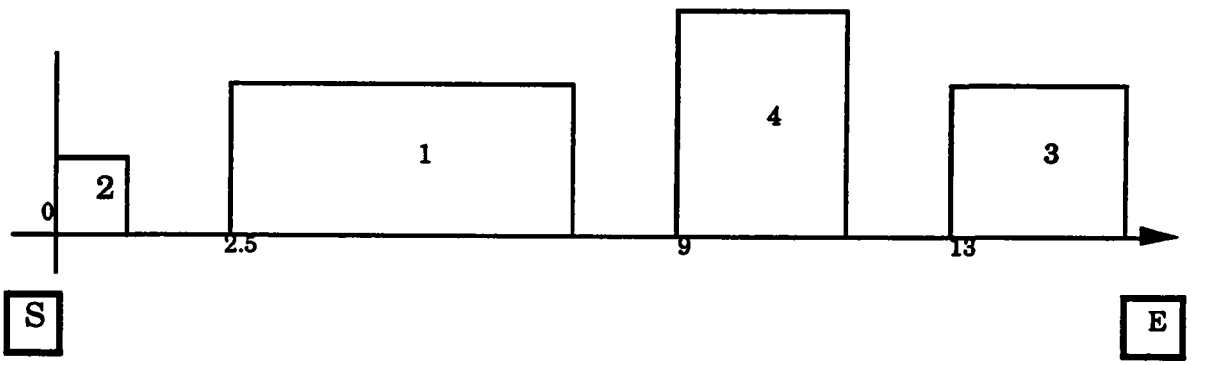
[2 <0.00>] [1 <2.50>] [4 <9.00>] [3 <13.00>]

L'entree se trouve a droite et la sortie a gauche de la ligne.

CONFIGURATION EN LIGNE DROITE



Configuration en ligne droite sans contrainte :
solution initiale
1.5 unités entre toutes les machines



E : Entrée

S : Sortie

Configuration en ligne droite sans contrainte :
solution finale
1.5 unités entre toutes les machines

1.2 Détermination d'une configuration circulaire et robot comme moyen de transport

EXEMPLE 1 : CONFIGURATION CIRCULAIRE SANS CONTRAINTES
****

Le rayon du bras de manipulation du robot : rayon = 5
La distance minimale entre les machines est : dist_min = 1.5

LA CONFIGURATION INITIALE

La sequence des machines et leur angle de la configuration initiale sont :

[2 <0.00>] [3 <28.66>] [4 <74.52>] [1 <120.38>]

RESULTAT

Le nombre total d'iterations est : 505

Le temps de traitement est : 22 sec. CPU

Le cout de la configuration initiale est : 3150.000000

Le cout de la configuration finale est : 999.999817

Le taux d'amelioration est de : 68.25 pour-cent

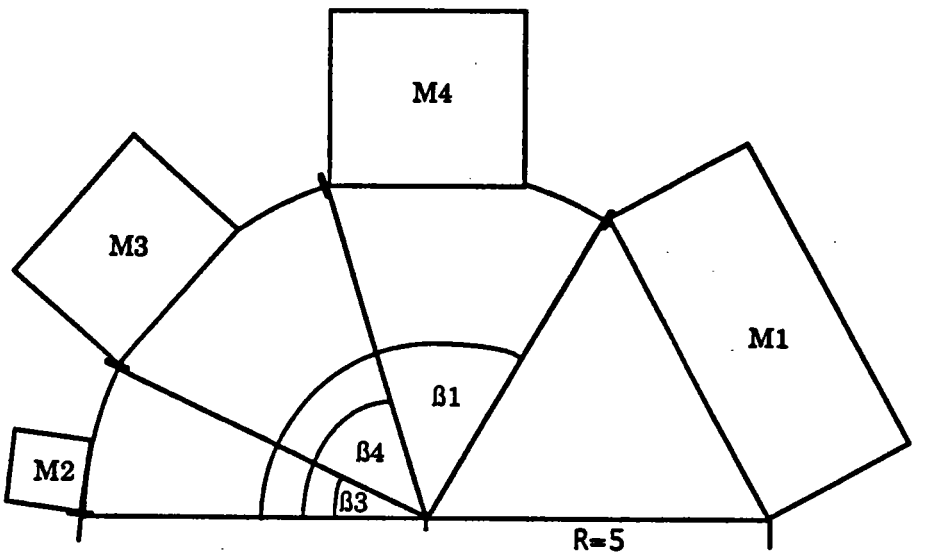
LA CONFIGURATION FINALE

La sequence des machines et leur angle de la configuration finale sont :

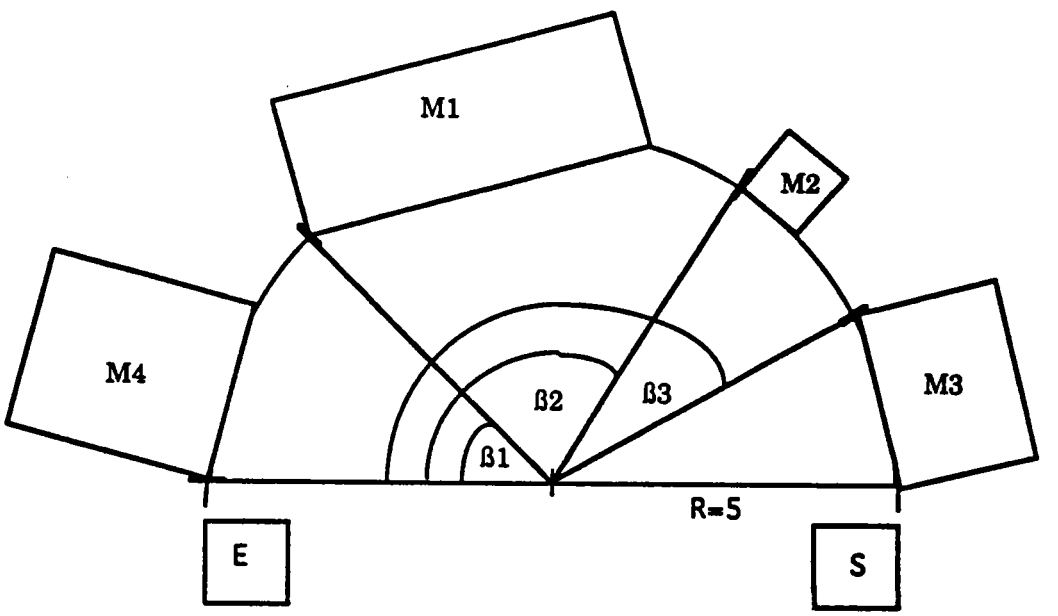
4 < 0.00> 1 < 45.86> 2 < 120.38> 3 < 149.04>

L'entree est a gauche et la sortie a droite de la configuration.

CONFIGURATION CIRCULAIRE



Configuration circulaire initiale sans contrainte :



Configuration circulaire finale sans contrainte :

EXEMPLE 2 :

2.1 Détermination d'une configuration en ligne double et AGV, convoyeur, chariot, ou pont roulant comme moyen de transport

2.1.1 Sans contraintes supplémentaires

2.1.2 Avec contraintes supplémentaires

2.2 Détermination d'une configuration multi-lignes

2.2.1 Sans contraintes supplémentaires et AGV comme moyen de transport

2.2.2 Avec contraintes supplémentaires et convoyeur ou chariot comme moyen de transport

2.2.3 Sans contraintes supplémentaires et pont roulant comme moyen de transport

EXEMPLE2 AGENCEMENT-INTRA-CELLULAIRE
****

Nombre_de_types_de_produit_a_fabriquer_: 17
Nombre_de_machines_dans_la_cellule_: 8

Produit_1_:
Quantite_a_produire_: 8
Taille_du_lot_: 1
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 1

Indice_de_la_machine_2_du_routage_: 3

Produit_2_:
Quantite_a_produire_: 10
Taille_du_lot_: 1
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 4

Indice_de_la_machine_2_du_routage_: 8

Indice_de_la_machine_3_du_routage_: 7

Indice_de_la_machine_4_du_routage_: 8

Produit_3_:
Quantite_a_produire_: 20
Taille_du_lot_: 4
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 5

Indice_de_la_machine_2_du_routage_: 4

Produit_4_:
Quantite_a_produire_: 40
Taille_du_lot_: 20
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 2

Indice_de_la_machine_2_du_routage_: 1

Indice_de_la_machine_3_du_routage_: 8

Indice_de_la_machine_4_du_routage_: 1

Produit_5_:
Quantite_a_produire_: 10
Taille_du_lot_: 5
Nombre_de_machines_du_routage_: 5

Indice_de_la_machine_1_du_routage_: 2

Indice_de_la_machine_2_du_routage_: 6

Indice_de_la_machine_3_du_routage_: 4

Indice_de_la_machine_4_du_routage_: 7

Indice_de_la_machine_5_du_routage_: 2

Produit_6_:
Quantite_a_produire_: 4
Taille_du_lot_: 2
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 7

Indice_de_la_machine_2_du_routage_: 4

Indice_de_la_machine_3_du_routage_: 6

Indice_de_la_machine_4_du_routage_: 2

Produit_7_:
Quantite_a_produire_: 30
Taille_du_lot_: 30
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 1

Indice_de_la_machine_2_du_routage_: 4

Indice_de_la_machine_3_du_routage_: 5

Indice_de_la_machine_4_du_routage_: 1

Produit_8_:
Quantite_a_produire_: 100
Taille_du_lot_: 10
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 8

Indice_de_la_machine_2_du_routage_: 4

Produit_9_:
Quantite_a_produire_: 10
Taille_du_lot_: 5
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 1

Indice_de_la_machine_2_du_routage_: 2

Indice_de_la_machine_3_du_routage_: 5

Indice_de_la_machine_4_du_routage_: 2

Produit_10_:
Quantite_a_produire_: 20
Taille_du_lot_: 5
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 4

Indice_de_la_machine_2_du_routage_: 5

Indice_de_la_machine_3_du_routage_: 6

Indice_de_la_machine_4_du_routage_: 5

Produit_11_:
Quantite_a_produire_: 12
Taille_du_lot_: 4
Nombre_de_machines_du_routage_: 4

Indice_de_la_machine_1_du_routage_: 3

Indice_de_la_machine_2_du_routage_: 2

Indice_de_la_machine_3_du_routage_: 3

Indice_de_la_machine_4_du_routage_: 1

Produit_12_:
Quantite_a_produire_: 10
Taille_du_lot_: 10
Nombre_de_machines_du_routage_: 6

Indice_de_la_machine_1_du_routage_: 1

Indice_de_la_machine_2_du_routage_: 5

Indice_de_la_machine_3_du_routage_: 6

Indice_de_la_machine_4_du_routage_: 7

Indice_de_la_machine_5_du_routage_: 6

Indice_de_la_machine_6_du_routage_: 8

Produit_13_:
Quantite_a_produire_: 2
Taille_du_lot_: 1
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 2

Indice_de_la_machine_2_du_routage_: 7

Produit_14_:
Quantite_a_produire_: 30
Taille_du_lot_: 6
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 3

Indice_de_la_machine_2_du_routage_: 1

Produit_15_:
Quantite_a_produire_: 5
Taille_du_lot_: 5
Nombre_de_machines_du_routage_: 3

Indice_de_la_machine_1_du_routage_: 8

Indice_de_la_machine_2_du_routage_: 6

Indice_de_la_machine_3_du_routage_: 5

Produit_16_:
Quantite_a_produire_: 50

Taille_du_lot_: 10
Nombre_de_machines_du_routage_: 3

Indice_de_la_machine_1_du_routage_: 6

Indice_de_la_machine_2_du_routage_: 5

Indice_de_la_machine_3_du_routage_: 6

Produit_17_:
Quantite_a_produire_: 5
Taille_du_lot_: 5
Nombre_de_machines_du_routage_: 2

Indice_de_la_machine_1_du_routage_: 4

Indice_de_la_machine_2_du_routage_: 1

****

MATRICE DES FLUX INTER-MACHINES

	M1	M2	M3	M4	M5	M6	M7	M8
M1	0.000000	2.000000	8.000000	1.000000	1.000000	0.000000	0.000000	2.000000
M2	2.000000	0.000000	3.000000	0.000000	2.000000	2.000000	2.000000	0.000000
M3	8.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
M4	1.000000	0.000000	0.000000	0.000000	5.000000	2.000000	2.000000	10.000000
M5	1.000000	2.000000	0.000000	5.000000	0.000000	10.000000	0.000000	0.000000
M6	0.000000	2.000000	0.000000	2.000000	10.000000	0.000000	1.000000	1.000000
M7	0.000000	2.000000	0.000000	2.000000	0.000000	1.000000	0.000000	10.000000
M8	2.000000	0.000000	0.000000	10.000000	0.000000	1.000000	10.000000	0.000000

					PRODUITS														
				numéro	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
MACHINES				Quantité	8	10	20	40	10	4	30	100	10	20	12	10	2	30	5	50	5
Indice Mach	Long. [dm]	Larg. [dm]	Poids [x 50 kg]	Taille lot	1	1	4	20	5	2	30	10	5	5	4	10	1	6	5	10	5
1	20	20	40	Gamme	1			2,4			1,4		1		4	1		2			2
2	10	10	10					1	1,5	4			2,4		2		1				
3	15	15	23		2										1,3			1			
4	10	10	10			1	2		3	2	2	2		1							1
5	15	15	30				1				3		3	2,4		2			3	2	
6	15	15	38						2	3				3		3,5			2	1,3	
7	10	10	10			3			4	1						4	2				
8	10	10	15			2,4		3				1				6			1		

Tableau IV.2 Exemple 2 : 17 produits à fabriquer
8 machines utilisées

2.1 Détermination d'une configuration en ligne double et AGV, convoyeur, chariot, ou pont roulant comme moyen de transport

2.1.1 Sans contraintes supplémentaires

EXEMPLE 2 : CONFIGURATION EN LIGNE DOUBLE SANS CONTRAINTES
****

La distance entre les machines sur deux lignes adjacentes est : dist_lignes = 10
La distance entre les machines d'une meme ligne est : dist_machs = 5

LA CONFIGURATION INITIALE

La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [1 <0.00>] [6 <25.00>] [8 <45.00>] [3 <60.00>] [7 <80.00>]

Ligne 2 : [4 <0.00>] [2 <15.00>] [5 <30.00>]

LA CONFIGURATION FINALE

Le nombre total d'iterations est : 185

Le temps de traitement est : 4 sec. CPU

Le cout de la configuration initiale est : 5725.000000

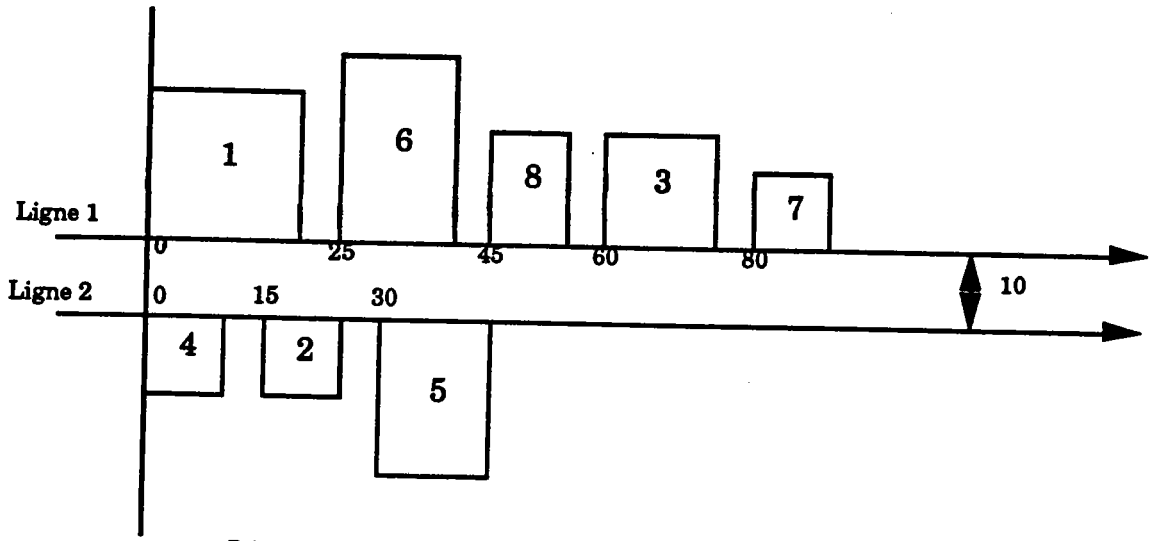
Le cout de la configuration finale est : 3605.000000

Le taux d'amelioration est de : 37.03 pour-cent

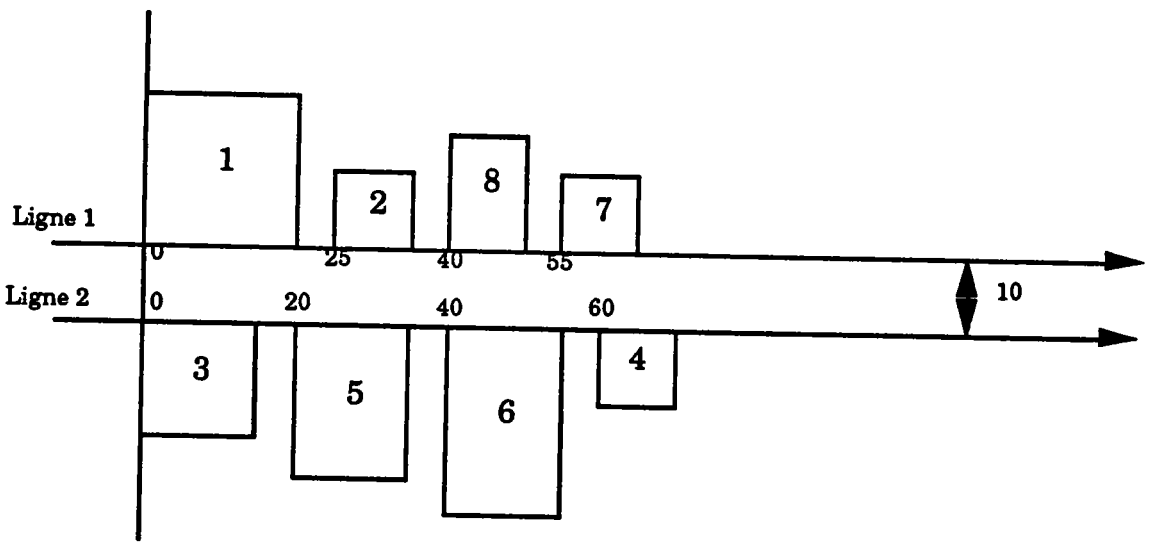
La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [1 <0.00>] [2 <25.00>] [8 <40.00>] [7 <55.00>]

Ligne 2 : [3 <0.00>] [5 <20.00>] [6 <40.00>] [4 <60.00>]



Ligne double sans contrainte solution initiale



Ligne double sans contrainte solution finale

2.1 Détermination d'une configuration en ligne double et AGV, convoyeur, chariot, ou pont roulant comme moyen de transport

2.1.2 Avec contraintes supplémentaires

EXEMPLE 2 : CONFIGURATION EN LIGNE DOUBLE AVEC CONTRAINTES
****

Les contraintes de rapprochement sont :

Rapprochement entre les machines 1 et 4 : d_max = 30
Rapprochement entre les machines 5 et 7 : d_max = 40

Les contraintes d'eloingement sont :

Eloignement entre les machines 3 et 7 : d_min = 40
Eloignement entre les machines 2 et 8 : d_min = 50

La contrainte de fixation d'une machine sur un site est :

Machine 6 doit etre fixee sur le site 2 de la ligne 2

La distance entre les machines sur deux lignes adjacentes est : dist_lignes = 10
La distance entre les machines d'une meme ligne est : dist_machs = 5

LA CONFIGURATION INITIALE

La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [2 <0.00>] [3 <15.00>] [5 <35.00>] [1 <55.00>] [8 <80.00>]

Ligne 2 : [7 <0.00>] [6 <15.00>] [4 <35.00>]

LA CONFIGURATION FINALE

Le nombre total d'iterations est : 178

Le temps de traitement est : 2 sec. CPU

Le cout de la configuration initiale est : 6837.770996

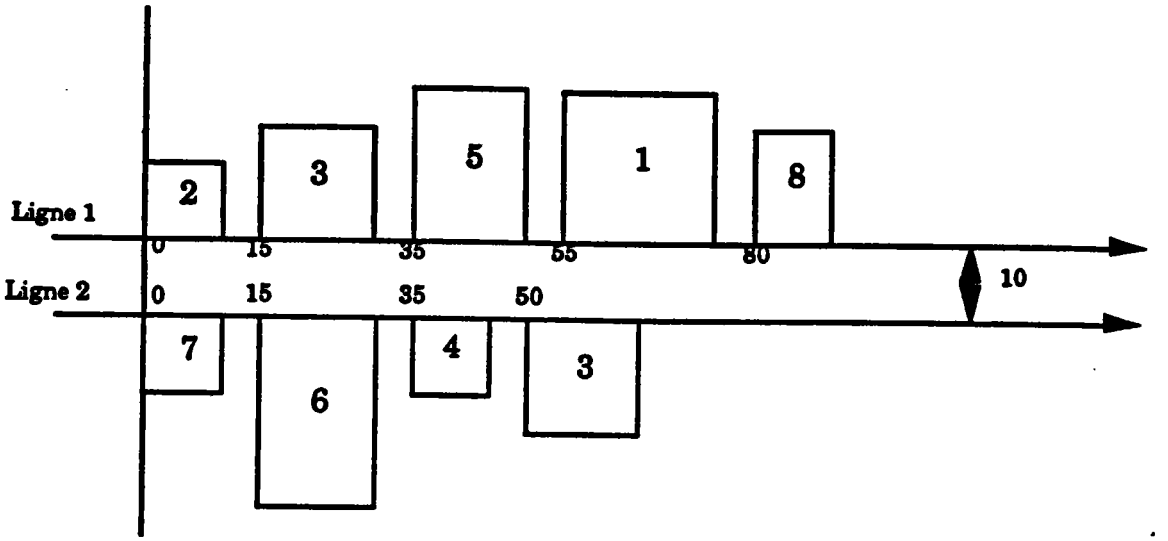
Le cout de la configuration finale est : 4435.000000

Le taux d'amelioration est de : 35.14 pour-cent

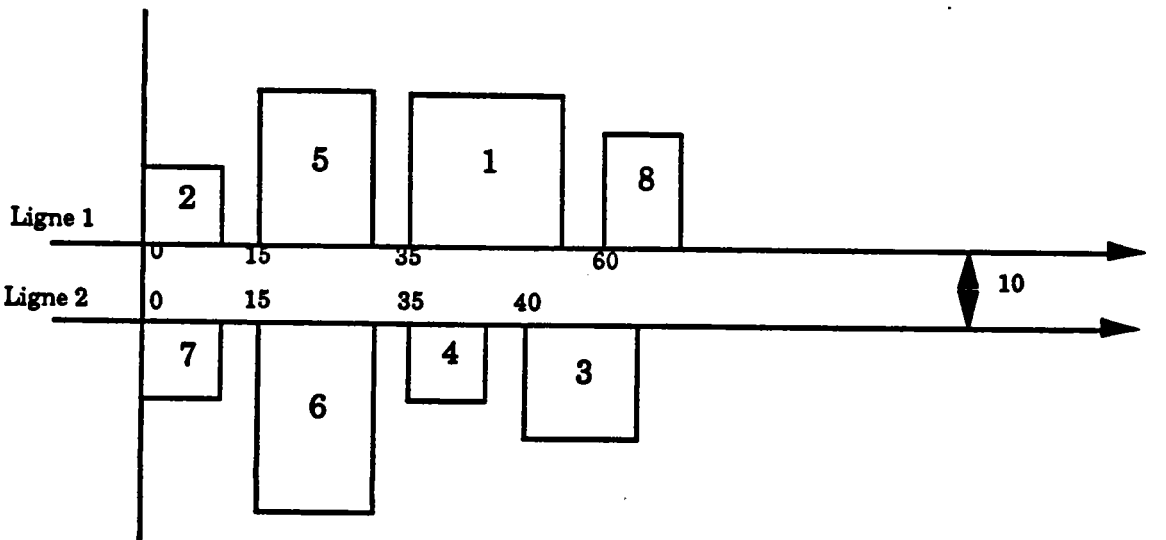
La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [2 <0.00>] [5 <15.00>] [1 <35.00>] [8 <60.00>]

Ligne 2 : [7 <0.00>] [6 <15.00>] [4 <35.00>] [3 <50.00>]



Configuration en ligne double avec contraintes d'éloignement et de rapprochement : solution initiale



Configuration en ligne double avec contraintes d'éloignement et de rapprochement : solution finale

2.2 Détermination d'une configuration multi-lignes

2.2.1 Sans contraintes supplémentaires et AGV comme moyen de transport

EXEMPLE 2 : CONFIGURATION EN MULTI-LIGNE SANS CONTRAINTES ET
AGV COMME MOYEN DE TRANSPORT
****

La distance entre les machines sur deux lignes adjacentes est : dist_lignes = 10
La distance entre les machines d'une meme ligne est : dist_machs = 10
Le nombre maximal de machines par ligne est : no_mach_ligne = 3
Le nombre maximal de lignes est : no_lignes = 3

LA CONFIGURATION INITIALE

La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [2 <0.00>] [6 <15.00>]

Ligne 2 : [3 <0.00>] [8 <20.00>] [1 <35.00>]

Ligne 3 : [7 <0.00>] [4 <15.00>] [5 <30.00>]

LA CONFIGURATION FINALE

Le nombre total d'iterations est : 184

Le temps de traitement est : 5 sec. CPU

Le cout de la configuration initiale est : 5415.000000

Le cout de la configuration finale est : 4155.000000

Le taux d'amelioration est de : 23.27 pour-cent

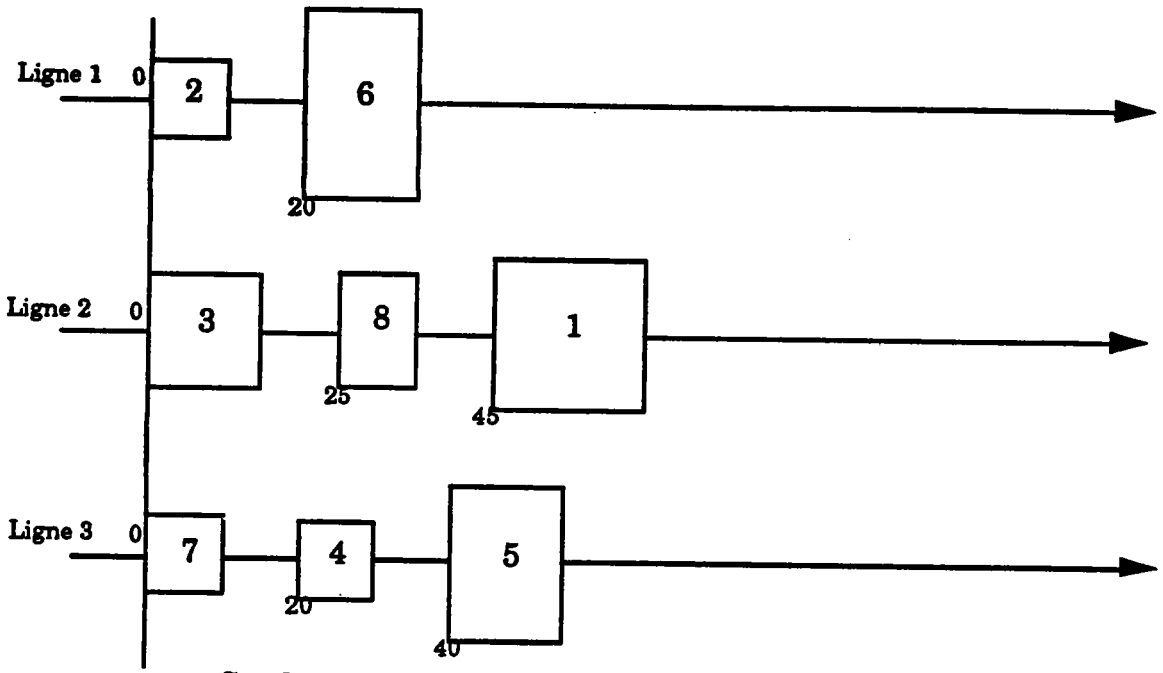
La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [3 <0.00>] [1 <20.00>]

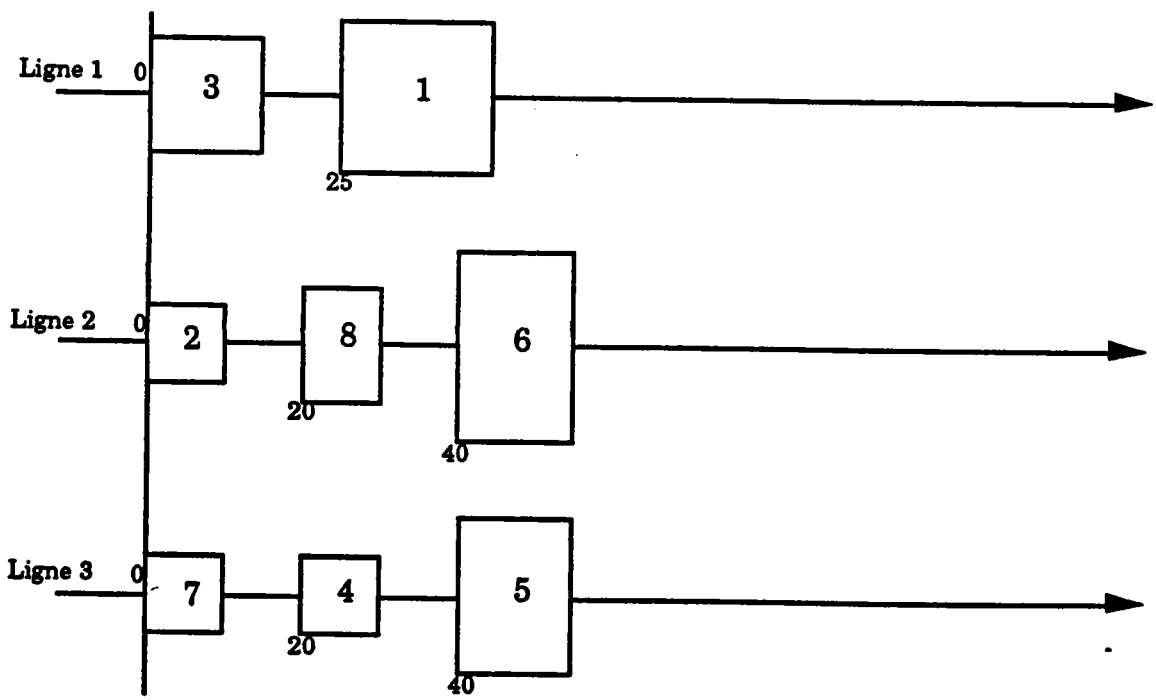
Ligne 2 : [2 <0.00>] [8 <15.00>] [6 <30.00>]

Ligne 3 : [7 <0.00>] [4 <15.00>] [5 <30.00>]

AGV



Configuration en multi-lignes avec contrainte :
solution initiale
10 unités entre toutes les machines



Configuration en multi-lignes avec contrainte :
solution finale
10 unités entre toutes les machines

2.2 Détermination d'une configuration multi-lignes

2.2.2 Avec contraintes supplémentaires et convoyeur ou chariot comme moyen de transport

EXEMPLE 2 : CONFIGURATION EN MULTI-LIGNES AVEC CONTRAINTES ET
CONVOYEUR OU CHARIOT COMME MOYEN DE TRANSPORT
****

La distance entre les machines sur deux lignes adjacentes separees par le chemin du moyen de transport est : dist_lignes_chemin = 15
La distance entre les machines sur deux lignes adjacentes non separees par le chemin du moyen de transport est : dist_lignes = 5
La distance entre les machines d'une meme ligne est : dist_machs = 5
Le nombre maximal de machines par ligne est : no_mach_ligne = 3
Le nombre maximal de lignes est : no_lignes = 3

Les contraintes de rapprochement sont :

Rapprochement entre les machines 1 et 4 : d_max = 30
Rapprochement entre les machines 5 et 7 : d_max = 40

Les contraintes d'eloignement sont :

Eloignement entre les machines 3 et 7 : d_min = 40
Eloignement entre les machines 2 et 8 : d_min = 50

La contrainte de fixation d'une machine sur un site est :

Machine 6 doit etre fixee sur le site 2 de la ligne 2

LA CONFIGURATION INITIALE

La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [4 <0.00>] [1 <15.00>] [8 <40.00>]

Ligne 2 : [3 <0.00>] [6 <20.00>] [2 <40.00>]

Ligne 3 : [7 <0.00>] [5 <15.00>]

LA CONFIGURATION FINALE

Le nombre total d'iterations est : 514

Le temps de traitement est : 15 sec. CPU

Le cout de la configuration initiale est : 6355.000000

Le cout de la configuration finale est : 4695.000000

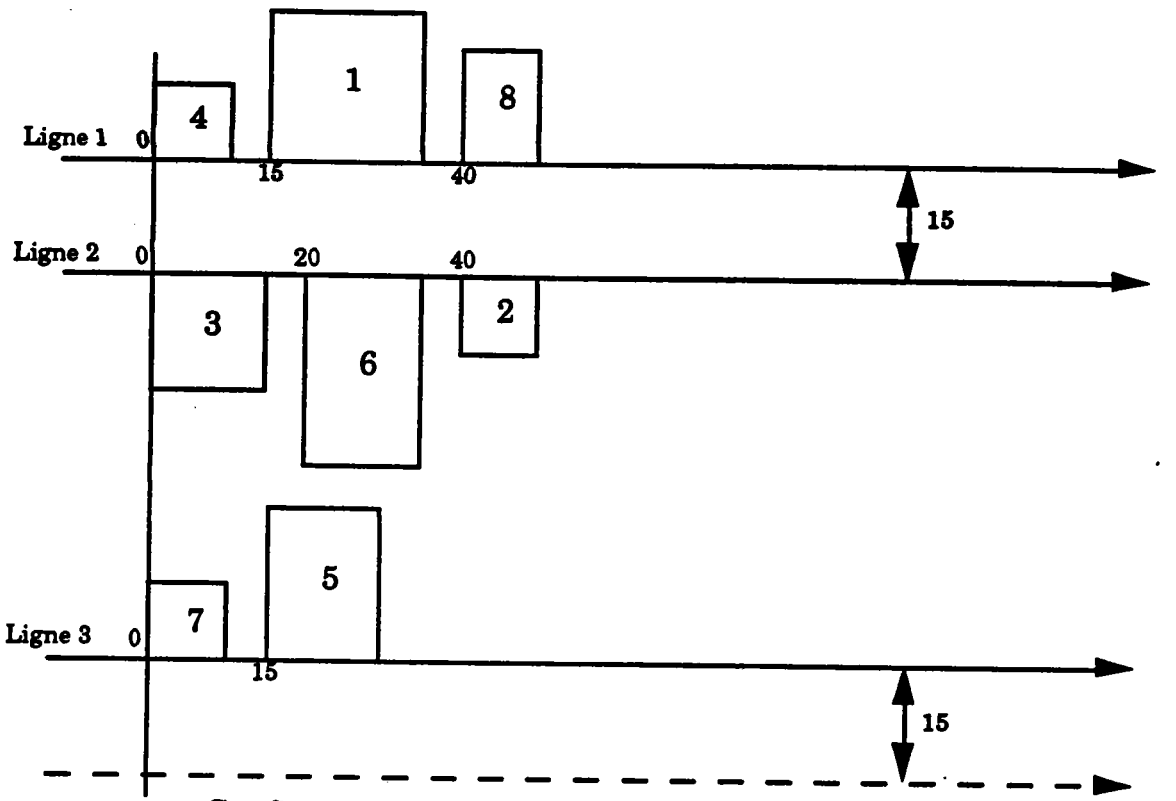
Le taux d'amelioration est de : 26.12 pour-cent

La sequence des machines et les coordonnees de leurs sites sont :

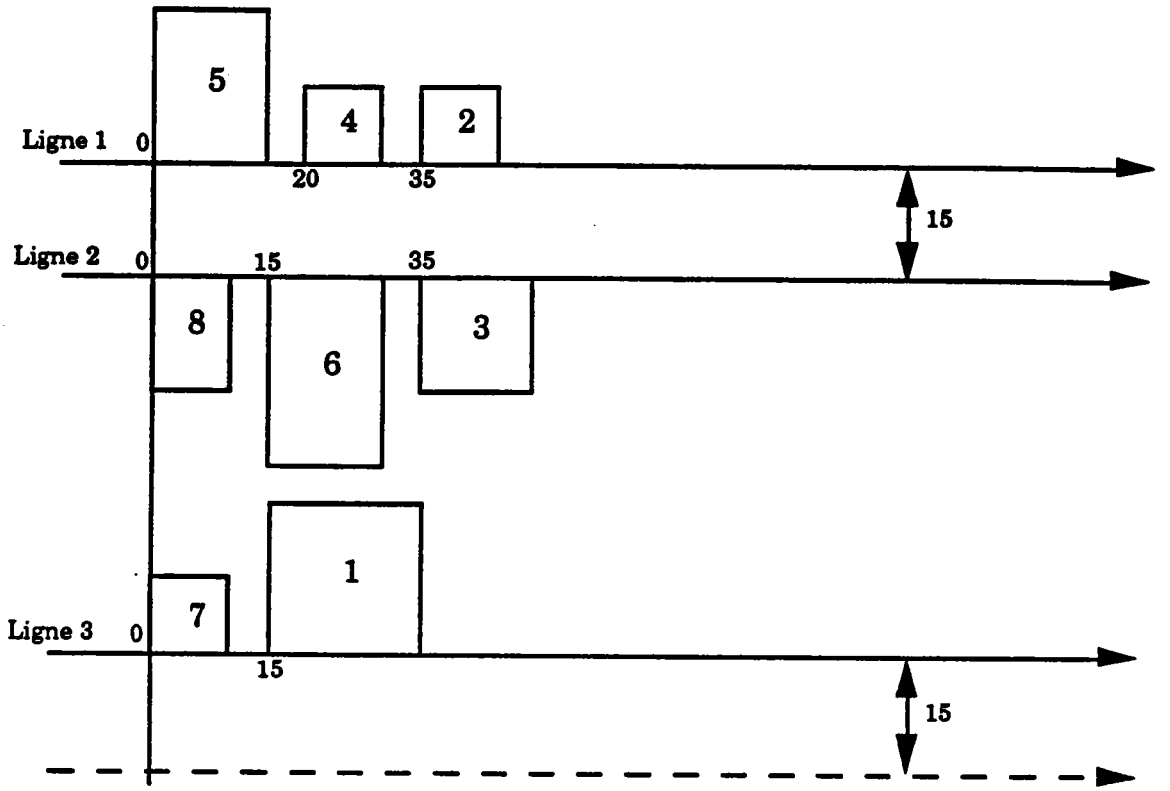
Ligne 1 : [5 <0.00>] [4 <20.00>] [2 <35.00>]

Ligne 2 : [8 <0.00>] [6 <15.00>] [3 <35.00>]

Ligne 3 : [7 <0.00>] [1 <15.00>]



Configuration en multi-lignes avec contraintes :
solution initiale
5 unités entre toutes les machines
Passages de 15 unités entre les lignes



Configuration en multi-lignes avec contraintes :
solution finale
5 unités entre toutes les machines
Passages de 15 unités entre les lignes

2.2 Détermination d'une configuration multi-lignes

2.2.3 Sans contraintes supplémentaires et pont roulant comme moyen de transport

EXEMPLE 2 : CONFIGURATION EN MULTI-LIGNE SANS CONTRAINTES ET
PONT ROULANT COMME MOYEN DE TRANSPORT
****

La distance entre les machines sur deux lignes adjacentes est : dist_lignes = 5
La distance entre les machines d'une meme ligne est : dist_machs = 5
Le nombre maximal de machines par ligne est : no_mach_ligne = 3
Le nombre maximal de lignes est : no_lignes = 3

LA CONFIGURATION INITIALE

La sequence des machines et les coordonnees de leurs sites sont :

Ligne 1 : [2 <0.00>] [6 <15.00>]

Ligne 2 : [3 <0.00>] [8 <20.00>] [1 <35.00>]

Ligne 3 : [7 <0.00>] [4 <15.00>] [5 <30.00>]

LA CONFIGURATION FINALE

Le nombre total d'iterations est : 505

Le temps de traitement est : 22

Le cout de la configuration initiale est : 5490.000000

Le cout de la configuration finale est : 4855.000000

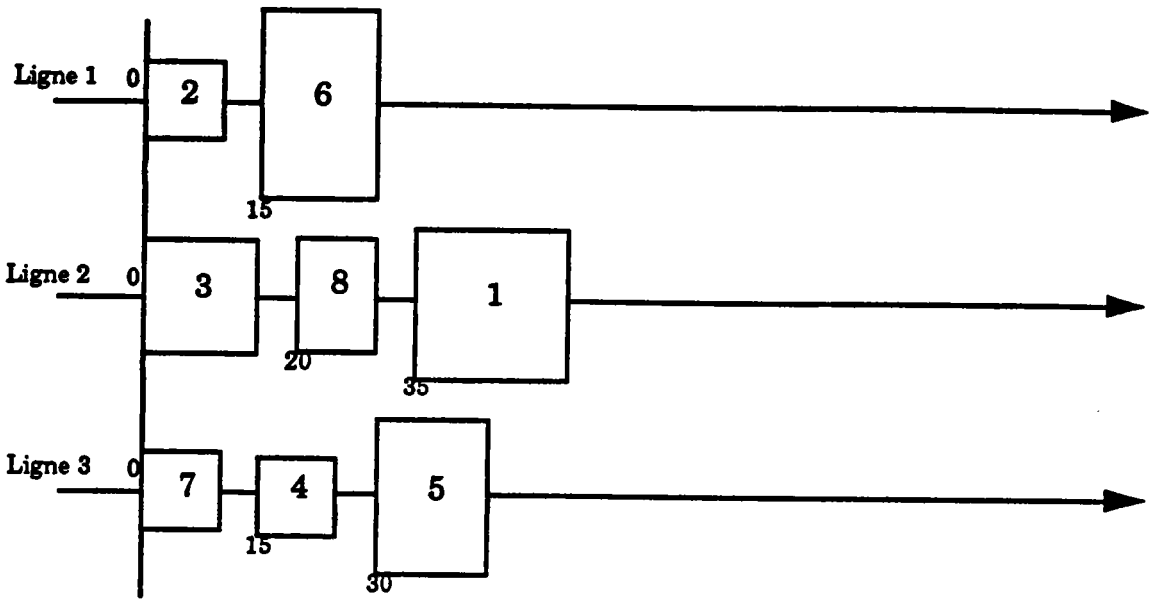
Le taux d'amelioration est de : 11.57 pour-cent

La sequence des machines et les coordonnees de leurs sites sont :

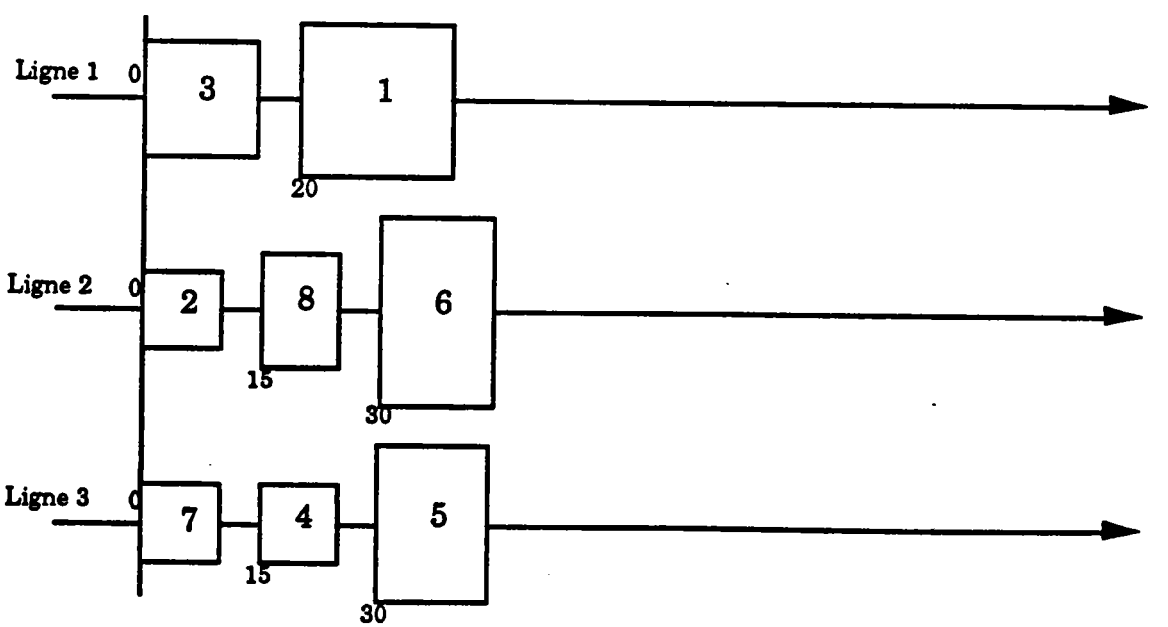
Ligne 1 : [3 <0.00>] [1 <20.00>]

Ligne 2 : [2 <0.00>] [8 <15.00>] [6 <30.00>]

Ligne 3 : [7 <0.00>] [4 <15.00>] [5 <30.00>]



Configuration en multi-lignes avec contrainte :
solution initiale
5 unités entre toutes les machines



Configuration en multi-lignes avec contrainte :
solution finale
5 unités entre toutes les machines

CONCLUSION GENERALE

CONCLUSION GENERALE

Dans cette thèse, nous nous sommes intéressés à une approche globale en vue de résoudre le problème d'agencement des machines à l'intérieur des cellules des systèmes de fabrication. Dans une première étape, nous avons mis au point des systèmes experts pour sélectionner le moyen de transport et la configuration de base de la cellule. Ensuite, pour résoudre la partie combinatoire du problème, nous avons utilisé la méthode de Recuit Simulé pour réaliser l'affectation des machines à des sites sur la surface de la cellule de manière à minimiser un critère basé sur le trafic entre les machines et la distance de transport entre les machines.

Cette approche prend en compte les types de moyens de transport les plus fréquemment utilisés dans les cellules de fabrication. Elle s'applique à tous les types de configurations de base les plus employés pour les cellules (ligne droite, configuration circulaire ou en fer à cheval, double ligne et multi-lignes). Une première originalité du travail a été de prendre en compte les configurations quelconques (composées d'un nombre quelconque de sites) qui peuvent éventuellement être imposées par les utilisateurs pour des besoins fortement contraints. Une deuxième originalité a été de prendre en compte les distances les plus précises possibles des parcourts effectifs des produits entre les machines dans le calcul du critère d'optimisation.

Bien que le problème de placement des machines sur des sites à l'intérieur de la cellule se formule différemment suivant le type de moyen de transport utilisé et la configuration choisie, l'utilisation de la méthode du Recuit Simulé comme moyen unique de résolution de ce problème combinatoire a un certain nombre d'avantages comme il a été discuté en fin de chapitre III. Le plus important est sans conteste le fait que l'utilisateur peut relancer plusieurs fois l'algorithme pour chercher à améliorer la solution. Ceci lui donne une plus grande confiance dans l'utilisation d'un tel outil informatique. L'inconvénient majeur est bien sûr qu'on ne garantit pas que la solution trouvée soit optimale, mais elle est, si les paramètres du Recuit Simulé ont été fixés correctement, proche de l'optimum.

Les approches développées dans la littérature sont souvent partielles ou trop générales par rapport à notre méthode. Aussi, il est difficile de les

comparer avec notre travail. Cependant, lorsqu'une comparaison est possible, les résultats obtenus dans ce travail sont souvent plus performants.

Nous pensons que ce travail peut trouver des débouchés dans l'industrie sous forme de produit logiciel pour l'agencement des ressources. Cependant, bien que de nombreux facteurs techniques aient déjà été pris en compte, la réalisation d'un tel produit implique l'élargissement de la base de connaissances des deux systèmes experts et l'inclusion d'informations plus détaillées. En particulier, d'autres types de systèmes de transport doivent être pris en compte, peut être en fonction des secteurs industriels d'application, comme par exemple des balancelles pour l'industrie automobile. De plus, pour chaque type de moyens de transport, la base de connaissances doit être enrichie avec les données relatives aux systèmes de transport commerciaux les plus courants.

Pour la suite des travaux, on peut penser à prendre en compte le problème du positionnement des produits sur les machines en plus du positionnement devant les machines ce qui a une influence sur le choix du moyen de transport. En effet, certaines machines requièrent du moyen de transport la possibilité d'introduire les produits dans ou sur la machine. Cette possibilité est fournie bien sûr par un robot articulé, mais il existe aussi des automatismes qui peuvent être adaptés aux moyens de transport pour réaliser ces fonctions (mouvements de translation ou de rotation).

Enfin, une approche plus réaliste devrait aussi tenir compte de la forme exacte des machines. Cela implique une discrétisation fine de la surface de la cellule et des machines (par exemple, par un maillage). Le problème deviendrait alors fortement combinatoire et beaucoup plus coûteux à résoudre en terme de temps de calcul.

En conclusion, nous pensons avoir démontré la faisabilité de notre approche et avoir développé un prototype suffisamment réaliste qui, sans fournir une solution parfaite du problème, conduit à des résultats directement exploitables et suffisamment précis pour l'implantation d'une cellule de fabrication dans un site industriel.

VI. REFERENCES

VI. REFERENCES

Abdon G., Dutta S.P. (1989), An integrated approach to facilities layout using expert systems, *Int. J. Prod. Res.*, Vol.28, No. 4, pp. 687-708.

Adiga A., Gadre M. (1990), Object-oriented software modeling of a flexible manufacturing system, *J. Intelligent and Robotic Sys. 3*, pp. 147-165.

Anciaux D. (1990), Etude d'Agencement Spatial dans un Atelier de Production selon le Concept de Technologie de Groupe, *Thèse de doctorat*, Université de Metz, France.

Bazaraa M.S., Kirca O. (1983), A branch-and- bound-based heuristic for solving the QAP, *Naval Research Logistics Quarterly* 30, pp. 287-304.

Bazaraa M.S., Sherali M.D. (1980), Benders' partitioning scheme applied to a new formulation of the quadratic assignement problem, *Naval Research Logistics Quarterly*, 27 (1), pp. 29-41.

Begert U. (1988), Integrated transport and storage systems in FMS, Digitron AG, Switzerland, *Proc. 7th Int. Conf. on Manuf. Sys.*, pp. 99-119.

Beziat Ph. (1990), Conception d'un système d'implantation d'atelier de production : PLOOT, *Thèse de doctorat, Université de Montpellier.*

Block T.E. (1978), FATE : A new construction algorithm for facilities layout, *Journal of Engineering Production*, 2, pp. 111-120.

Brandt H.-P. (1989), Beitrag zur rechnergestützten Layoutplanung von Industriebetrieben, *Dissertation an der Universität Dortmund, Abteilung Maschinenbau*, Hauptberichter: Prof. Dr.-Ing. R. Jünemann, Dortmund, Germany.

Browne J., Chan W.W., Rathmill K. (1985), An integrated FMS design procedure, *Annals of Operations Research 3*, pp. 207-237.

Buffa E.S., Armour G.C., Vollmann T.E. (1964), Allocating facilities with CRAFT, *Harvard Buisness Review* 42, pp. 136-158.

Burkard R.E., Bonninger T. (1983), A heuristic for quadratic Boolean program with applications to quadratic assignement problems, *Eurepean Journal of Operational Research*, No. 13, pp. 374-386.

Chaudhry S.S., Kincaid R. (1990), Comments on "Facilitiy location problem on network under multiple criteria fuzzy set theoric approach" *Int. J. Systems Sci.*, Vol.21, No.11, pp. 2387-2391.

Dangelmaier, W. (1985), *Algorithmen und Verfahren zur Erstellung innerbetrieblicher Anordnungspläne*, Habilitationsschrift genehmigt von der Universität Stuttgart, Fakultät Fertigungstechnik, Hauptberichter: Prof. Dr.-Ing. H.-J. Warnecke, Stuttgart, Germany.

Deisenroth M.P., Apple J.M. (1972), A computerized plant layout analysis and evaluation technique, *Annual AIIE Conference*, Norcross, GA.

Drezner Z. (1980), "DISCON : A new method for the layout problem", *Operations Research* 25(6), pp. 1375-1384.

Edwards H.K., Gillet B.E., Hale M.C. (1970), Modular allocation technique (MAT), *Management Science*,17(3), pp. 161-169.

Elsayed E.A., Kao T.Y. (1990), Machine assignments in production systems with manufacturing cells, *Int. J. Prod. Res.*, Vol. 28, No. 3, pp. 489-501.

Elshafei A.N. (1977), Hospital layout as a quadratic assignment problem, *Operations Research Quarterly* 28(1), pp. 167-179.

El-Rayad T.E., Hollier H.P. (1970), A review of plant design technics, *Int. J. Prod. Res.*, Vol.8, No.3, pp. 263-278.

Eaves B.C., Zangwill W.I. (1971), Generalized cutting plane algorithms, *SIAM J. Control 9*, pp. 529-542.

Expert Systems Strategies, (1990), Design++: An Expert System for automating the engineering design process, Vol.6, No.6.

Finke G., Burkard R.E., Rendl F. (1985), Quadratic assignement problems, working paper, *Dept. of Applied Mathematics*, Technical University of Nova Scotia, Halifax, NS, Canada.

Fisher E.L. (1986), An AI-Based Methodology for Factory Design : FADES, *AI MAGAZINE*, Vol. 7, No. 4.

Gavett J.W., Plyter N.V. (1966), The optimal assignment of facilities to locations by branch and bound, *Operations Research* ,14, pp. 210-232.

Gilmore P.C. (1962), Optimal and suboptimal algorithms for the quadratic assignment problem, Journal of the Society for Industrial and Applied Mathematics 10, pp. 305-313.

Gracia H., Proth J.M. (1985), Group technology in production management : the short horizon planning level, *Applied Stochastic Models and Data Analysis*, 1, pp. 25-34.

Grant I.W. (1952), Factoy planning and plant layout, *Pretice-Hall, Inc. Englewood Cliffs*, New Jersey.

Gray N.A.B. (1990), Capturing knowledge through Top-Down Induction of decision trees, Tools and Techniques, *University of Wollongong*, Australia.

Groover M.P., Zimmers E.W. (1984), *CAD/CAM : Computer Aided Design and Manufacturing*, Prentice-Hall, Englewood Cliffs, NJ.

Gunasekaran A., Babu A.S., Ramaswamy N. (1990), Machine assignment policies in a multistage, multifaclilty and multiproduct production inventory system, *Int. J. Systems Sci.*, Vol. 21, No.1 1, pp. 2305-2315.

Gunasingh K.R., Lashkari R.S. (1989), Machine grouping problem in cellular manufacturing systems - an integer programming approach, *Int. J. Prod. Res.* , Vol. 27, No. 9, pp. 1465-1473.

Hamann Th., Vernadat F. (1991), Automated Materials Handling Systems in FMS for COALA: Viewpoint of INRIA, Working paper COALA, Version 1.0, INRIA-Lorraine, Metz, France.

Hamann Th., Proth J.M., Xie X. (1991), Manufacturing Cell Formation with Identical Machines : A Simulated Annealing Approach, *7th International Conference on CAD/CAM, Robotics and Factories of the Future*, London, UK.

Hamann Th., Proth J-M., Souilah A., Vernadat F.,Xie X. (1992), COALA: A manufacturing plant layout approach, In(G.J. Olling, F. Kimura, EDS.)

Human Aspects in Computer-Integrated Manufacturing, North-Holland, Amsterdam, 1992, pp. 789-796. (Proc. of PROLAMAT '92, Tokyo, Japan, June 24-26.

Hamann Th., Vernadat F. (1992), The intra-cell layout problem in automated manufacturing systems, *8th International Conference on CAD/CAM, Robotics and Factories of the Future*, Metz ,France.

Harhalakis G., Nagi R., Proth J.M. (1990a), An efficient heuristic in manufacturing cell formation for group technology applications, *Int. J. Prod. Res.*, Vol. 28, No. 1, pp. 185-198.

Harhalakis G., Proth J.M., Xie X. (1990b), Manufacturing cell design using simulated annealing : an industrial application, *J. Intelligent Manufacturing*, No. 1, pp. 185-191.

Harhalakis G., Minis I., Proth J.M., Satish J., (1992), CLASS : Computerized layout solutions using simulated annealing, *Int. J. Prod. Res.*, Vol. 30, No. 1,pp. 95-108.

Hassan M.M.D., Hogg G.L., Smith D.R. (1986), A construction algorithm for area placement evaluation : SHAPE, *Int. J. Prod. Res.*, Vol. 24, No. 5, pp. 1283-1295.

Hatvany J. (1983), World Survey of CAM, published by *Butterworths*, Borough Green, Sevenoaks, Kent, TN 15 8PH, UK.

Heragu S., Kusiak A. (1986), A construction algorithm for the facility layout problem, *Working paper No. 14/86*, Dep. of Mechanical and Industriel Engineering, University of Manitoba, Winnipeg, Manitoba, Canada.

Heragu S., Kusiak A. (1987), Analysis of expert systems in manufacturing design ; *IEEE Transactions on systems, man and cybernetics*; Vol.SMC-17, No.6.

Heragu S., Kusiak A. (1988a), Machine layout problem in flexible manufacturing systems, *Operations Research*, Vol. 36, No. 2.

Heragu S., Kusiak A. (1988b), Knowledge based system for machine layout (KBML), *Int. Industrial Engineering Conference Proceedings*,.

Heragu S., Kusiak A. (1990), Machine layout: an optimization and knowledge-based approach, *Int. J. Prod. Res.*, Vol. 28, No. 4, pp. 615-635.

Heragu S., Kusiak A. (1991), Efficient models for the facility layout problem, *Eurepean Journal of Operational Research, No. 53, pp. 1-13.*

Hillier F.S. (1963), Quantitative tools for plant layout analysis, *Journal of Industrial Engineering* , 14, pp. 33-40.

Hillier F.S., Connors M.M. (1966), Quadratic assignment problem algorithms and the location of indivisible facilities, *Management Science* 13, pp. 42-57.

Hitchings G.G., Cottam M. (1976), An efficient heuristic procedure for solving the layout design problem, *Omega*, 4(2), pp. 205-214.

Hogg G.L., Hassan M.M.D. (1987), A review of graph theory application to the facilities layout problem, *OMEGA Int. J. of Mgmt. Sci.*, Vol. 15, No. 4, pp. 291-300.

Hollier R.H., Gelders L.F. (1988), Automated guided vehicle systems, Proceedings of the 6th international conference, 25-26 October 1988, Brussels, Belgium, *IFS Publications/Springer Verlag.*

ILOG S.A. (1991), SMECI - Générateur de Systèmes Experts, Version 1.5, *ILOG S.A., 2,avenue Galliéni, 94250 Gentilly,* France.

Jacob R.J.K., Froscher J.N. (1990), A software engineering methodology for rule-based systems, *IEEE Transactions on knowledge and data engineering,* Vol. 2, No. 2.

Jetschny W. (1991), CADLAY - Baustein LAYOS zur rechnerunterstützten Layoutoptimierung und -simulation, *TU Dresden, Fakultät Maschinenwesen,* Dresden, Germany.

Kaku B.K., Thompson G.L., Baybars I. (1986), A heuristic method for the multi-story layout problem, *Eurepean Journal of Operational Research,* Vol. 37, pp. 384-397.

Khalil T.M. (1973), Facilities relative allocation technique (FRAT), *Int. J. Prod. Res.*, 11, (2), pp. 183-194.

Khator S., Moodie C. (1984), Computer assisted plant layout using a graphics editor, *Computer & Indus. Engineering,* Vol. 8, No. 3/4, pp. 171-179.

Kirkpatrick C.D., Gelatt Jr., Vecchi M.P. (1983), Optimization by Simulated Annealing, *Science*, No. 4598, Vol. 220, pp. 671-680.

Knight T.P., Kim S.H. (1991), A knowledge system for integrated product design, *J. Int. Manuf.*, Vol. 2, pp. 17-25.

Koch U. (1989), Layoutplanung - Theorie und Praxis, (CIM-TT LAYOUT 1 UWK 121 IPA), *Fraunhofer-Institut für Produktionstechnik und Automatisierung*, Stuttgart, Germany.

Koopmans T.C., Beckman M. (1957), Assignement problems and the location of economic activities, *Econometrica* 25, 53-76.

Kumara S.R.T., Kashyap R.L., Moodie C.L. (1987), Expert system for industrial facilities layout planning and analysis, *Comp. ind. Engineering*, Vol. 12, No. 2, pp. 143-152.

Kusiak A. (1987a), The generalized group technology concept, *Int. J. Prod. Res.*, Vol. 25, No. 4, pp. 561-569.

Kusiak A. (1987b), Artificial Intelligence and operations research in flexible manufacturing systems, *University of Manitoba*, Canada, Infor vol. 25, No. 1.

Kusiak A. (1990), *Intelligent Manufacturing systems*, Prentice Hall International Series in Industrial and Systems Engineering, Englewood Cliffs, New Jersey, 07632

Kusiak A., Heragu S. (1987), The facility layout problem, *Eurepean Journal of Operational Research*, Vol. 29, pp. 229-251.

Kusiak A. (1988), EXGT-S : A knowledge based system for group technology, *Int. J. Prod. Res.*,Vol. 26, No. 5, pp. 887-904.

Kusiak A., Chow W.S. (1988), Decomposition of manufacturing systems, *IEEE J. of robotics and automation*, Vol. 4, No. 5.

Land A.H. (1963), A problem of assignment with interrelated costs, *Operations Research Quarterly*, 14, pp. 185-198.

Lawler E.L. (1963), The quadratic assignment problem, Management Science, 9, pp. 586-599.

Lee R.C., Moore J.M. (1967), CORELAP - COmputerized RElationship LAyout Planning, Dep. of industrial engineering, Northeastern Univertity, U.S.A., *The J. of Industial Engineering*, Vol. 18, No. 3.

Levi P. (1988), Planen für autonome Montageroboter, *Technische Universität München*, Institut für Informatik, München, Germany.

Mahadevan B., Narendran T.T. (1990), Design of an automated guided vehicle-based material handling system for a flexible manufacturing system, *Int. J. Prod. Res.*, Vol. 28, No. 9, pp. 1611-1622.

Matson J.O., White A. (1982), Operational research and material handling, *Eurepean Journal of Operational Research*, No. 11, pp. 309-318, North Hollland Publishing Company.

Malakooti B. (1987), Computer-aided facility layout selection (CAFLAS) with applications to multiple criteria manufacturing planning problems, *Large Scale Systems* 12, pp. 109-123.

Malakooti B. (1988), Multiple objective facility layout: a heuristic to generate efficient alternatives, *University of Cleveland*, Ohio.

Mellichamp J.M., Won O.J., Wahab A.F.A. (1990), FMS Designer : An expert system for flexible manufacturing system design, *Int. J. Prod. Res.*, Vol. 28, No. 11, pp. 2013-2024.

Metropolis N., Rosenbluth A., Teller A., Teller E. (1953), Equation of state calculations by fast computing machine, J. Chem. Phys. 21, pp. 1087-1092.

Müller Th. (1983), Automated guided vehicles, *IFS (publications) Ltd*, UK, Springer-Verlag, Berlin, Heidelberg, New York.

Muther R., McPherson K. (1970), Four approaches to computerized layout planning, *Industrial Engineering*, pp. 39-43, february 1970.

Nozari A., Enscore jr. E.E. (1981), Computerized facility layout with graph theory, *Computer & Industrial Engineering*, Vol. 5, No. 3, pp. 183-193.

Nugent C.E., Vollmann T.E., Ruml R. (1968), An experimenal comparison of techniques for the assignment of facilities to locations, *Operations Research*, 16, pp. 150-173.

Oba F., Tsumura T., Kato K., Yasuda K. (1985), Design Concept and Information Processing for fundamental design of flexible manufacturing systems, *Nagova University*, U.S.A.

O'Brien C., Abdel Barr S.E.Z. (1980), An interactive approach to computer aided facility layout, *Int. J. Prod. Res.*, 18(2), pp. 201-211.

O'Hare G.M.P. (1990), Designing Intelligent Manufacturing Systems : A distributed artifical intelligence approach, IMS '89, *Comuters in Industry*, Vol. 15, pp. 17-25.

Pham D.T., Yeo S.H. (1991), Strategies for gripper design and selection in robotic assembly, *Int. J. Prod. Res.*, Vol. 29, No. 2, pp. 303-316.

Picone C.J., Wilhelm W.E. (1984), Perturbation scheme to improve Hillier's solution to the facilities layout problem, *Management Science*, 30(10), pp. 1238-1249.

Proth J.M. (1992), Conception et gestion des systèmes de production, Presses Universitaires de France, No 37997

Proth J.M., Souilah A. (1991), Agencement des systèmes de fabrication cellulaires, *Rapports de Recherche No.1442 de l'INRIA*, France, Juin 1991.

Proth J.M., Souilah A. (1992), Near-Optimal Layout Algorithm Based on Simulated Annealing, *Int. J. Systems Automation: Research and Applications (SARA)*, 2, pp. 227-243.

Pun L. (1990), Utilisation of artificial intelligence techniques for the design of production management systems, IMS '89, *Computers in Industry*, Vol. 15, pp. 7-15.

Rehr W. (1989), Automatisierung mit Industrierobotern, Komponenten, Programmierung, Anwendung, *Vieweg Verlag*, Germany.

Richard M.D. (1955), Practical Plan Layout, *McGraw-Hill Book Company*, 1955.

Sahni S., Gonzalez T. (1976), P-complete approximation problem, *Journal of Associated Computing Machinery* 23(3), pp. 555-565.

Scriabin M., Vergin R.C. (1985), A cluster-analytic approach to facility layout, *Management Science*, 31(1), pp. 33-49.

Seehof J.M., Evans W.O. (1967), Automated layout design program, *The journal of Industrial Engineering* 18(2), pp. 690-695.

Souilah A. (1991), Agencement des ressources dans un atelier de fabrication, *Thèse de Magister*, Centre de Développement des Technologies Avancées, Algérie.

Tam K.Y., Li S.G. (1991), A hierarchical approach to the facility layout problem, *Int. J. Prod. Res.*, Vol. 29, No. 1, pp. 165-184.

Tompkins J.A., Reed R. Jr. (1976), An applied model for the facilities design problem, *Int. J. Prod. Res..*, 14(5), pp. 583-595.

Tompkins J.A. (1983), TOMPKINS Associates Inc., Plant Layout, *Handbook of industrial engineering*, Salvendy G., Purdue University, John Wiley & Sons, Berlin, Heidelberg, New York.

Urban T. (1981), A multiple criteria model for the facility layout problem, *Int. J. Prod. Res.*, Vol. 25, No. 12, pp. 183-193.

Vannelli A., Kumar K.R. (1986), A method for finding minimal bottle-neck cells for grouping part-machine families, *Int. J. Prod. Res.*, Vol. 24, No. 2, pp. 387-400.

Vohra T., Chen D.S., Chang J.C., Cheng H.C. (1990), A network approach to cell formation in cellular manufacturing, *Int. J. Prod. Res.*, Vol. 28, No. 11, pp. 2075-2084.

Vollmann T.E., Nugent C.E., Zartler (1968), A computerized model for office layout, *The journal of Industrial Engineering*, 19, pp. 321-327.

Waghodekar P.H., Sahu S. (1984), Machine-component cell formation in group technology : MACE, *Int. J. Prod. Res.*, Vol. 22, No. 6, pp. 937-948.

Warnecke H.J., SCHRAFT R.D., STURZ W. (1980), Computer Aided Production - Industrial Robots-, *Computer Aided Production Seminary*, Bordeaux, France.

Warnecke H.J., Steinhilper R., Roth H.-P., Weber T. (1989), Planning and realization of FMS: Case studies, experiences and recommendations, *Fraunhofer Institute for Manufacturing Engineering and Automation (IPA)*, Stuttgart, West-Germany.

Zoller K., Adendorff K. (1972), Layout planning by computer simulation, *AIIE Transactions* 4(2), pp. 116-125.

ANNEXE A

LE RECUIT SIMULE

Historique et présentation

1. Historique

Le recuit simulé est une méthode d'optimisation globale qui cherche à éviter les optima locaux. Il est né d'études menées dans le domaine de la physique de la matière désordonnée. En 1953, Metropolis, Rosenbluth et Teller ont proposé un algorithme de simulation de l'équilibre thermique d'un solide. En 1983, Kirkpatrick, Gellat et Vecchi d'une part et Cerny d'autre part ont découvert qu'il existait une analogie entre l'optimisation d'une fonction de coût et le refroidissement d'un solide, et qu'il était possible d'optimiser une fonction en appliquant le critère de Metropolis (Metropolis et al., 1953). On peut ainsi résoudre des problèmes combinatoires NP-complets comme le problème du voyageur de commerce.

2. Le phénomène physique

La méthode d'optimisation appelée "recuit simulé" est issue d'un phénomène physique nommé "recuit" qui peut s'énoncer comme suit : lorsque l'on abaisse de manière contrôlée la température d'un ensemble de particules en interaction, ce système trouve naturellement, à basse température, un état d'énergie minimale.

En effet, si l'on part d'un état désordonné d'énergie élevée à haute température, le milieu refroidi progressivement et les molécules se déplacent de moins en moins facilement et ont tendance à chercher leur place dans un état ordonné d'énergie minimale, appelé "cristal". Un processus d'optimisation très complexe s'effectue ainsi spontanément. Cette descente vers le cristal peut se réaliser en passant par des états d'énergie supérieure.

Par contre, si l'on abaisse brutalement la température, les molécules sont "gelées" et on obtient un état désordonné d'énergie élevé, appelé "chaos".

A chaque température T, le solide peut se trouver dans un état d'équilibre d'énergie E avec une probabilité donnée par la loi de Boltzmann :

$$\text{Prob (Energie} = E / T) = (1 / Z(T)) \cdot \exp(- E / KT)$$

où :

Z(T) est un facteur de normalisation connu comme étant la fonction de partition dépendant de la température T :

$$Z(T) = \int_0^{+\infty} \exp(-E/KT)dE$$

K est la constante de Boltzmann.

3. Le principe de la méthode

Pour simuler l'évolution de l'équilibre thermique d'un solide en fonction d'une température T fixée, Metropolis a proposé l'approche suivante :

Etant donné l'état d'un solide caractérisé par la position de ces particules, une petite perturbation est effectuée par un petit déplacement d'une particule. Cette perturbation produit une différence d'énergie ΔE entre l'état perturbé et l'état initial.

Si $\Delta E < 0$, c'est-à-dire si la perturbation entraîne une diminution de l'énergie, alors le processus continue avec le nouvel état.

Si $\Delta E \geq 0$, alors la probabilité d'acceptation de l'état perturbé est donnée par :

$$P = \exp(-\Delta E / KT)$$

Cette méthode d'acceptation des nouveaux états est appelée algorithme de Metropolis. Elle permet, quand la température est encore élevée, d'accepter des configurations d'énergie plus élevée, ce qui permet d'éviter les optima locaux et d'obtenir l'optimum global. Notons toutefois que plus la température est basse, plus il est difficile de dégrader l'état du solide du fait que la probabilité d'acceptation de l'état perturbé s'affaiblit petit à petit.

Lorsque l'on est en mesure de réaliser les analogies suivantes, il est alors possible d'appliquer l'algorithme de Metropolis à un problème d'optimisation combinatoire.

Système physique	Problème d'optimisation
Energie	Fonction de coût
Etats d'un solide	Configuration
Température	Paramètre de contrôle
Perturbation	Modification élémentaire

L'algorithme du recuit simulé peut alors être considéré comme une succession d'algorithmes de Metropolis évalués au moyen d'une succession de valeurs décroissantes du paramètre de contrôle.

L'algorithme du recuit simulé fait partie de la famille des heuristiques par voisinage car à chaque étape on recherche une nouvelle solution dans le voisinage de la solution courrante.

4. Application du recuit simulé à la formation des cellules

Dans notre étude, le problème d'optimisation est le suivant :

(1) fonction de coût → trafic intra-cellulaire x distances
(2) configuration → partition de l'ensemble des machines
(3) paramètre de contrôle → paramètre de contrôle
(4) modification élémentaire → 3 transformations élémentaires possibles

Les trois premiers points ont été définis ci-dessus. Les trois transformations élémentaires possibles pour définir une nouvelle solution dans le voisinage de la solution courrante sont définies dans le paragraphe III.4.2. Remarquons d'une part que toute configuration voisine d'une configuration admissible est encore une configuration admissible. D'autre part, toute configuration admissible peut être atteinte à partir d'une configuration admissible quelconque. (T_f est la valeur finale du paramètre de contrôle, q ($o<q<1$) est le coefficient de réduction de la température tout au long des calculs (dans la pratique, on le choisit en général dans l'intervale [0,8, 0,98]), S* est la meilleure solution trouvée, L est le nombre d'itérations pour chaque valeur du paramètre de contrôle et TR est la fonction de coût, c'est à dire le trafic intra-cellulaire x distances).

Avant de lancer l'algorithme, l'utilisateur doit au préalable fixer les valeurs des paramètres T_f, q et L.

L'algorithme du recuit simulé est alors le suivant :

1. Produire une solution admissible initiale S.
2. Choisir une valeur du paramètre de contrôle T_0 et faire $T = T_0$
3. Tant que $T > T_f$, faire :
   3.1. Répéter L fois les étapes 3.1.1 à 3.1.4
      3.1.1. Produire une solution admissible S' dans le voisinage de S
      3.1.2. Faire $\Delta TR = TR(S') - TR(S)$
         3.1.2.1. Si $\Delta TR \leq 0$
            3.1.2.1.1. $S = S'$
            3.1.2.1.2. $S^* = S'$ et $TR(S^*) = TR(S')$
         3.1.2.2. Si $\Delta TR > 0$
      3.1.3. Tirer aléatoirement une valeur R dans [0,1]
      3.1.4. Si $\exp(-\Delta TR / T) > R$, faire $S = S'$
   3.2. Diminuer la valeur du paramètre de contrôle $T = q.T$

Fintantque

On voit donc que grâce à la condition de Metropolis on peut, pour une même valeur du paramètre de contrôle, garder en mémoire des solutions qui n'améliorent pas le critère et ce avec une certaine probabilité pour continuer l'algorithme donnant ainsi une chance de "sortir" d'un optimum local. Cependant, cette probabilité décroit avec la valeur du paramètre de contrôle T.

ANNEXE B

1. SYSTEMES EXPERTS

2. AGENCEMENT ET EVALUATION

1. SYSTEMES EXPERTS

1.1 Liste des règles pour choisir un moyen de transport

1.2 Listings des objets, des prototype et des règles

Règles pour choisir un AGV

SI les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un AGV peut transporter (300 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un AGV peut transporter (2 m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un AGV peut transporter (1 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un AGV peut transporter (1 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un AGV (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un AGV sur les axes x(∞), y(∞), et z (0m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(1 m/s),y(1 m/s), et z(0 m/s) d'un AGV

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un AGV (0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un AGV (1 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un AGV (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un AGV (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un AGV (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un AGV (50°C)

ET le type d'énergie disponible = au type d'énergie d'un AGV (électrique)

ALORS utiliser AGV comme moyen de transport.

Règles pour choisir un AGV-Palette

SI les produits peuvent être transportés avec une palette (Possibilité de palettisation des produits = 'vrais')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un AGV-palette peut transporter (300kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un AGV-palette peut transporter (2 m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un AGV-palette peut transporter (1 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un AGV-palette peut transporter (1 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un AGV-palette (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un AGV-palette sur les axes x(∞), y(∞), et z (0 m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(1 m/s),y(1 m/s), et z(0 m/s) d'un AGV-palette

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un AGV-palette (0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un AGV-palette (2 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un AGV-palette (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un AGV-palette (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un AGV-palette (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un AGV-palette (50°C)

ET le type d'énergie disponible = au type d'énergie d'un AGV-palette (électrique)

ALORS utiliser **AGV-Palette** comme moyen de transport.

Règles pour choisir un Chariot

SI les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un chariot peut transporter (500 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un chariot peut transporter (2m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un chariot peut transporter (1 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un chariot peut transporter (1 m)

ET le flux de produits est unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un chariot (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un chariot sur les axes x(∞), y(∞), et z (0m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(0,5 m/s),y(0,5 m/s), et z(0 m/s) d'un chariot

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un chariot (0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un chariot (5 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un chariot (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un chariot (petit)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un chariot (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un chariot (70°C)

ET le type d'énergie disponible = au type d'énergie d'un chariot (électrique, moteur à combustion)

ALORS utiliser **Chariot** comme moyen de transport.

Règles pour choisir un Chariot poids lourd

SI les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un chariot poids lourd peut transporter (1000 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un chariot poids lourd peut transporter (2m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un chariot poids lourd peut transporter (1 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un chariot poids lourd peut transporter (1 m)

ET le flux de produits est unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un chariot poids lourd (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un chariot poids lourd sur les axes x(∞), y(∞), et z (0m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(0,3 m/s),y(0,3 m/s), et z(0 m/s) d'un chariot poids lourd

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un chariot poids lourd (0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un chariot poids lourd (10 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un chariot poids lourd (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un chariot poids lourd (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un chariot poids lourd (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un chariot poids lourd (70°C)

ET le type d'énergie disponible = au type d'énergie d'un chariot poids lourd (électrique, moteur à combustion)

ALORS utiliser **Chariot poids lourd** comme moyen de transport.

Règles pour choisir un Chariot palette

SI les produits peuvent être transportés avec une palette (Possibilité de palettisation des produits = 'vrais')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un chariot palette peut transporter (500kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un chariot palette peut transporter (2m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un chariot palette peut transporter (1m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un chariot palette peut transporter (1m)

ET le flux de matériel est unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un chariot palette (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un chariot palette sur les axes x(∞), y(∞), et z (0m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(0,5 m/s),y(0,5 m/s), et z(0 m/s) d'un chariot palette

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un chariot palette (0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un chariot palette (5mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un chariot palette (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un chariot palette (petit)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un chariot palette (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un chariot palette (70°C)

ET le type d'énergie disponible = au type d'énergie d'un chariot palette (électrique, moteur à combustion)

ALORS utiliser **Chariot palette** comme moyen de transport.

Règles pour choisir un Chariot palette lourd

SI les produits peuvent être transportés avec une palette (Possibilité de palettisation des produits = 'vrais')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un chariot palette lourd peut transporter (1000kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un chariot palette lourd peut transporter (2m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un chariot palette lourd peut transporter (1m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un chariot palette lourd peut transporter (1m)

ET le flux de matériel est unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un chariot palette lourd (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un chariot palette lourd sur les axes x(∞), y(∞), et z (0m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(0,25 m/s),y(0,25 m/s), et z(0 m/s) d'un chariot palette lourd

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un chariot palette lourd(0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un chariot palette lourd (10mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un chariot palette lourd (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un chariot palette lourd (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un chariot palette lourd (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un chariot palette lourd (70°C)

ET le type d'énergie disponible = au type d'énergie d'un chariot palette lourd (moteur à combustion)

ALORS utiliser **Chariot palette lourd** comme moyen de transport.

Règles pour choisir un Convoyeur

SI les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un convoyeur peut transporter (300 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un convoyeur peut transporter (2m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un convoyeur peut transporter (1m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un convoyeur peut transporter (1 m)

ET le flux de matériel est unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un convoyeur (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un convoyeur sur les axes x(∞), y(∞), et z (0m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(1 m/s),y(1 m/s), et z(0 m/s) d'un convoyeur

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un convoyeur (0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un convoyeur (2mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un convoyeur (petit)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un convoyeur (important)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un convoyeur (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un convoyeur (90°C)

ET le type d'énergie disponible = au type d'énergie d'un convoyeur (électrique)

ALORS utiliser **Convoyeur** comme moyen de transport.

Règles pour choisir un Convoyeur Palette

SI les produits peuvent être transportés avec une palette (Possibilité de palettisation des produits = 'vrais')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un Convoyeur palette peut transporter (300 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un Convoyeur palette peut transporter (2m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un Convoyeur palette peut transporter (1m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un Convoyeur palette peut transporter (1 m)

ET le flux de matériel est unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un Convoyeur palette (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un Convoyeur palette sur les axes x(∞), y(∞), et z (0m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(1 m/s),y(1 m/s), et z(0 m/s) d'un Convoyeur palette

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un Convoyeur palette (0)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un Convoyeur palette (2mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un Convoyeur palette (petit)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un Convoyeur palette (important)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un Convoyeur palette (moyen)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un Convoyeur palette (90°C)

ET le type d'énergie disponible = au type d'énergie d'un Convoyeur palette (électrique)

ALORS utiliser **Convoyeur Palette** comme moyen de transport.

Règles pour choisir un Pont roulant léger 2D

SI le nombre de machines dans la cellule ≤ à 10

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un pont roulant léger 2D peut transporter (25 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un pont roulant léger 2D peut transporter (0,5m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un pont roulant léger 2D peut transporter (0,5 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un pont roulant léger 2D peut transporter (0,5 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un pont roulant léger 2D (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un pont roulant léger 2D sur les axes x(8m), y(0m), et z (3m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(2,5 m/s),y(0 m/s), et z(2,5 m/s) d'un pont roulant léger 2D

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un pont roulant léger 2D (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un pont roulant léger 2D (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un pont roulant léger 2D (180°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un pont roulant léger 2D (0,1 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un pont roulant léger 2D (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un pont roulant léger 2D (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un pont roulant léger 2D **(important)**

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un pont roulant léger 2D **(70°C)**

ET le type d'énergie disponible = au type d'énergie d'un pont roulant léger 2D **(électrique, air sous pression)**

ALORS utiliser **Pont roulant léger 2D** comme moyen de transport.

Règles pour choisir un Pont roulant léger 3D

SI le nombre de machines dans la cellule ≤ à 12

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un pont roulant léger 3D peut transporter (25 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un pont roulant léger 3D peut transporter (0,5m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un pont roulant léger 3D peut transporter (0,5 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un pont roulant léger 3D peut transporter (0,5 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un pont roulant léger 3D (3)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un pont roulant léger 3D sur les axes x(6m), y(6m), et z (3m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(2,5 m/s),y(2,5 m/s), et z(2,5 m/s) d'un pont roulant léger 3D

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un pont roulant léger 3D (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un pont roulant léger 3D (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un pont roulant léger 3D (180°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un pont roulant léger 3D (0,1 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un pont roulant léger 3D (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un pont roulant léger 3D (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un pont roulant léger 3D **(important)**

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un pont roulant léger 3D **(70°C)**

ET le type d'énergie disponible = au type d'énergie d'un pont roulant léger 3D **(électrique, air sous pression)**

ALORS utiliser **Pont roulant léger 3D** comme moyen de transport.

Règles pour choisir un Pont roulant moyen 2D

SI le nombre de machines dans la cellule ≤ à 10

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un pont roulant moyen 2D peut transporter (100 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un pont roulant moyen 2D peut transporter (0,5m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un pont roulant moyen 2D peut transporter (0,5 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un pont roulant moyen 2D peut transporter (0,5 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un pont roulant moyen 2D (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un pont roulant moyen 2D sur les axes x (10m), y (0m), et z (3m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x (1,5 m/s),y (0 m/s), et z (1,5 m/s) d'un pont roulant moyen 2D

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un pont roulant moyen 2D (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x (360°), y (360°), et z (360°) d'un pont roulant moyen 2D

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un pont roulant moyen 2D (120°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un pont roulant moyen 2D (0,2 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un pont roulant moyen 2D (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un pont roulant moyen 2D (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un pont roulant moyen 2D **(important)**

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un pont roulant moyen 2D **(70°C)**

ET le type d'énergie disponible = au type d'énergie d'un pont roulant moyen 2D **(électrique, hydraulique)**

ALORS utiliser **Pont roulant moyen 2D** comme moyen de transport.

Règles pour choisir un Pont roulant moyen 3D

SI le nombre de machines dans la cellule ≤ à 15

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un pont roulant moyen 3D peut transporter (100 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un pont roulant moyen 3D peut transporter (0,5m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un pont roulant moyen 3D peut transporter (0,5 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un pont roulant moyen 3D peut transporter (0,5 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un pont roulant moyen 3D (3)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un pont roulant moyen 3D sur les axes x(10m), y(10m), et z (3m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(1,5 m/s),y(1,5 m/s), et z(1,5 m/s) d'un pont roulant moyen 3D

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un pont roulant moyen 3D (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un pont roulant moyen 3D (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un pont roulant moyen 3D (120°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un pont roulant moyen 3D (0,2 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un pont roulant moyen 3D (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un pont roulant moyen 3D (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un pont roulant moyen 3D **(important)**

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un pont roulant moyen 3D **(70°C)**

ET le type d'énergie disponible = au type d'énergie d'un pont roulant moyen 3D **(électrique, hydraulique)**

ALORS utiliser **Pont roulant moyen 3D** comme moyen de transport.

Règles pour choisir un Pont roulant lourd 2D

SI le nombre de machines dans la cellule ≤ à 10

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un pont roulant lourd 2D peut transporter (300 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un pont roulant lourd 2D peut transporter (0,5m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un pont roulant lourd 2D peut transporter (0,5 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un pont roulant lourd 2D peut transporter (0,5 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un pont roulant lourd 2D (2)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un pont roulant lourd 2D sur les axes x(12m), y(0m), et z (3m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(1 m/s),y(0 m/s), et z(1 m/s) d'un pont roulant lourd 2D

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un pont roulant lourd 2D (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un pont roulant lourd 2D (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un pont roulant lourd 2D (90°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un pont roulant lourd 2D (0,5 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un pont roulant lourd 2D (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un pont roulant lourd 2D (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un pont roulant lourd 2D **(important)**

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un pont roulant lourd 2D **(70°C)**

ET le type d'énergie disponible = au type d'énergie d'un pont roulant lourd 2D **(électrique, hydraulique)**

ALORS utiliser **Pont roulant lourd 2D** comme moyen de transport.

Règles pour choisir un Pont roulant lourd 3D

SI le nombre de machines dans la cellule ≤ à 21

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un pont roulant lourd 3D peut transporter (300 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un pont roulant lourd 3D peut transporter (0,5m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un pont roulant lourd 3D peut transporter (0,5 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un pont roulant lourd 3D peut transporter (0,5 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un pont roulant lourd 3D (3)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un pont roulant lourd 3D sur les axes x(12m), y(12m), et z (3m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(1 m/s),y(1 m/s), et z(1 m/s) d'un pont roulant lourd 3D

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un pont roulant lourd 3D (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un pont roulant lourd 3D (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un pont roulant lourd 3D (90°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un pont roulant lourd 3D (0,5 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un pont roulant lourd 3D (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un pont roulant lourd 3D (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un pont roulant lourd 3D **(important)**

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un pont roulant lourd 3D **(70°C)**

ET le type d'énergie disponible = au type d'énergie d'un pont roulant lourd 3D **(électrique, hydraulique)**

ALORS utiliser **Pont roulant lourd 3D** comme moyen de transport.

Règles pour choisir un Pont roulant spécial lourd 3D

SI le nombre de machines dans la cellule ≤ à 21

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un pont roulant spécial lourd 3D peut transporter (500 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un pont roulant spécial lourd 3D peut transporter (0,5m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un pont roulant spécial lourd 3D peut transporter (0,5 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un pont roulant spécial lourd 3D peut transporter (0,5 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un pont roulant spécial lourd 3D (3)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes x, y et z ≤ la distance maximale de transport d'un pont roulant spécial lourd 3D sur les axes x(12m), y(12m), et z (3m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x(0,25 m/s),y(0,25 m/s), et z(0,25 m/s) d'un pont roulant spécial lourd 3D

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un pont roulant spécial lourd 3D (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un pont roulant spécial lourd 3D (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un pont roulant spécial lourd 3D (60°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un pont roulant spécial lourd 3D (2mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un pont roulant spécial lourd 3D (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un pont roulant spécial lourd 3D (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un pont roulant spécial lourd 3D **(important)**

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un pont roulant spécial lourd 3D **(70°C)**

ET le type d'énergie disponible = au type d'énergie d'un pont roulant spécial lourd 3D **(électrique, hydraulique)**

ALORS utiliser **Pont roulant spécial lourd 3D** comme moyen de transport.

Règles pour choisir un Robot Articulé

SI le nombre de machines dans la cellule ≤ à 5

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un robot articulé peut transporter (10 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un robot articulé peut transporter (1 m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un robot articulé peut transporter (1 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un robot articulé peut transporter (1 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un robot articulé (3)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes ≤ la distance maximale de transport d'un robot articulé sur les axes x, y, et z (2m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x,y, et z d'un robot articulé (2,5 m/s)

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un robot articulé (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un robot articulé (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un robot articulé (120°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un robot articulé (0,1 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un robot articulé (important)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un robot articulé (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un robot articulé (petit)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un robot articulé (70°C)

ET le type d'énergie disponible = au type d'énergie d'un robot articulé (électrique, air sous pression)

ALORS utiliser **Robot Articulé** comme moyen de transport.

Règles pour choisir un Robot Poids Lourd

SI le nombre de machines dans la cellule ≤ à 5

ET les produits ne peuvent pas être transportés avec une palette (Possibilité de palettisation des produits = 'faux')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un robot poids lourd peut transporter (50 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un robot poids lourd peut transporter (1 m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un robot poids lourd peut transporter (1 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un robot poids lourd peut transporter (1 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un robot poids lourd (3)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes ≤ la distance maximale de transport d'un robot poids lourd sur les axes x, y, et z (2m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x,y, et z d'un robot poids lourd (0,5 m/s)

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un robot poids lourd (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un robot poids lourd (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un robot poids lourd (90°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un robot poids lourd (1 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un robot poids lourd (moyen)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un robot poids lourd (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un robot poids lourd (petit)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un robot poids lourd (70°C)

ET le type d'énergie disponible = au type d'énergie d'un robot poids lourd (électrique, hydraulique)

ALORS utiliser **Robot Poids Lourd** comme moyen de transport.

Règles pour choisir un Robot Palette

SI le nombre de machines dans la cellule ≤ à 5

ET les produits peuvent être transportés avec une palette (Possibilité de palettisation des produits = 'vrais')

ET le poids maximal des produits ≤ au poids maximal des produits qu'un robot palette peut transporter (50 kg)

ET la longueur maximale des produits ≤ à la longueur maximale des produits qu'un robot palette peut transporter (1 m)

ET la largeur maximale des produits ≤ à la largeur maximale des produits qu'un robot palette peut transporter (1 m)

ET la hauteur maximale des produits ≤ à la hauteur maximale des produits qu'un robot palette peut transporter (1 m)

ET le flux de matériel n'est pas unidirectionnel

ET le nombre d'axes de transport nécessaires pour servir les machines en produits ≤ au nombre d'axes de translations possibles d'un robot palette (3)

ET la distance de transport nécessaire (l'amplitude de déplacement) sur les axes ≤ la distance maximale de transport d'un robot palette sur les axes x, y, et z (2m)

ET la vitesse de transport nécessaire des produits sur les axes x,y, et z ≤ la vitesse de transport sur les axes x,y, et z d'un robot palette (0,5 m/s)

ET le nombre de rotations de transport nécessaires pour servir les machines ≤ au nombre de rotations possibles avec un robot palette (3)

ET l'angle maximum nécessaire de rotation autour des axes x, y, et z ≤ l'angles maximum de rotation autour des axes x, y, et z d'un robot palette (360°)

ET la vitesse nécessaire de rotation autour des axes x, y, et z ≤ la vitesse de rotation autour des axes x, y, et z d'un robot palette (90°/s)

ET la précision maximale du positionnement des produits devant les machines ≤ à la précision de positionnement d'un robot palette (1 mm)

ET le niveau de flexibilité globale de la cellule ≤ au niveau de flexibilité d'un robot palette (moyen)

ET le niveau de sécurité de manipulation nécessité par les produits ≤ au niveau de sécurité offert par un robot palette (moyen)

ET le niveau de vibration des machines ou de l'environnement ≤ au niveau de vibration pour utiliser un robot palette (petit)

ET la température maximale de l'environnement ≤ à la température d'environnement maximale pour un robot palette (70°C)

ET le type d'énergie disponible = au type d'énergie d'un robot palette **(électrique, hydraulique)**

ALORS utiliser **Robot Palette** comme moyen de transport.

```
defrule select-mhs-start
author claude
explanation Beginning of MHS selection
end-explanation
then
action $(print "Beginning of MHS selection")
end-rule



defrule select-articulated-robot
author thomas
explanation general conditions for the selection of an ARTICULATED-ROBOT
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an ArticulatedRobot
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL <> unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-articulated-robot")
end-rule

defrule possibility-of-palletisation-articulated-robot
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an ArticulatedRobot
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-articulated-robot")
end-rule

defrule max-weight-of-product-articulated-robot
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an ArticulatedRobot in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-articulated-robot")
end-rule

defrule max-length-of-product-articulated-robot
author thomas
explanation maximum length of the longest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an ArticulatedRobot in results^PROBLEM
```



```
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-articulated-robot")
end-rule

defrule max-width-of-product-articulated-robot
author thomas
explanation maximum width of the widest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an ArticulatedRobot in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-articulated-robot")
end-rule

defrule max-height-of-product-articulated-robot
author thomas
explanation maximum height of the highest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an ArticulatedRobot in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-articulated-robot")
end-rule

defrule machine-positioning-accuracy-articulated-robot
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an ArticulatedRobot in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-articulated-robot")
end-rule

defrule prefered-transportation-velocity-articulated-robot
author thomas
explanation prefered transportation velocity inside the cell
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    CELL the cell of PROBLEM and
    MHS an ArticulatedRobot in results^PROBLEM
```

```
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity and
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-articulated-robot")
end-rule



defrule select-light-gantry-robot
author thomas
explanation general conditions for the selection of an LIGHT-GANTRY-MHS
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an LightGantryRobot
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL <> unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-light-gantry-robot")
end-rule

defrule possibility-of-palletisation-light-gantry-robot
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an LightGantryRobot
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-light-gantry-robot")
end-rule

defrule max-weight-of-product-light-gantry-robot
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an LightGantryRobot in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-light-gantry-robot")
end-rule

defrule max-length-of-product-light-gantry-robot
author thomas
explanation maximum length of the longest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
```



```
    PRODUCT the product of PROBLEM and
    MHS an LightGantryRobot in results^PROBLEM
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-light-gantry-robot")
end-rule

defrule max-width-of-product-light-gantry-robot
author thomas
explanation maximum width of the widest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an LightGantryRobot  in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-light-gantry-robot")
end-rule

defrule max-height-of-product-light-gantry-robot
author thomas
explanation maximum height of the highest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an LightGantryRobot in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-light-gantry-robot")
end-rule

defrule machine-positioning-accuracy-light-gantry-robot
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an LightGantryRobot in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-light-gantry-robot")
end-rule

defrule prefered-transportation-velocity-light-gantry-robot
author thomas
explanation prefered transportation velocity inside the cell
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
```

```
    CELL the cell of PROBLEM and
    MHS an LightGantryRobot in results^PROBLEM
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity-light-gantry-robot and
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-light-gantry-robot")
end-rule



defrule select-heavy-duty-robot
author thomas
explanation general conditions for the selection of an HEAVY-GANTRY-MHS
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an HeavyDutyRobot
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL <> unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-heavy-duty-robot")
end-rule

defrule max-weight-of-product-heavy-duty-robot
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyDutyRobot in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-heavy-duty-robot")
end-rule

defrule possibility-of-palletisation-heavy-duty-robot
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyDutyRobot
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-heavy-duty-robot")
end-rule

defrule max-length-of-product-heavy-duty-robot
author thomas
explanation maximum length of the longest product
```



```
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyDutyRobot in results^PROBLEM
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-heavy-duty-robot")
end-rule

defrule max-width-of-product-heavy-duty-robot
author thomas
explanation maximum width of the longest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyDutyRobot in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-heavy-duty-robot")
end-rule

defrule max-height-of-product-heavy-duty-robot
author thomas
explanation maximum height of the longest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyDutyRobot in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-heavy-duty-robot")
end-rule

defrule machine-positioning-accuracy-heavy-duty-robot
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an HeavyDutyRobot in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-heavy-duty-robot")
end-rule

defrule prefered-transportation-velocity-heavy-duty-robot
author thomas
explanation prefered transportation velocity inside the cell
```

```
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    CELL the cell of PROBLEM and
    MHS an HeavyDutyRobot in results^PROBLEM
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity-heavy-duty-robot and
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-heavy-duty-robot")
end-rule



defrule select-medium-gantry-robot
author thomas
explanation general conditions for the selection of an MEDIUM-GANTRY-MHS
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an MediumGantryRobot
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL <> unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-medium-gantry-robot")
end-rule

defrule possibility-of-palletisation-medium-gantry-robot
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an MediumGantryRobot
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-medium-gantry-robot")
end-rule

defrule max-weight-of-product-medium-gantry-robot
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an MediumGantryRobot in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-medium-gantry-robot")
end-rule

defrule max-length-of-product-medium-gantry-robot
author thomas
```



```
explanation maximum length of the longest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an MediumGantryRobot in results^PROBLEM
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-medium-gantry-robot")
end-rule

defrule max-width-of-product-medium-gantry-robot
author thomas
explanation maximum width of the widest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an MediumGantryRobot  in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-medium-gantry-robot")
end-rule

defrule max-height-of-product-medium-gantry-robot
author thomas
explanation maximum height of the highest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an MediumGantryRobot in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-medium-gantry-robot")
end-rule

defrule machine-positioning-accuracy-medium-gantry-robot
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an MediumGantryRobot in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-medium-gantry-robot")
end-rule

defrule prefered-transportation-velocity-medium-gantry-robot
author thomas
```

```
explanation prefered transportation velocity inside the cell
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    CELL the cell of PROBLEM and
    MHS an MediumGantryRobot in results^PROBLEM
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity-medium-gantry-robot and
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-medium-gantry-robot")
end-rule

defrule select-conveyor
author thomas
explanation general conditions for the selection of an CONVEYOR
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an Conveyor
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL = unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-conveyor")
end-rule

defrule possibility-of-palletisation-conveyor
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an Conveyor
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-conveyor")
end-rule

defrule max-weight-of-product-conveyor
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an Conveyor in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-conveyor")
end-rule

defrule max-length-of-product-conveyor
author thomas
explanation maximum length of the longest product
end-explanation
```



```
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an Conveyor in results^PROBLEM
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-conveyor")
end-rule

defrule max-width-of-product-conveyor
author thomas
explanation maximum width of the widest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an Conveyor  in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-conveyor")
end-rule

defrule max-height-of-product-conveyor
author thomas
explanation maximum height of the highest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an Conveyor in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-conveyor")
end-rule

defrule machine-positioning-accuracy-conveyor
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an Conveyor in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-conveyor")
end-rule

defrule prefered-transportation-velocity-conveyor
author thomas
explanation prefered transportation velocity inside the cell
end-explanation
```

```
let PROBLEM a mhs-problem in *mhs-problem* and
    CELL the cell of PROBLEM and
    MHS an Conveyor in results^PROBLEM
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity-conveyor and
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-conveyor")
end-rule



defrule select-AGV
author thomas
explanation general conditions for the selection of an AGV
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an AGV
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL <> unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-AGV")
end-rule

defrule possibility-of-palletisation-AGV
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an AGV
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-AGV")
end-rule

defrule max-weight-of-product-AGV
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an AGV in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-AGV")
end-rule

defrule max-length-of-product-AGV
author thomas
explanation maximum length of the longest product
```



```
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an AGV in results^PROBLEM
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-AGV")
end-rule

defrule max-width-of-product-AGV
author thomas
explanation maximum width of the widest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an AGV  in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-AGV")
end-rule

defrule max-height-of-product-AGV
author thomas
explanation maximum height of the highest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an AGV in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-AGV")
end-rule

defrule machine-positioning-accuracy-AGV
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an AGV in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-AGV")
end-rule

defrule prefered-transportation-velocity-AGV
author thomas
explanation prefered transportation velocity inside the cell
```

```
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    CELL the cell of PROBLEM and
    MHS an AGV in results^PROBLEM
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity-AGV and
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-AGV")
end-rule



defrule select-heavy-gantry-robot
author thomas
explanation general conditions for the selection of an HeavyGantryRobot
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an HeavyGantryRobot
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL <> unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-heavy-gantry-robot")
end-rule

defrule possibility-of-palletisation-heavy-gantry-robot
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyGantryRobot
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-heavy-gantry-robot")
end-rule

defrule max-weight-of-product-heavy-gantry-robot
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyGantryRobot in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-heavy-gantry-robot")
end-rule

defrule max-length-of-product-heavy-gantry-robot
author thomas
```



```
explanation maximum length of the longest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyGantryRobot in results^PROBLEM
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-heavy-gantry-robot")
end-rule

defrule max-width-of-product-heavy-gantry-robot
author thomas
explanation maximum width of the widest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyGantryRobot  in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-heavy-gantry-robot")
end-rule

defrule max-height-of-product-heavy-gantry-robot
author thomas
explanation maximum height of the highest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an HeavyGantryRobot in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-heavy-gantry-robot")
end-rule

defrule machine-positioning-accuracy-heavy-gantry-robot
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an HeavyGantryRobot in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-heavy-gantry-robot")
end-rule

defrule prefered-transportation-velocity-heavy-gantry-robot
author thomas
```

```
explanation prefered transportation velocity inside the cell
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    CELL the cell of PROBLEM and
    MHS an HeavyGantryRobot in results^PROBLEM
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity-heavy-gantry-robot and
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-heavy-gantry-robot")
end-rule



defrule select-special-heavy-gantry-robot
author thomas
explanation general conditions for the selection of an SpecialHeavyGantryRobot
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    CELL the cell of PROBLEM and
    MHS an SpecialHeavyGantryRobot
if NoMachines^CELL <= MaxNoMachines^MHS and
   TypeOfMaterialFlow^CELL <> unidirectional
then MHS & results^PROBLEM
action $(print "Rule application: select-special-heavy-gantry-robot")
end-rule

defrule possibility-of-palletisation-special-heavy-gantry-robot
author thomas
explanation is it possible to transport the products with pallets
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an SpecialHeavyGantryRobot
if PossibilityPalletisation^PRODUCT <> PossibilityOfPalletHandling^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = possibility-of-palletisation-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: possibility-of-palletisation-special-heavy-gantry-robo
)
end-rule

defrule max-weight-of-product-special-heavy-gantry-robot
author thomas
explanation maximum weight of the product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an SpecialHeavyGantryRobot in results^PROBLEM
if ProductWeight^PRODUCT > MaxTranspWeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-weight-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-weight-of-product-special-heavy-gantry-robot")
end-rule
```



```
defrule max-length-of-product-special-heavy-gantry-robot
author thomas
explanation maximum length of the longest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an SpecialHeavyGantryRobot in results^PROBLEM
if ProductLength^PRODUCT > MaxProductLength^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-length-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-length-of-product-special-heavy-gantry-robot")
end-rule

defrule max-width-of-product-special-heavy-gantry-robot
author thomas
explanation maximum width of the widest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an SpecialHeavyGantryRobot  in results^PROBLEM
if ProductWidth^PRODUCT > MaxProductWidth^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-width-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-width-of-product-special-heavy-gantry-robot")
end-rule

defrule max-height-of-product-special-heavy-gantry-robot
author thomas
explanation maximum height of the highest product
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    PRODUCT the product of PROBLEM and
    MHS an SpecialHeavyGantryRobot in results^PROBLEM
if ProductHeight^PRODUCT > MaxProductHeight^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = max-height-of-product and
     anomaly-locality^ANOMALY = PRODUCT and
     MHS /& results^PROBLEM
action $(print "Rule application: max-height-of-product-special-heavy-gantry-robot")
end-rule

defrule machine-positioning-accuracy-special-heavy-gantry-robot
author thomas
explanation necessary positioning accuracy to serve the machines
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    MACHINE the machine of PROBLEM and
    MHS an SpecialHeavyGantryRobot in results^PROBLEM
if MachPositioningAccuracy^MACHINE < PositioningAccuracyMHS^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = machine-positioning-accuracy and
     anomaly-locality^ANOMALY = MACHINE and
     MHS /& results^PROBLEM
action $(print "Rule application: machine-positioning-accuracy-special-heavy-gantry-robo
)
end-rule
```

```
defrule prefered-transportation-velocity-special-heavy-gantry-robot
author thomas
explanation prefered transportation velocity inside the cell
end-explanation
let PROBLEM a mhs-problem in *mhs-problem* and
    CELL the cell of PROBLEM and
    MHS an SpecialHeavyGantryRobot in results^PROBLEM
if PreferedTranspVelocity^CELL > TranspVelocity^MHS
then create ANOMALY a mhs-anomaly puis
     mhs-tested^ANOMALY = MHS and
     anomaly-name^ANOMALY = prefered-transportation-velocity-special-heavy-gantry-robot
d
     anomaly-locality^ANOMALY = CELL and
     MHS /& results^PROBLEM
action $(print "Rule application: prefered-transportation-velocity-special-heavy-gantry-
bot")
end-rule




defrule select-mhs-end
author claude
explanation End of MHS selection
end-explanation
then
action $(print "End of MHS selection")
end-rule




defrule select-layout-type-start
author thomas
explanation Beginning of layout type selection
end-explanation
then
action $(print "Beginning of layout type selection")
end-rule




defrule select-circular-single-row-layout
author thomas
explanation general conditions for the selection of an CircularSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    TYPELAYOUT a CircularSingleRowLayout and
    MHS an ArticulatedRobot
if type-mhs^SELECTEDMHS = MHS
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-circular-single-row-layout")
end-rule

defrule select-circular-single-row-layout-2
```



```
author thomas
explanation general conditions for the selection of an CircularSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    TYPELAYOUT a CircularSingleRowLayout and
    MHS a HeavyDutyRobot
if type-mhs^SELECTEDMHS = MHS
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-circular-single-row-layout-3")
end-rule

defrule select-circular-single-row-layout-3
author thomas
explanation general conditions for the selection of an CircularSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    TYPELAYOUT a CircularSingleRowLayout and
    MHS a PalletHandlingRobot
if type-mhs^SELECTEDMHS = MHS
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-circular-single-row-layout-3")
end-rule

defrule select-linear-single-row-layout
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearSingleRowLayout-6 and
    MHS an AGV
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout")
end-rule

defrule select-linear-single-row-layout-2
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearSingleRowLayout-6 and
    MHS an PalletAGV
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-2")
end-rule

defrule select-linear-single-row-layout-3
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
```

```
    TYPELAYOUT a LinearSingleRowLayout-6 and
    MHS a Conveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-3")
end-rule

defrule select-single-loop-layout
author thomas
explanation general conditions for the selection of an SingleLoopLayout
end-explanation
let  PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a SingleLoopLayout-6 and
    MHS a Conveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-single-loop-layout")
end-rule

defrule select-single-loop-layout-2
author thomas
explanation general conditions for the selection of an SingleLoopLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a SingleLoopLayout-6 and
    MHS a PalletConveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-single-loop-layout-2")
end-rule

defrule select-double-loop-layout
author thomas
explanation general conditions for the selection of an DoubleLoopLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a DoubleLoopLayout-12 and
    MHS a Conveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-double-loop-layout")
end-rule

defrule select-double-loop-layout-2
author thomas
explanation general conditions for the selection of an DoubleLoopLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
```



```
    TYPELAYOUT a DoubleLoopLayout-12 and
    MHS a PalletConveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-double-loop-layout")
end-rule

defrule select-linear-single-row-layout-4
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearSingleRowLayout-6 and
    MHS a PalletConveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-4")
end-rule

defrule select-linear-single-row-layout-5
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearSingleRowLayout-8 and
    MHS a TransportationCart
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-5")
end-rule

defrule select-linear-single-row-layout-6
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearSingleRowLayout-8 and
    MHS a PalletTransportationCart
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-6")
end-rule

defrule select-linear-single-row-layout-7
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
```

```
    TYPELAYOUT a LinearSingleRowLayout-5 and
    MHS a LightGantryRobot2
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-7")
end-rule

defrule select-linear-single-row-layout-8
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearSingleRowLayout-6 and
    MHS a MediumGantryRobot2
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-8")
end-rule

defrule select-linear-single-row-layout-9
author thomas
explanation general conditions for the selection of an LinearSingleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearSingleRowLayout-8 and
    MHS a HeavyGantryRobot2
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-single-row-layout-9")
end-rule

defrule select-linear-double-row-layout-1
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-12 and
    MHS a PartAGV
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-1")
end-rule

defrule select-linear-double-row-layout-2
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-12 and
    MHS a PalletAGV
if type-mhs^SELECTEDMHS = MHS and
```



```
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-2")
end-rule

defrule select-linear-double-row-layout-3
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-12 and
    MHS a Conveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-3")
end-rule

defrule select-linear-double-row-layout-4
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-12 and
    MHS a PalletConveyor
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT and
   SameEntrance/ExitPlace^CELL = SameEntrance/ExitPlace^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-4")
end-rule

defrule select-linear-double-row-layout-5
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-8 and
    MHS a TransportationCart
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL <> LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-5")
end-rule

defrule select-linear-double-row-layout-6
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-16 and
    MHS a TransportationCart
if type-mhs^SELECTEDMHS = MHS and
```

```
   CellFlexibilityDeg^CELL <> LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-6")
end-rule

defrule select-linear-double-row-layout-7
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let  PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-8 and
    MHS a PalletTransportationCart
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL <> LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-7")
end-rule

defrule select-linear-double-row-layout-8
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-16 and
    MHS a PalletTransportationCart
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL <> LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-8")
end-rule

defrule select-linear-double-row-layout-9
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-5 and
    MHS a LightGantryRobot2
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-9")
end-rule

defrule select-linear-double-row-layout-10
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-6 and
    MHS a MediumGantryRobot2
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
```



```
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-10")
end-rule

defrule select-linear-double-row-layout-11
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-8 and
    MHS a HeavyGantryRobot2
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-11")
end-rule

defrule select-linear-double-row-layout-12
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-6 and
    MHS a LightGantryRobot
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-12")
end-rule

defrule select-linear-double-row-layout-13
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-8 and
    MHS a MediumGantryRobot
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-13")
end-rule

defrule select-linear-double-row-layout-14
author thomas
explanation general conditions for the selection of an LinearDoubleRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a LinearDoubleRowLayout-8 and
    MHS a HeavyGantryRobot
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-linear-double-row-layout-14")
end-rule
```

```
defrule select-multi-row-layout-1
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-12 and
    MHS a PartAGV
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-1")
end-rule

defrule select-multi-row-layout-2
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-12 and
    MHS a PartAGV
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-2")
end-rule

defrule select-multi-row-layout-3
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-12 and
    MHS a PalletAGV
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL <= MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-3")
end-rule

defrule select-multi-row-layout-4
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-12 and
    MHS a PalletAGV
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-4")
end-rule

defrule select-multi-row-layout-5
author thomas
```



```
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-8 and
    MHS a TransportationCart
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-5")
end-rule

defrule select-multi-row-layout-6
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-8 and
    MHS a PalletTransportationCart
if type-mhs^SELECTEDMHS = MHS and
   CellFlexibilityDeg^CELL = LayoutFlexibility^TYPELAYOUT and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-6")
end-rule

defrule select-multi-row-layout-7
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-6 and
    MHS a LightGantryRobot
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-7")
end-rule

defrule select-multi-row-layout-8
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-8 and
    MHS a MediumGantryRobot
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-8")
end-rule

defrule select-multi-row-layout-9
author thomas
explanation general conditions for the selection of an MultiRowLayout
end-explanation
```

```
let PROBLEM2 a layout-problem in *layout-problem* and
    SELECTEDMHS a Selected-mhs and
    CELL the cell of PROBLEM2 and
    TYPELAYOUT a MultiRowLayout-8 and
    MHS a HeavyGantryRobot
if type-mhs^SELECTEDMHS = MHS and
   NoMachines^CELL > MaxNoMachines^TYPELAYOUT
then TYPELAYOUT & results^PROBLEM2
action $(print "Rule application: select-multi-row-layout-9")
end-rule

defrule select-layout-type-end
author thomas
explanation End of layout type selection
end-explanation
then
action $(print "End of layout type selection")
end-rule
```



```
HeavyDutyRobot
name HeavyDutyRobot-1
MaxNoMachines 5
PossibilityOfPalletHandling false
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 1.
MaxTranspWeight 50.
TranspVelocity .5
MaxProductLength 1.
FO
LightGantryRobot
name Light-Gantry-Robot-1
MaxNoMachines 12
PossibilityOfPalletHandling false
MaxProductWidth .5
MaxProductHeight .5
PositioningAccuracyMHS .09999998
MaxTranspWeight 25.
TranspVelocity 2.5
MaxProductLength .5
FO
MediumGantryRobot
name MediumGantryRobot-1
MaxNoMachines 15
PossibilityOfPalletHandling false
MaxProductWidth .5
MaxProductHeight .5
PositioningAccuracyMHS .5
MaxTranspWeight 100.
TranspVelocity 1.5
MaxProductLength .5
FO
HeavyGantryRobot
name HeavyGantryRobot-1
MaxNoMachines 21
PossibilityOfPalletHandling false
MaxProductWidth .5
MaxProductHeight .5
PositioningAccuracyMHS 1.
MaxTranspWeight 500.
TranspVelocity 1.
MaxProductLength .5
FO
ArticulatedRobot
name ArticulatedRobot-1
MaxNoMachines 5
PossibilityOfPalletHandling false
MaxProductWidth 1.5
MaxProductHeight 1.5
PositioningAccuracyMHS .09999998
MaxTranspWeight 8.
TranspVelocity 2.5
MaxProductLength 1.5
FO
SpecialHeavyGantryRobot
name SpecialHeavyGantryRobot-1
MaxNoMachines 21
PossibilityOfPalletHandling false
MaxProductWidth .5
MaxProductHeight .5
PositioningAccuracyMHS 2.
MaxTranspWeight 1000.
TranspVelocity .25
```

```
MaxProductLength .5
FO
Product
name product6
PossibilityPalletisation false
ProductLength .05
ProductWeight .2
ProductHeight .05
ProductWidth .05
QuantityOfEachProductType medium
QuantityOfEachProductType-1 (low
medium
high
)
FO
Product
name product7
PossibilityPalletisation false
ProductLength .5
ProductWeight 200.
ProductHeight .5
ProductWidth .4
QuantityOfEachProductType low
QuantityOfEachProductType-1 (low
medium
high
)
FO
PartAGV
name PartAGV-1
MaxNoMachines 10000
PossibilityOfPalletHandling false
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 1.
MaxTranspWeight 500.
TranspVelocity 1.
MaxProductLength 2.
FO
PalletAGV
name PalletAGV-1
MaxNoMachines 10000
PossibilityOfPalletHandling true
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 2.
MaxTranspWeight 500.
TranspVelocity 1.
MaxProductLength 2.
FO
PalletConveyor
name PalletConveyor-1
MaxNoMachines 10000
PossibilityOfPalletHandling true
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 2.
MaxTranspWeight 500.
TranspVelocity 1.5
MaxProductLength 2.
FO
PartConveyor
name PartConveyor-1
MaxNoMachines 10000
```



```
PossibilityOfPalletHandling false
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 2.
MaxTranspWeight 500.
TranspVelocity 1.
MaxProductLength 2.
FO
PalletHandlingRobot
name PalletHandlingRobot-1
MaxNoMachines 5
PossibilityOfPalletHandling true
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 1.
MaxTranspWeight 50.
TranspVelocity .5
MaxProductLength 1.
FO
Product
name product8
PossibilityPalletisation false
ProductLength 1.5
ProductWeight 2.5
ProductHeight .05
ProductWidth .2
QuantityOfEachProductType low
QuantityOfEachProductType-1 (low
medium
high
)
FO
ArticulatedRobot
name ArticulatedRobot-2
MaxNoMachines 5
PossibilityOfPalletHandling false
MaxProductWidth 1.5
MaxProductHeight 1.5
PositioningAccuracyMHS .09999998
MaxTranspWeight 8.
TranspVelocity 2.5
MaxProductLength 1.5
FO
LinearSingleRowLayout-5
name LinearSingleRowLayout-5-1
MaxNoMachines 5
FO
LinearSingleRowLayout-6
name LinearSingleRowLayout-6-1
MaxNoMachines 6
LayoutFlexibility low
SameEntrance/ExitPlace false
FO
LinearSingleRowLayout-8
name LinearSingleRowLayout-8-1
MaxNoMachines 8
LayoutFlexibility low
FO
LinearDoubleRowLayout-12
name LinearDoubleRowLayout-12-1
MaxNoMachines 12
LayoutFlexibility medium
SameEntrance/ExitPlace false
FO
```

```
LinearDoubleRowLayout-8
name LinearDoubleRowLayout-8-1
MaxNoMachines 8
LayoutFlexibility low
FO
LinearDoubleRowLayout-16
name LinearDoubleRowLayout-16-1
MaxNoMachines 16
LayoutFlexibility high
FO
LinearDoubleRowLayout-5
name LinearDoubleRowLayout-5-1
MaxNoMachines 5
FO
LinearDoubleRowLayout-6
name LinearDoubleRowLayout-6-1
MaxNoMachines 6
FO
MultiRowLayout-6
name MultiRowLayout-6-1
MaxNoMachines 6
FO
MultiRowLayout-8
name MultiRowLayout-8-1
MaxNoMachines 8
LayoutFlexibility high
FO
MultiRowLayout-12
name MultiRowLayout-12-1
MaxNoMachines 12
LayoutFlexibility high
FO
TransportationCart
name PartTransportationCart-1
MaxNoMachines 10000
PossibilityOfPalletHandling false
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 5.
MaxTranspWeight 3200
TranspVelocity 1.
MaxProductLength 2.
FO
TransportationCart
name HeavyPartTransportationCart-1
MaxNoMachines 10000
PossibilityOfPalletHandling false
MaxProductWidth 1.
MaxProductHeight 1.
PositioningAccuracyMHS 10.
MaxTranspWeight 10000.
TranspVelocity .3
MaxProductLength 2.
FO
PalletTransportationCart
name HeavyPalletTransportationCart-1
FO
PalletTransportationCart
name PalletTransportationCart-1
FO
DoubleLoopLayout-12
name DoubleLoopLayout-12-1
FO
SingleLoopLayout-6
```



```
name SingleLoopLayout-6-1
FO
S.RuleBase
name solve2-rb
rule-list (select-layout-type-start
select-circular-single-row-layout
select-circular-single-row-layout-2
select-circular-single-row-layout-3
select-linear-single-row-layout
select-linear-single-row-layout-2
select-linear-single-row-layout-3
select-single-loop-layout
select-single-loop-layout-2
select-double-loop-layout
select-double-loop-layout-2
select-linear-single-row-layout-4
select-linear-single-row-layout-5
select-linear-single-row-layout-6
select-linear-single-row-layout-7
select-linear-single-row-layout-8
select-linear-single-row-layout-9
select-linear-double-row-layout-1
select-linear-double-row-layout-2
select-linear-double-row-layout-3
select-linear-double-row-layout-4
select-linear-double-row-layout-5
select-linear-double-row-layout-6
select-linear-double-row-layout-7
select-linear-double-row-layout-8
select-linear-double-row-layout-9
select-linear-double-row-layout-10
select-linear-double-row-layout-11
select-linear-double-row-layout-12
select-linear-double-row-layout-13
select-linear-double-row-layout-14
select-multi-row-layout-1
select-multi-row-layout-2
select-multi-row-layout-3
select-multi-row-layout-4
select-multi-row-layout-5
select-multi-row-layout-6
select-multi-row-layout-7
select-multi-row-layout-8
select-multi-row-layout-9
select-layout-type-end
)
instantiate first-once
firing-mode sequential
sorting-mode strictly-ordered
FO
CircularSingleRowLayout
name CircularSingleRowLayout-1
MaxNoMachines 5
FO
MaterialHandlingSystem
name MHSchosen
FO
Selected-mhs
name Selected-mhs-1
type-mhs #.(smeci-object-reference HeavyGantryRobot HeavyGantryRobot-1)
FO
S.BreadthFirstStrategy
name intra-cell-layout-strategy
max-time 100.
```

```
max-states 200
FO
mhs-anomaly
name mhs-anomaly-8
mhs-tested #.(smeci-object-reference SpecialHeavyGantryRobot SpecialHeavyGantryRobot-1)
anomaly-name max-length-of-product
anomaly-locality #.(smeci-object-reference Product product8)
FO
Product
name product9
PossibilityPalletisation false
ProductLength .5
ProductWeight 200.
ProductHeight .5
ProductWidth .4
QuantityOfEachProductType low
QuantityOfEachProductType-1 (low
medium
high
)
FO
mhs-anomaly
name mhs-anomaly-1
mhs-tested #.(smeci-object-reference ArticulatedRobot ArticulatedRobot-2)
anomaly-name max-weight-of-product
anomaly-locality #.(smeci-object-reference Product product9)
FO
mhs-anomaly
name mhs-anomaly-2
mhs-tested #.(smeci-object-reference HeavyDutyRobot HeavyDutyRobot-1)
anomaly-name max-weight-of-product
anomaly-locality #.(smeci-object-reference Product product9)
FO
mhs-anomaly
name mhs-anomaly-3
mhs-tested #.(smeci-object-reference LightGantryRobot Light-Gantry-Robot-1)
anomaly-name max-weight-of-product
anomaly-locality #.(smeci-object-reference Product product9)
FO
mhs-anomaly
name mhs-anomaly-4
mhs-tested #.(smeci-object-reference MediumGantryRobot MediumGantryRobot-1)
anomaly-name max-weight-of-product
anomaly-locality #.(smeci-object-reference Product product9)
FO
mhs-anomaly
name mhs-anomaly-5
mhs-tested #.(smeci-object-reference PalletConveyor PalletConveyor-1)
anomaly-name possibility-of-palletisation-of-product
anomaly-locality #.(smeci-object-reference Product product9)
FO
mhs-anomaly
name mhs-anomaly-6
mhs-tested #.(smeci-object-reference PalletAGV PalletAGV-1)
anomaly-name possibility-of-palletisation-of-product
anomaly-locality #.(smeci-object-reference Product product9)
FO
mhs-anomaly
name mhs-anomaly-7
mhs-tested #.(smeci-object-reference SpecialHeavyGantryRobot SpecialHeavyGantryRobot-1)
anomaly-name prefered-transportation-velocity-special-heavy-gantry-robot
anomaly-locality #.(smeci-object-reference Cell cell_cara_1)
FO
Machine
```



```
name Machine-5
MachineLength 1.
MachineHeigth .5
MachineWidth 3.
MachPositioningAccuracy 3.
MachineWeight 1200.
FO
mhs-problem
name g115
product #.(smeci-object-reference Product product9)
cell #.(smeci-object-reference Cell cell_cara_1)
results (#.(smeci-object-reference HeavyGantryRobot HeavyGantryRobot-1)
)
FO
S.BreadthFirstStrategy
name intra-cell-mhs-strategy
max-time 100.
max-states 200
FO
Machine
name Machine-75
MachineLength 2.
MachineHeigth 1.5
MachineWidth 2.
MachPositioningAccuracy 2.
MachineWeight 800.
FO
mhs-problem
name g116
product #.(smeci-object-reference Product product9)
cell #.(smeci-object-reference Cell cell_cara_1)
results (#.(smeci-object-reference HeavyGantryRobot HeavyGantryRobot-1)
)
FO
mhs-problem
name g117
product #.(smeci-object-reference Product product9)
cell #.(smeci-object-reference Cell cell_cara_1)
results (#.(smeci-object-reference HeavyGantryRobot HeavyGantryRobot-1)
)
machine #.(smeci-object-reference Machine Machine-75)
FO
S.Task
name intra-cell-mhs-task
rule-base #.(smeci-object-reference S.RuleBase solve-rb)
continuation-task #.(smeci-object-reference S.SystemTask pop-task)
FO
S.TaskAgenda
name intra-cell-mhs-agenda
initial-task #.(smeci-object-reference S.Task intra-cell-mhs-task)
FO
layout-problem
name g118
machine #.(smeci-object-reference Machine Machine-75)
product #.(smeci-object-reference Product product9)
cell #.(smeci-object-reference Cell cell_cara_1)
results (#.(smeci-object-reference LinearDoubleRowLayout-8 LinearDoubleRowLayout-8-1)
)
selected-mhs #.(smeci-object-reference Selected-mhs Selected-mhs-1)
FO
S.Task
name intra-cell-layout-task
rule-base #.(smeci-object-reference S.RuleBase solve2-rb)
continuation-task #.(smeci-object-reference S.SystemTask pop-task)
```

```
FO
S.TaskAgenda
name intra-cell-layout-agenda
initial-task #.(smeci-object-reference S.Task intra-cell-layout-task)
FO
MultiRowLayout-20
name MultiRowLayout-20-1
MaxNoMachines 20
LayoutFlexibility high
FO
```

2. AGENCEMENT ET EVALUATION

2.1 Listings des programmes de la partie agencement et évaluation

1. Le programme exécutable

```
intracell:ag_intra_cell.o acqdon.o tradon.o smp_lin.o smp_cir.o dbl_lin.o\
 mul_lin.o free_lin.o contraintes.o
        cc -g -o intracell ag_intra_cell.o acqdon.o tradon.o smp_lin.o\
 smp_cir.o dbl_lin.o mul_lin.o free_lin.o contraintes.o -lm -lg
ag_intra_cell.o:ag_intra_cell.c def.h
        cc -c -g ag_intra_cell.c
acqdon.o:acqdon.c def.h
        cc -c -g acqdon.c
tradon.o:tradon.c def.h
        cc -c -g tradon.c
smp_lin.o:smp_lin.c def.h
        cc -c -g smp_lin.c
smp_cir.o:smp_cir.c def.h
        cc -c -g smp_cir.c
dbl_lin.o:dbl_lin.c def.h
        cc -c -g dbl_lin.c
mul_lin.o:mul_lin.c def.h
        cc -c -g mul_lin.c
free_lin.o:free_lin.c def.h
        cc -c -g free_lin.c
contraintes.o:contraintes.c def.h
        cc -c -g contraintes.c
```

```
/* structure de contraintes */
/* ************************ */
struct contr {
        int M1;
        int M2;
        int type;
        float Dmami;
        float poids;
};
typedef struct contr contrainte;

/* Constants */
/* ********* */
#define NB_PROD_MAX 50
#define NB_MACH_MAX 50
#define NB_LIGN_MAX 20
#define NB_M/LN_MAX 10
```

2. Le programme principal

```
                    /* ********************************************************** */
                    /* *    PROGRAMME PRINCIPAL D'AGENCEMENT INTRA-CELLULAIRE    * */
                    /* *    ------------------------------------------------     * */
                    /* *                      T. HAMANN                          * */
                    /* ********************************************************** */


#include <stdio.h>
#include "def.h"

int type;

main()
{
        int     rep;
        char    ans;

        void    acq_dat(),      /* acquisition des donnees */
                trait_dat(),    /* traitement des donnees */
                sim_lig_lin(),  /* agencement en simple ligne lineaire */
                sim_lig_cir(),  /* agencement en simple ligne circulaire */
                dou_lig_lin(),  /* agencement en double ligne lineaire */
                mul_lig(),      /* agencement en multi-lignes */
                free_lig();     /* agencement en configuration libre */

        printf ("\f\t\t\t ~~~   AGENCEMENT INTRA-CELLULAIRE   ~~~");

        do {
                printf ("\n\n\n\n\n\t\t    Acquisition des donnees general");
                acq_dat();

                printf ("\n\n\n\t\t     Traitement des donnees ");
                trait_dat();

                printf ("\f\n\n\n\n\t\t\t     ****      M E N U    P R I N C I P A L     ****");
                printf ("\n\n\n\t\t   1.  Configuration en simple ligne lineaire");
                printf ("\n\n\n\t\t   2.  Configuration en simple ligne circulaire");
                printf ("\n\n\n\t\t   3.  Configuration en double ligne lineaire");
                printf ("\n\n\n\t\t   4.  Configuration en multi-lignes");
                printf ("\n\n\n\t\t   5.  Configuration libre");
                printf ("\n\n\n\n\t\t\t FAITE VOTRE CHOIX : ");
                do {
                        scanf ("%d", &rep);
                        if ((rep <= 0) || (rep > 5))    printf ("\n\n Donnez une autre reponse entre 1 et 5 : ");
                }
                while ((rep <= 0) || (rep > 5));

                if ( rep == 1 )         { type = 1; sim_lig_lin(); }
                else if (rep == 2)      { type = 2; sim_lig_cir(); }
                else if (rep == 3)      { type = 3; dou_lig_lin(); }
                else if (rep == 4)      { type = 4; mul_lig(); }
                else if (rep == 5)      { type = 5; free_lig(); }

                printf ("\n\n\n Voulez vous construir une autre cellule o/n : ");
                scanf ("%s", &ans);
        }
        while ( ans == 'o' );
}
```

3. Programme d'acquisition des données globales

```
/* ************************************************************ */
/* *     Sous-programme d'acquisition des donnees generales     * */
/* *     ---------------------------------------------------    * */
/* *                       par T. HAMANN                        * */
/* ************************************************************ */


#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include "def.h"

        int     Q[500],         /* Quantite a produire */
                U[500],         /* Taille du lot transporte */
                M[100],         /* Nombre de machines du routage */
                Ind[50],        /* Indices des machines */
                cont[50][50],   /* Type de contrainte */
                nb_mach,        /* Nombre de machine */
                nb_prod,        /* Nombre de produits */
                M_def,          /* Indice montrant l'existance de machines a placer definitivement */
                nb_m_def,       /* Nombre de machines a placer definitivement */
                Mdef[100],      /* Indice d'une machine a placer definitivement */
                Sdef[100];      /* Indice du site de la machine a placer definitivement */


        float   dis[50][50],    /* Distance inter machine */
                poids[50][50],  /* Poids contrainte */
                dist_min,       /* Distance inter_machines mini. */
                pm[100],        /* Poids machine */
                wm[100],        /* Largeur machine */
                lm[100];        /* Longueur machine */

        float   F[50][50];      /* Matrice des flux */



acq_dat()
{
        int     i, j, rep, a, b, e, g, repi;
        float   dist_inter_mach(),
                QsU, f[50][50];

        FILE    *pt_fich, *fopen(), *fclose();
        char    pt_bin[200], nom_fich[15], rad;


        printf ("\f\n\n\t\t\t\t ~~~ **** ACQUISITION DES DONNEES GENERALES **** ~~~");

        printf ("\n\n\n\n\t\t\t\t A. DONNEES PRODUITS");
        printf ("\n\t\t\t\t    ~~~~~~~~~~~~~~~~");

        printf ("\n\n\n\t\t Vous pouvez : ");
        printf ("\n\n\t\t 1. Donnez la matrice des flux inter-machines");
        printf ("\n\n\t\t 2. La faire calculer par le systeme");
        printf ("\n\n\t\t Faite votre choix '1 ou 2' : ");
        do {
                scanf("%d", &repi);
                if ( repi != 1 && repi != 2 )   printf ("\n\n\t\t Repondez par 1 ou 2 : ");
        }
        while ( repi != 1 && repi != 2 );
        if ( repi == 1 ) {
                printf ("\n\n\n\t\t Vous pouvez : ");
                printf ("\n\n\t\t\t 1. Donner la matrice manuellement");
                printf ("\n\n\t\t\t 2. Prendre la matrice dans un fichier");
                printf ("\n\n\t\t Faite votre choix '1 ou 2' : ");
                do {
                        scanf("%d", &rep);
                        if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
                }
                while ( rep != 1 && rep != 2 );
                if ( rep == 1 ) {
                        printf ("\n\n\n\t\t Donnez le nom du fichier de sauvegarde de la matrice 'Fmat...' : ");
                        scanf ("%s", nom_fich);
                        pt_fich = fopen(nom_fich, "w");

                        printf ("\n\n\n\t\t Donnez le nombre de machines de cette cellules : ");
                        scanf ("%d", &nb_mach);
                        fprintf (pt_fich, "%d\n", nb_mach);
                        for (i = 0; i < nb_mach; i++) {
                                for (j = 0; j < nb_mach; j++) {
                                        if ( i != j ) {
                                                printf ("\n\n Y a t-il un flux entre les machines %d et %d 'o ou n': ", i+1, j+1);
                                                do {
                                                        scanf ("%s", &rad);
                                                        if ( (rad != 'o') && (rad != 'n') )
                                                                printf ("\n\n\t\t repondez par 'o' ou 'n' : ");
                                                }
                                                while ( (rad != 'o') && (rad != 'n') );
                                                if (rad == 'o') {
                                                        printf ("\n\n\t\t Donnez la valeure du flux entre M%d et M%d : ", i+1, j+1);
                                                        scanf ("%f", &f[i][j]);
                                                }
                                                else if (rad == 'n')
                                                        f[i][j] = 0.0;
                                        }
                                        else    f[i][j] = 0.0;
                                        fprintf (pt_fich, "%f\n", f[i][j]);

                                }
                        }
```

```
                fclose(pt_fich);
        }
        else if (rep == 2) {
                printf ("\n\n\n\t\t Donnez le nom du fichier contenant la matrice des flux 'Fmat...' : ");
                scanf ("%s", nom_fich);
                pt_fich = fopen(nom_fich, "r");

                fscanf (pt_fich, "%d", &nb_mach);
                for (i = 0; i < nb_mach; i++) {
                        for (j = 0; j < nb_mach; j++) {
                                fscanf (pt_fich, "%f", &f[i][j]);
                        }
                }
                fclose(pt_fich);
        }

        /* Visualisation de la matrice des flux */
        printf ("\n\n\n\t\t MATRICE DES FLUX INTER-MACHINES \n\n");
        for ( i = 0; i < nb_mach; i++) {
                printf("\n\t\t\t");
                for (j = 0; j < nb_mach; j++) {
                        F[i][j] = f[i][j];
                        printf ("%f   ", F[i][j]);
                }
        }
}

else if (repi == 2) {

        printf ("\n\n\n\t\t Vous pouvez : ");
        printf ("\n\n\t\t\t 1. Fournir les donnees manuellement");
        printf ("\n\n\t\t\t 2. Prendre les donnees dans un fichier");
        printf ("\n\n\t\t Faite votre choix '1 ou 2' : ");
        do {
                scanf("%d", &rep);
                if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
        }
        while ( rep != 1 && rep != 2 );
        if ( rep == 1 ) {
                printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'dp...' : ");
                scanf ("%s", nom_fich);
                pt_fich = fopen(nom_fich, "w");

                printf ("\n\n\n\t\t Donnez le nombre de types de produits a fabriquer dans cette cellules : ");
                scanf ("%d", &nb_prod);
                fprintf (pt_fich, "Nombre_de_types_de_produit_a_fabriquer_: ");
                fprintf (pt_fich, "%d\n", nb_prod);

                printf ("\n\n\n\t\t Donnez le nombre de machines composant cette cellules : ");
                scanf ("%d", &nb_mach);
                fprintf (pt_fich, "Nombre_de_machines_dans_la_cellule_: ");
                fprintf (pt_fich, "%d\n", nb_mach);

                /* Initialisation de la matrice des flux */
                for (i = 0; i < nb_mach; i++) {
                        for (j = 0; j < nb_mach; j++)
                                f[i][j] = 0.0;
                }

                for (i = 0; i < nb_prod; i++) {
                        printf ("\n\n\t Donnez la quantite de produits de type %d a produire durant l'horizon choisie : ", i+1);
                        scanf ("%d", &Q[i]);
                        fprintf (pt_fich,"Quantite_a_produire_: ");
                        fprintf (pt_fich, "%d\n", Q[i]);
                        printf ("\n\n\t Donnez la taille du lot de produit du type %d : ", i+1);
                        scanf ("%d", &U[i]);
                        fprintf (pt_fich,"Taille_du_lot_: ");
                        fprintf (pt_fich,"%d\n", U[i]);
                        QsU = (float)Q[i] / (float)U[i];
                        printf ("\n\n\t Donnez le nombre de machines du routage des produits du type %d : ", i+1);
                        scanf ("%d", &M[i]);
                        fprintf (pt_fich,"Nombre_de_machines_du_routage_: ");
                        fprintf (pt_fich,"%d\n\n", M[i]);
                        if (i > 0) {
                                for (j = 0; j < M[i-1]; j++)
                                        Ind[j] = -1;
                        }
                        for (j = 0; j < M[i]; j++) {
                                printf ("\n\n\t\t Donnez l'indice de la machine %d du routage %d : ", j+1, i+1);
                                scanf ("%d", &Ind[j]);
                                Ind[j]--;
                                fprintf (pt_fich,"Indice_de_la_machine_%d_du_routage_: ", j+1);
                                fprintf (pt_fich,"%d\n\n", Ind[j]);
                                if (j >= 1) {
                                        a = Ind[j-1];   b = Ind[j];
                                        f[a][b] = f[a][b] + QsU;
                                }
                        }
                        printf ("\n\n\t\t\t\t **********************************************\n\n");
                }
                printf ("\n\n\n\t\t\t MATRICE DES FLUX INTER-MACHINES \n");
                printf ("\t\t\t ---------------------------------- \n");

                for (e = 0; e < nb_mach; e++) {
                        printf ("\n\t\t\t\t");
                        for (g = 0; g < nb_mach; g++) {
                                F[e][g] = f[e][g];
                                printf ("%5f  ",F[e][g]);
                        }
                }
                fclose(pt_fich);
```

```
            printf ("\n\n\n\n\t\t\t\t ##################################################################\n\n");
            }
            else {
                    printf ("\n\n\n\t\t Donnez le nom du fichier des donnees 'dp...' : ");
                    scanf ("%s", nom_fich);
                    pt_fich = fopen(nom_fich, "r");
                    fscanf (pt_fich, "%s", pt_bin);
                    fscanf (pt_fich, "%d", &nb_prod);

                    fscanf (pt_fich, "%s", pt_bin);
                    fscanf (pt_fich, "%d", &nb_mach);

                    /* Initialisation de la matrice des flux */
                    for (i = 0; i < nb_mach; i++) {
                            for (j = 0; j < nb_mach; j++)
                                    f[i][j] = 0.0;
                    }

                    for (i = 0; i < nb_prod; i++) {
                            fscanf (pt_fich, "%s", pt_bin);
                            fscanf (pt_fich, "%d", &Q[i]);
                            fscanf (pt_fich, "%s", pt_bin);
                            fscanf (pt_fich, "%d", &U[i]);
                            fscanf (pt_fich, "%s", pt_bin);
                            QsU = (float)Q[i] / (float)U[i];
                            fscanf (pt_fich, "%d", &M[i]);

                            if (i > 0) {
                                    for (j = 0; j < M[i-1]; j++)
                                            Ind[j] = -1;
                            }

                            for (j = 0; j < M[i]; j++) {
                                    fscanf (pt_fich, "%s", pt_bin);
                                    fscanf (pt_fich, "%d", &Ind[j]);
                                    if (j >= 1) {
                                            a = Ind[j-1];   b = Ind[j];
                                            f[a][b] = f[a][b] + QsU;
                                    }
                            }
                    }

                    printf ("\n\n\n\t\t\t MATRICE DES FLUX INTER-MACHINES \n");
                    printf ("\t\t\t ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \n");

                    for (e = 0; e < nb_mach; e++) {
                    printf ("\n\t\t\t\t");
                            for (g = 0; g < nb_mach; g++) {
                                    F[e][g] = f[e][g];
                                    printf ("%f  ",F[e][g]);
                            }
                    }

                    fclose(pt_fich);

            }
    }

    printf ("\n\n\n\t\t\t\t B. DONNEES MACHINES");
    printf ("\n\t\t\t\t    ~~~~~~~~~~~~~~~~");
    printf ("\n\n\n\t\t Vous pouvez : ");
    printf ("\n\n\t\t\t 1. Fournir les donnees manuellement");
    printf ("\n\n\t\t\t 2. Prendre les donnees dans un fichier");
    printf ("\n\n\t\t Faite votre choix '1 ou 2' : ");
    do {
            scanf("%d", &rep);
            if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
    }
    while ( rep != 1 && rep != 2 );
    if ( rep == 1 ) {
            printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'dm...' : ");
            scanf ("%s", nom_fich);
            pt_fich = fopen(nom_fich, "w");

            for (i = 0; i < nb_mach; i++) {
                    printf ("\n\n\t\t REMARQUE : La longueur de la machine est sa dimension le long la ligne.");
                    printf ("\n\n\n\t\t\t Donnez la longueur de la machine %d : ", i+1);
                    scanf ("%f", &lm[i]);
                    fprintf (pt_fich, "Longueur_machine_%d_: ", i+1);
                    fprintf (pt_fich, "%f\n", lm[i]);

                    printf ("\n\n\t\t\t Donnez la largeure de la machine %d : ", i+1);
                    scanf ("%f", &wm[i]);
                    fprintf (pt_fich, "Largeur_machine_%d_: ", i+1);
                    fprintf (pt_fich, "%f\n", wm[i]);

                    printf ("\n\n\t\t\t Donnez le poid de la machine %d : ", i+1);
                    scanf ("%f", &pm[i]);
                    fprintf (pt_fich, "Poid_machine_%d_: ", i+1);
                    fprintf (pt_fich, "%f\n", pm[i]);

            }
            printf ("\n\n\t\t\t Donnez la distance inter-machines minimale : ");
            scanf ("%f", &dist_min);
            fprintf (pt_fich, "Distance_inter_machines_minimale_: ");
            fprintf (pt_fich, "%f\n", dist_min);

            fclose(pt_fich);
    }
    else if (rep == 2) {
            printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'dm...' : ");
```

```
            scanf ("%s", nom_fich);
            pt_fich = fopen(nom_fich, "r");

            for (i = 0; i < nb_mach; i++) {
                    fscanf (pt_fich, "%s", pt_bin);
                    fscanf (pt_fich, "%f", &lm[i]);

                    fscanf (pt_fich, "%s", pt_bin);
                    fscanf (pt_fich, "%f", &wm[i]);

                    fscanf (pt_fich, "%s", pt_bin);
                    fscanf (pt_fich, "%f", &pm[i]);
            }
            fscanf (pt_fich, "%s", pt_bin);
            fscanf (pt_fich, "%f", &dist_min);

            fclose(pt_fich);
        }
}
```

4. Programme de traitement des données

```
                                        /* ************************************************** */
                                        /* *    Sous-programme de traitement de donnees     * */
                                        /* *    ---------------------------------------     * */
                                        /* *                  par T. HAMANN                 * */
                                        /* ************************************************** */

#include <stdio.h>
#include <math.h>
#include "def.h"

extern int      Q[500], /* Quantite a produire */
                U[500], /* Taille du lot transporte */
                M[100],         /* Nombre de machines du routage */
                Ind[100][100],  /* Indices des machines */
                cont[100][100], /* Type de contrainte */
                nb_mach,        /* Nombre de machine */
                nb_prod;        /* Nombre de produits */

extern float    dis[100][100],  /* Distance inter machine */
                F[50][50],      /* Matrice des flux */
                pm[100],        /* Poids machine */
                wm[100],        /* Largeur machine */
                lm[100];        /* Longueur machine */

float           lmax, wmax, Fmax;

trait_dat()
{
        int     i, j;
        float   lmax, wmax;

        /* Determination de la machine de plus grande longueur */
        lmax = 0;
        for (i = 0; i < nb_mach; i++) {
                if (lm[i] > lmax)       lmax = lm[i];
        }

        /* Determination de la machine de plus grande largeur */
        wmax = 0;
        for (i = 0; i < nb_mach; i++) {
                if (wm[i] > wmax)       wmax = wm[i];
        }

        /* Calcul du plus grand flux */
        Fmax = 0;
        for (i = 0; i < nb_mach; i++) {
                for (j = 0; j < nb_mach; j++)
                        if ( F[i][j] > Fmax ) Fmax = F[i][j];
        }
}
```

5. Programme d'acquisition des contraintes

```
                        /* ******************************************************* */
                        /* *    Sous-programme d'acquisition des contraintes     * */
                        /* *    ____________________________________________     * */
                        /* *                    par T. HAMANN                    * */
                        /* ******************************************************* */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include "def.h"

extern int      type,           /* Type de la configuration */
                nb_ligne,       /* Nombre de lignes dans le cas d'un multi-ligne */
                nb_mach;        /* Nombre de machine */

        int     pres_cont,      /* Indique la presence des contraintes */
                Couples,        /* Indique la presence des contraintes de positionnement relatif */
                M_def,          /* Indique l'existance de machines a placer definitivement */
                nb_m_def,       /* Nombre de machines a placer definitivement */
                nb_couple,      /* Nombre de couples de machines soumises a un positionnement relatif */
                Mdef[50],       /* Indice d'une machine a placer definitivement */
                Ldef[50],       /* Indice de la ligne contenant une machine */
                Sdef[50];       /* Indice du site de la machine a placer definitivement */

contrainte      Cont[50];       /* Contrainte de positionnement relatif entre les cellules */

void cont_pos()
{
        int     i, rep, nb_lin;

        char    nom_fich[15], repo, repi;

        FILE    *pt_fich, *fopen(), *fclose();

        printf ("\n\n\n\t LES CONTRAINTES");
        printf ("\n\t ***************");

        printf ("\n\n\t\t Y a t-il des contraintes dans ce systeme (o/n) ?");
        do {
                scanf("%s", &repi);
                if (repi != 'o' && repi != 'n')         printf ("\n\n\t\t Repondez par o ou n : ");
        }
        while (repi != 'o' && repi != 'n');

        /* Cas ou il y a des contraintes */
        /* ***************************** */
        if (repi == 'o') {
            pres_cont = 1;

            /* Cas du simple ligne lineaire ou circulaire ou d'une configuration libre */
            /* ************************************************************************ */
            if ( (type == 1) || (type == 2) || (type == 5) ) {
                printf ("\n\n\n\t\t Vous pouvez : ");
                printf ("\n\n\t\t\t 1. Fournir les contraintes manuellement");
                printf ("\n\n\t\t\t 2. Prendre les contraintes dans un fichier");
                printf ("\n\n\t\t Faite votre choix '1 ou 2' : ");
                do {
                        scanf("%d", &rep);
                        if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
                }
                while ( rep != 1 && rep != 2 );
                if ( rep == 1 ) {

                        printf ("\n\n\t\t\t CONTRAINTES DE POSITIONNEMENT RELATIF ENTRE LES MACHINES\n\n");

                        printf ("\n\n\t Y a t-ils des contraintes de positionnement relatif (o/n) ?");
                        do {
                                scanf ("%s", &repo);
                                if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n' : ");
                        }
                        while (repo != 'o' && repo != 'n');
                        if (repo == 'o') {
                                Couples = 1;
                                nb_couple = 0;
                                do {
                                        printf ("\n\n\t Donnez l'indice de la premiere machine : ");
                                        scanf ("%d", &Cont[nb_couple].M1); Cont[nb_couple].M1--;
                                        printf ("\n\n\t Donnez l'indice de la deuxieme machine : ");
                                        scanf ("%d", &Cont[nb_couple].M2); Cont[nb_couple].M2--;
                                        Cont[nb_couple + 1].M1 = Cont[nb_couple].M2;
                                        Cont[nb_couple + 1].M2 = Cont[nb_couple].M1;
                                        printf ("\n\n\t\t  La contrainte, est elle : \n\n\t\t\t 1. de RAPROCHEMENT");
                                        printf ("\n\n\t\t\t 2. ou d'ELOIGNEMENT\n\n\t\t Repondez par 1 ou 2 : ");
                                        do {
                                                scanf ("%d", &rep);
                                                if ( rep != 1 && rep != 2 ) { printf ("\n\n\t\t Repondez par 1 ou 2 : "); }
                                        }
                                        while ( rep != 1 && rep != 2 );
                                        if (rep == 1) {
                                                Cont[nb_couple].type = Cont[nb_couple + 1].type = 1;
                                                printf ("\n\n\t\t Donnez la distance maximale : ");
                                                scanf ("%f", &Cont[nb_couple].Dmami);
                                                Cont[nb_couple + 1].Dmami = Cont[nb_couple].Dmami;
                                                printf ("\n\n\t\t Donnez le poids de la contrainte : ");
                                                scanf ("%f", &Cont[nb_couple].poids);
                                                Cont[nb_couple + 1].poids = Cont[nb_couple].poids;
                                        }
                                        else if (rep == 2) {
                                                Cont[nb_couple].type = Cont[nb_couple + 1].type = 2;
                                                printf ("\n\n\t\t Donnez la distance minimale : ");
                                                scanf ("%f", &Cont[nb_couple].Dmami);
```

```
                        Cont[nb_couple + 1].Dmami = Cont[nb_couple].Dmami;
                        printf ("\n\n\t\t Donnez le poids de la contrainte : ");
                        scanf ("%f", &Cont[nb_couple].poids);
                        Cont[nb_couple + 1].poids = Cont[nb_couple].poids;
                }
                nb_couple = nb_couple + 2;

                printf ("\n\n\n\n\t Y a t-il un autre couple de machines soumises a une contrainte (o/n) ?");
                do {
                        scanf ("%s", &repo);
                        if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n': ");
                }
                while (repo != 'o' && repo != 'n');
        }
        while (repo == 'o');

        /* Sauvegarde des couples dans un fichier */
        /* ************************************** */
        printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'cont...' : ");
        scanf ("%s", nom_fich);
        pt_fich = fopen(nom_fich, "w");
        fprintf (pt_fich, "\n%d", Couples);
        fprintf (pt_fich, "\n%d", nb_couple);
        for (i = 0; i < nb_couple; i++) {
                fprintf (pt_fich, "\n%d", Cont[i].M1);
                fprintf (pt_fich, "\n%d", Cont[i].M2);
                fprintf (pt_fich, "\n%d", Cont[i].type);
                fprintf (pt_fich, "\n%f", Cont[i].poids);
                fprintf (pt_fich, "\n%f", Cont[i].Dmami);
        }
}
else if (repo == 'n') {
        Couples = 0;
        fprintf (pt_fich, "\n%d", Couples);
}

printf ("\n\n\t CONTRAINTES DE POSITIONNEMENT DEFINITIF DES MACHINES\n\n");

printf ("\n\n\t\t Y-a t-il des machines a positionnez definitivement sur un site 'o/n' : ");
do {
        scanf ("%s", &repo);
        if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n' : ");
}
while (repo != 'o' && repo != 'n');
if (repo == 'o') {
        M_def = 1;      /* Indique qu'il y a des machines a position definitives */
        nb_m_def = 0;
        do {
                i = nb_m_def;
                printf ("\n\n\t Donnez l'indice de la %deme machine a placer definitivement : ", i + 1);
                do {
                        scanf ("%d", &Mdef[i]);
                        if (Mdef[i] < 1 || Mdef[i] > nb_mach) printf ("\n\n\t Donnez une autre reponse : ");
                }
                while (Mdef[i] < 1 || Mdef[i] > nb_mach);
                Mdef[i] = Mdef[i] - 1;

                printf ("\n\n\t Donnez l'indice du site de la %deme machine a placer definitivement : ", i + 1);
                do {
                        scanf ("%d", &Sdef[Mdef[i]]);
                        if (Sdef[Mdef[i]] < 1 || Sdef[Mdef[i]] > nb_mach)
                                printf ("\n\n\t Donnez une autre reponse : ");
                }
                while (Sdef[Mdef[i]] < 1 || Sdef[Mdef[i]] > nb_mach);
                Sdef[Mdef[i]] = Sdef[Mdef[i]] - 1;
                fprintf (pt_fich, "\n%d", Sdef[Mdef[i]]);

                nb_m_def++;
                printf ("\n\n\n\t Y a t-il une autre machine a affecter definitivement (o/n) ?");
                do {
                        scanf ("%s", &repo);
                        if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n': ");
                }
                while (repo != 'o' && repo != 'n');
        }
        while (repo == 'o');
        fprintf (pt_fich, "\n%d", M_def);
        fprintf (pt_fich, "\n%d", nb_m_def);
        for (i = 0; i < nb_m_def; i++) {
                fprintf (pt_fich, "\n%d", Mdef[i]);
                fprintf (pt_fich, "\n%d", Sdef[Mdef[i]]);
        }
}
else {
        M_def = 0;
        fprintf (pt_fich, "\n%d", M_def);
}
fclose(pt_fich);
}
else if (rep == 2) {
        printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'cont...' : ");
        scanf ("%s", nom_fich);
        pt_fich = fopen(nom_fich, "r");
        fscanf (pt_fich, "%d", &Couples);
        if ( Couples == 1 ) {
                fscanf (pt_fich, "%d", &nb_couple);
                for ( i = 0; i < nb_couple; i++) {
                        fscanf (pt_fich, "%d", &Cont[i].M1);
                        fscanf (pt_fich, "%d", &Cont[i].M2);
                        fscanf (pt_fich, "%d", &Cont[i].type);
                        fscanf (pt_fich, "%f", &Cont[i].poids);
```

```
                        fscanf (pt_fich, "%f", &Cont[i].Dmami);
                    }
                }
                fscanf (pt_fich, "%d", &M_def);
                if (M_def == 1) {
                        fscanf (pt_fich, "%d", &nb_m_def);
                        for (i = 0; i < nb_m_def; i++) {
                                fscanf (pt_fich, "%d", &Mdef[i]);
                                fscanf (pt_fich, "%d", &Sdef[Mdef[i]]);
                        }
                }
                fclose(pt_fich);
        }
    }

    /* Cas du double et du multi-ligne */
    /* ******************************* */

    else if ( (type == 3) || (type == 4) ) {

        printf ("\n\n\n\t\t Vous pouvez : ");
        printf ("\n\n\t\t\t 1. Fournir les contraintes manuellement");
        printf ("\n\n\t\t\t 2. Prendre les contraintes dans un fichier");
        printf ("\n\n\t\t Faite votre choix '1 ou 2' : ");
        do {
                scanf("%d", &rep);
                if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
        }
        while ( rep != 1 && rep != 2 );
        if ( rep == 1 ) {

                printf ("\n\n\t\t\t CONTRAINTES DE POSITIONNEMENT RELATIF ENTRE LES MACHINES\n\n");

                printf ("\n\n\t Y a t-ils des contraintes de positionnement relatif (o/n) ?");
                do {
                        scanf ("%s", &repo);
                        if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n': ");
                }
                while (repo != 'o' && repo != 'n');
                if (repo == 'o') {
                        Couples = 1;
                        nb_couple = 0;
                        do {
                                printf ("\n\n\t Donnez l'indice de la premiere machine : ");
                                scanf ("%d", &Cont[nb_couple].M1); Cont[nb_couple].M1--;
                                printf ("\n\n\t Donnez l'indice de la deuxieme machine : ");
                                scanf ("%d", &Cont[nb_couple].M2); Cont[nb_couple].M2--;
                                Cont[nb_couple + 1].M1 = Cont[nb_couple].M2;
                                Cont[nb_couple + 1].M2 = Cont[nb_couple].M1;
                                printf ("\n\n\t\t  La contrainte, est elle : \n\n\t\t\t 1. de RAPROCHEMENT");
                                printf ("\n\n\t\t\t 2. ou d'ELOIGNEMENT\n\n\t\t Repondez par 1 ou 2 : ");
                                do {
                                        scanf ("%d", &rep);
                                        if ( rep != 1 && rep != 2 ) { printf ("\n\n\t\t Repondez par 1 ou 2 : "); }
                                }
                                while ( rep != 1 && rep != 2 );
                                if (rep == 1) {
                                        Cont[nb_couple].type = Cont[nb_couple + 1].type = 1;
                                        printf ("\n\n\t\t Donnez la distance maximale : ");
                                        scanf ("%f", &Cont[nb_couple].Dmami);
                                        Cont[nb_couple + 1].Dmami = Cont[nb_couple].Dmami;
                                        printf ("\n\n\t\t Donnez le poids de la contrainte : ");
                                        scanf ("%f", &Cont[nb_couple].poids);
                                        Cont[nb_couple + 1].poids = Cont[nb_couple].poids;
                                }
                                else if (rep == 2) {
                                        Cont[nb_couple].type = Cont[nb_couple + 1].type = 2;
                                        printf ("\n\n\t\t Donnez la distance minimale : ");
                                        scanf ("%f", &Cont[nb_couple].Dmami);
                                        Cont[nb_couple + 1].Dmami = Cont[nb_couple].Dmami;
                                        printf ("\n\n\t\t Donnez le poids de la contrainte : ");
                                        scanf ("%f", &Cont[nb_couple].poids);
                                        Cont[nb_couple + 1].poids = Cont[nb_couple].poids;
                                }
                                nb_couple = nb_couple + 2;

                                printf ("\n\n\n\n\t Y a t-il un autre couple de machines soumises a une contrainte (o/n) ?");
                                do {
                                        scanf ("%s", &repo);
                                        if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n': ");
                                }
                                while (repo != 'o' && repo != 'n');
                        }
                        while (repo == 'o');

                        /* Sauvegarde des couples dans un fichier */
                        /* ************************************** */
                        printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'cont_dm...' : ");
                        scanf ("%s", nom_fich);
                        pt_fich = fopen(nom_fich, "w");
                        fprintf (pt_fich, "\n%d", Couples);
                        fprintf (pt_fich, "\n%d", nb_couple);
                        for (i = 0; i < nb_couple; i++) {
                                fprintf (pt_fich, "\n%d", Cont[i].M1);
                                fprintf (pt_fich, "\n%d", Cont[i].M2);
                                fprintf (pt_fich, "\n%d", Cont[i].type);
                                fprintf (pt_fich, "\n%f", Cont[i].poids);
                                fprintf (pt_fich, "\n%f", Cont[i].Dmami);
                        }
                }
                else if (repo == 'n') {
```

```
                Couples = 0;
                fprintf (pt_fich, "\n%d", Couples);
        }


        /* Contraintes de positionnement definitif de certaines machines */
        /* ************************************************************* */
        if ( type == 3 )        nb_lin = 2;
        else if (type == 4 )    nb_lin = nb_ligne;

        printf ("\n\n\t\t Y-a t-il des machines a positionnez definitivement sur un site 'o/n' : ");
        do {
                scanf ("%s", &repo);
                if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n' : ");
        }
        while (repo != 'o' && repo != 'n');
        if (repo == 'o') {
                M_def = 1;
                nb_m_def = 0;
                do {
                        i = nb_m_def;
                        printf ("\n\n\t Donnez l'indice de la %deme machine a placer definitivement : ", i + 1);
                        do {
                                scanf ("%d", &Mdef[i]);
                                if (Mdef[i] < 1 || Mdef[i] > nb_mach) printf ("\n\n\t Donnez une autre reponse : ");
                        }
                        while (Mdef[i] < 1 || Mdef[i] > nb_mach);
                        Mdef[i] = Mdef[i] - 1;

                        printf ("\n\n\t Donnez l'indice de la ligne de la %deme machine a placer definitivement: ", i + 1);
                        do {
                                scanf ("%d", &Ldef[Mdef[i]]);
                                if (Ldef[Mdef[i]] < 1 || Ldef[Mdef[i]] > nb_lin)
                                        printf ("\n\n\t Donnez une autre reponse : ");
                        }
                        while (Ldef[Mdef[i]] < 1 || Ldef[Mdef[i]] > nb_lin);
                        Ldef[Mdef[i]] = Ldef[Mdef[i]] - 1;

                        printf ("\n\n\t Donnez l'indice du site de la %deme machine a placer definitivement : ", i + 1);
                        do {
                                scanf ("%d", &Sdef[Mdef[i]]);
                                if (Sdef[Mdef[i]] < 1 || Sdef[Mdef[i]] > nb_mach)
                                        printf ("\n\n\t Donnez une autre reponse : ");
                        }
                        while (Sdef[Mdef[i]] < 1 || Sdef[Mdef[i]] > nb_mach);
                        Sdef[Mdef[i]] = Sdef[Mdef[i]] - 1;

                        nb_m_def++;
                        printf ("\n\n\n\t Y a t-il une autre machine a affecter definitivement (o/n) ?");
                        do {
                                scanf ("%s", &repo);
                                if (repo != 'o' && repo != 'n') printf ("\n\t\t Donnez une autre reponse 'o/n': ");
                        }
                        while (repo != 'o' && repo != 'n');
                }
                while (repo == 'o');

                fprintf (pt_fich, "\n%d", M_def);
                fprintf (pt_fich, "\n%d", nb_m_def);
                for (i = 0; i < nb_m_def; i++) {
                        fprintf (pt_fich, "\n%d", Mdef[i]);
                        fprintf (pt_fich, "\n%d", Ldef[Mdef[i]]);
                        fprintf (pt_fich, "\n%d", Sdef[Mdef[i]]);
                }
        }
        else {
                M_def = 0;
                fprintf (pt_fich, "\n%d", M_def);
                fprintf (pt_fich, "\n%d", M_def);
        }
        fclose(pt_fich);
    }
    else if (rep == 2) {
        printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'cont_dm...' : ");
        scanf ("%s", nom_fich);
        pt_fich = fopen(nom_fich, "r");

        fscanf (pt_fich, "%d", &Couples);
        if ( Couples == 1 ) {
                fscanf (pt_fich, "%d", &nb_couple);
                for ( i = 0; i < nb_couple; i++) {
                        fscanf (pt_fich, "%d", &Cont[i].M1);
                        fscanf (pt_fich, "%d", &Cont[i].M2);
                        fscanf (pt_fich, "%d", &Cont[i].type);
                        fscanf (pt_fich, "%f", &Cont[i].poids);
                        fscanf (pt_fich, "%f", &Cont[i].Dmmmi);
                }
        }
        fscanf (pt_fich, "%d", &M_def);
        if (M_def == 1) {
                fscanf (pt_fich, "%d", &nb_m_def);
                for (i = 0; i < nb_m_def; i++) {
                        fscanf (pt_fich, "%d", &Mdef[i]);
                        fscanf (pt_fich, "%d", &Ldef[Mdef[i]]);
                        fscanf (pt_fich, "%d", &Sdef[Mdef[i]]);
                }
        }
        fclose(pt_fich);
    }

}
```

```
        }

        /* Cas ou il n'y a pas de contraintes */
        /* ********************************** */
        else if (repi == 'n') pres_cont = M_def = Couples = 0;
}
```

6. Programme de construction d'une configuration en ligne droite

```
                        /* ********************************************************** */
                        /* *    Sous-programme de placement de cellules sur une     * */
                        /* *       configuration en SIMPLE LIGNE RECTILIGNE         * */
                        /* *    ----------------------------------------------     * */
                        /* *                      par T. HAMANN                     * */
                        /* ********************************************************** */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include "def.h"

extern int      Q[50],          /* Quantite a produire */
                U[50],          /* Taille du lot transporte */
                M[50],          /* Nombre de machines du routage */
                Ind[50],        /* Indices des machines */
                nb_mach,        /* Nombre de machine */
                nb_prod,        /* Nombre de produits */
                nb_m_def,       /* Nombre de machines instalees definitivement */
                Mdef[20],       /* Indice d'une machine a placee definitivement */
                Sdef[20],       /* Indice d'un site occupe definitivement */
                pres_cont,      /* Indice d'existance des contraintes */
                M_def,          /* Indice d'existance des contraintes de positionnement definitif */
                Couples,        /* Indice d'existance des contraintes de positionnement relatif */
                nb_couple;      /* nombre de couples de machines sous contrainte */

extern float    QsU[50],        /* Rapport Q / U */
                F[50][50],      /* Matrice des flux */
                pm[50],         /* Poids machine */
                wm[50],         /* Largeur machine */
                lm[50],         /* Longueur machine */
                Fmax,           /* Le flux maximum */
                dist_min;       /* Distance inter-machines minimale */

extern contrainte Cont[50];     /* Contrainte de positionnement relatif entre les cellules */

int             mS[50],         /* Indice de la machine se trouvant sur un site */
                MInsta[50],     /* Indice de la machine instalee sur un site */
                MSacc[50],      /* Indice de la machine se trouvant sur un site dans une configuration acceptee */
                Sacc[50],       /* Indice du site d'une machine dans une configuration acceptee */
                MSm[50],        /* Indice de la m/c se trouvant sur un site dans la meil. conf. */
                cont[50][50],   /* Type de contrainte entre deux cellules lorsqu'elle existe */
                S[50];          /* Indice du site d'une machine */

float           dist[50][50],   /* Distance inter machines */
                Xacc[50],       /* Position d'une machine dans une configuration acceptee */
                X[50],          /* Abscisse d'une machine */
                Xm[50],         /* Abscisse d'une machine dans la configurattion */
                dis[50][50],    /* Distance max/min entre deux machines sous contrante */
                poids[50][50],  /* Poids d'une contrainte de positionnement relatif */
                delta;          /* Distance inter site maximale */


sim_lig_lin()
{
        int     i, j, k, temps,
                rep, ms, V,
                flag1, flag2,
                NbACC, NbREF,
                nb_acc, nb_ref,
                nb_iter,
                verifi_cont();

        float   cout0, cout, delta_cout, coufi,
                proba, eps,
                tempi, tempfi, tempera,
                raigeo,
                Xm[50],
                calc_cuta();

        double  drand48();

        char    repo,
                nom_fich[15];

        void    conf_init(),            /* Construction d'une configuration elementaire */
                trans_elem(),           /* Transformations elementaies */
                calc_dist(),            /* Calcul des distances inter-machines */
                ajustement(),           /* Ajustement d'une configuration */
                cons_conf_int(),        /* Conservation d'une configuration intermediaire */
                cons_conf_fin(),        /* Conservation d'une configuration en tant que finale */
                cont_pos(),             /* contraintes de positionnement des machines */
                pos_ent_sor();          /* Positionnement de l'entree et de la sortie de la cellule */

        FILE    *pt_fich, *fopen(), *fclose();


        printf ("\f\n\n\n\t\t\t\t ***************************************************");
        printf ("\n\t\t\t\t *   CONFIGURATION EN SIMPLE LIGNE LINEAIRE   *");
        printf ("\n\t\t\t\t ***************************************************");

        /* Acquisition des contraintes */
        cont_pos();

        /* transformation des contraintes de positionnement relatif */
        for (i = 0; i < nb_mach; i++)
                for (j = i + 1; j < nb_mach; j++) {
                        cont[i][j] = 0;
                        dis[i][j] = 0;
                        poids[i][j] = 0;
                        if ( Couples == 1 )
```

```
                        for (k = 0; k < nb_couple; k++)
                                if ( Cont[k].M1 == i && Cont[k].M2 == j ) {
                                        cont[i][j] = Cont[k].type;
                                        dis[i][j] = Cont[k].Dmami;
                                        poids[i][j] = Cont[k].poids;
                                }
                }

        /* Construction d'une configuration initiale */
        printf ("\n\n\n\t\t Vous pouvez : ");
        printf ("\n\n\t\t\t 1. Donnez une confuguration initiale ");
        printf ("\n\n\t\t\t 2. Faire construire la configuration initiale par le systeme ");
        printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
        do {
                scanf("%d", &rep);
                if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
        }
        while ( rep != 1 && rep != 2 );
        if ( rep == 1 ) {
                for (i = 0; i < nb_mach; i++)
                        mS[i] = -1;
                /* Pour chaque site */
                for (i = 0; i < nb_mach; i++) {
                        printf ("\n\n\t\t Donnez l'indice de la machine se trouvant sur le site %d : ", i + 1);
                        do {
                                scanf ("%d", &ms);
                                ms--;
                                if ( (mS[i] != -1) || (ms > nb_mach) || (ms < -1) ) printf ("\n\n\t\t\t Donnez une autre reponse : ");
                        }
                        while ( (mS[i] != -1) || (ms > nb_mach) || (ms < -1) );
                        mS[i] = ms;
                        S[ms] = i;
                }
                /* Operation d'ajustement de la configuration */
                ajustement();

                /* Calcul des distances inter-machines */
                calc_dist();

                /* Calcul de la valeur du cout */
                cout0 = calc_cuta();
        }
        else {
                /* Constuction d'une configuration initiale par le system */
                conf_init();

                /* Ajustement */
                ajustement();

                /* Calcul des distances inter machines */
                calc_dist();

                /* Calcul de la valeur du cout */
                cout0 = calc_cuta();
        }

        /* Visualisation de la configuration initiale */
        printf ("\f\n\n\n\t\t LA CONFIGURATION INITIALE \n\n\n\t");
        for (i = 0; i < nb_mach; i++)
                printf ("[%d  <%f>]   ", mS[i] + 1, X[mS[i]]);

        printf ("\n\n\n\t\t LE COUT DE CETTE CONFIGURATION EST : %f", cout0);

        /* Acquisition des parametres de controle du recuit simule */
        /* ******************************************************** */
        printf ("\n\n\n\t\t Vous pouvez : ");
        printf ("\n\n\t\t\t 1. Donnez les parametres de controle du R.S ");
        printf ("\n\n\t\t\t 2. Les prendre dans un fichier ");
        printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
        do {
                scanf("%d", &rep);
                if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
        }
        while ( rep != 1 && rep != 2 );
        if ( rep == 1 ) {
                printf ("\n\n\t\t Voulez vous sauvegarder ses donnees dans un fichier : ");
                scanf ("%s", &repo);
                if (repo == 'o') {
                        printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'paco...' : ");
                        scanf ("%s", nom_fich);
                        pt_fich = fopen(nom_fich, "w");

                        printf ("\n\n\t\t Donnez la temperature initiale : ");
                        scanf ("%f", &tempi);
                        fprintf(pt_fich,"%f",tempi);

                        printf ("\n\n\t\t Donnez la temperature finale : ");
                        scanf ("%f", &tempfi);
                        fprintf(pt_fich,"\n%f",tempfi);

                        printf ("\n\n\t\t Donnez la raison geometrique : ");
                        scanf ("%f", &raigeo);
                        fprintf(pt_fich,"\n%f",raigeo);

                        printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                        scanf ("%d", &nb_acc);
                        fprintf(pt_fich,"\n%d",nb_acc);

                        printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                        scanf ("%d", &nb_ref);
                        fprintf(pt_fich,"\n%d",nb_ref);
```

```
                fclose (pt_fich);
        }
        else {
                printf ("\n\n\t\t Donnez la temperature initiale : ");
                scanf ("%f", &tempi);

                printf ("\n\n\t\t Donnez la temperature finale : ");
                scanf ("%f", &tempfi);

                printf ("\n\n\t\t Donnez la raison geometrique : ");
                scanf ("%f", &raigeo);

                printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                scanf ("%d", &nb_acc);

                printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                scanf ("%d", &nb_ref);
        }
}
else {
        printf ("\n\n\n\t\t Donnez le nom du fichier des parametres de controle du R.S. 'paco...' : ");
        scanf ("%s", nom_fich);
        pt_fich = fopen(nom_fich, "r");

        fscanf (pt_fich,"%f",&tempi);
        fscanf (pt_fich,"%f",&tempfi);
        fscanf (pt_fich,"%f",&raigeo);
        fscanf (pt_fich,"%d",&nb_acc);
        fscanf (pt_fich,"%d",&nb_ref);

        fclose (pt_fich);
}

/* Processus du recuit simule */
tempera = tempi;        NbACC = NbREF = nb_iter = 0;
temps = time(0);
coufi = cout = cout0;
do {
        do {
                /* Transformation elementaire */
                trans_elem();

                /* Ajustement */
                ajustement();

                /* Calcul des distances */
                calc_dist();

                /* Calcul de la valeur du cout */
                cout = calc_cuta();

                printf ("\n\n Iteration No %3d|\t Cout : %f|\t cout de la meilleur iter. : %f|\t temps ecoule : %d\n",
                         nb_iter, cout, coufi, time(0) - temps);
                for (i = 0 ; i < nb_mach; i++)
                        printf ("%d <%8.2f>   ", mS[i] + 1, X[mS[i]]);

                /* Calcul de la variation du cout */
                delta_cout = cout - coufi;

                /* Test d'acceptation de Metropolis */
                if ( delta_cout <= 0 ) {
                        /* Conservation de cette configuration res. intermediaire */
                        cons_conf_int();
                        NbACC++;
                        coufi = cout;
                        for (i = 0; i < nb_mach; i++) {
                                MSm[i] = mS[i];         Xm[i] = X[i];
                        }
                }
                else if (delta_cout > 0) {
                        srand48(time(0));
                        do eps = drand48(); while (eps > 1.0);
                        proba = exp(-delta_cout / tempera);
                        if (proba > eps) {
                                /* Conservation de cette configuration res. intermediaire */
                                cons_conf_int();
                                NbACC++;
                        }
                        else {
                                /* Reprise a partir de le derniere configuration acceptee */
                                for (i = 0; i < nb_mach; i++) {
                                        /* Le lieu */
                                        S[i] = Sacc[i];
                                        mS[i] = MSacc[i];
                                        /* La position */
                                        X[i] = Xacc[i];
                                }
                                NbREF++;
                        }
                }

                nb_iter++;
        }
        while ((NbACC < nb_acc)  && (NbREF < nb_ref));
        tempera = tempera * raigeo;
}
/* Test d'arret */
while (tempera > tempfi);

/* Resultat */
```

```
        printf ("\n\n\n\t\t Le nombre total d'iterations est : %d", nb_iter);
        printf ("\n\n\t\t Le temps de traitement est : %d", time(0) - temps);
        printf ("\n\n\t\t Le cout de la configuration initiale est : %f", cout0);
        printf ("\n\n\t\t Le cout de la configuration finale est : %f", coufi);
        printf ("\n\n\t\t Le taux d'amelioration est de : %5.2f pour-cent", (cout0 - coufi) * 100 / cout0 );
        printf ("\n\n\t\t La sequence des machines de la configuration finale est la suivante :\n\n\t");
        for (i = 0 ; i < nb_mach; i++)
                printf ("%d <%8.2f>   ", MSm[i] + 1, Xm[MSm[i]]);

        /* Determination des positions de l'entree et de la sortie de la cellule */
        pos_ent_sor();
}

        /* ########################################################################################### */

                        /* ********************************************************* */
                        /* *     Procedure de construction d'une configuration     * */
                        /* *                        initiale                       * */
                        /* ********************************************************* */
void conf_init()
{
        int     i, j, nb_m_insta, flag, M;
        double  Xdr;

        srand48(time(0));

        /* Initialisation */
        nb_m_insta = 0;
        for (i = 0; i < nb_mach; i++) {
                mS[i] = -1;
                MInsta[i] = -1;
        }

        /* Affectation des machines a position definitive vers leur sites */
        if (pres_cont == 1 && M_def == 1)
                for (i = 0; i < nb_m_def; i++) {
                        S[Mdef[i]] = Sdef[Mdef[i]];
                        mS[S[Mdef[i]]] = Mdef[i];
                        MInsta[i] = Mdef[i];
                        nb_m_insta++;
                }

        /* Affectation aleatoire des machines restantes */
        for (i = 0; i < nb_mach; i++) {
                /* Verifi. site occupe ou pas */
                if ( mS[i] == -1 ) {
                        do {
                                do {Xdr = drand48();}  while ( Xdr > 1.0 );
                                do {
                                        M = (int)(Xdr * (double)nb_mach );
                                }
                                while ( M > nb_mach - 1 );
                                flag = 0;
                                for (j = 0; j < nb_m_insta; j++) {
                                        if ( M == MInsta[j] ) flag = 1;
                                        if (flag == 1) break;
                                }
                        }
                        while ( flag == 1 );
                        S[M] = i;
                        mS[i] = M;
                        nb_m_insta++;
                        MInsta[nb_m_insta - 1] = M;
                }
                if ( nb_m_insta == nb_mach ) break;
        }
}

        /* ########################################################################################### */

                        /* ********************************************************* */
                        /* *     Procedure de realisation d'une transformation     * */
                        /* *              elementaire de reaffectation             * */
                        /* ********************************************************* */
void trans_elem()
{
        int     flag1, i, l, Ma,
                m1, m2, K, s, si,
                trans;
        double  X, drand48();

        srand48(time(0));

        /* Choix de la transformation */
        do {
                X = drand48();
                trans = (int)(X * 2.0);
        }
        while (trans > 1);

        /* Transfert */
        if ( trans == 0 ) {
                /* Tirrage aleatoire d'une machine a transferer */
                do {
                        X = drand48();
                        Ma = (int)(X * (float)nb_mach);
                }
                while (Ma >= nb_mach);

                /* Choix aleatoire du site */
                do {
```

```
                X = drand48();
                s = (int)(X * (float)(nb_mach + 1));
        }
        while (s == S[Ma]);

        /* Transfert vers un site libre */
        if ( s >= nb_mach ) {
                printf ("\n\n Transfert vers site libre \n\n");
                s = nb_mach - 1;
                si = S[Ma];
                for (i = si; i < s; i++)
                        mS[i] = mS[i+1];
                for (i = si; i < s; i++)
                        S[mS[i]] = i;

                S[Ma] = s;
                mS[s] = Ma;
        }

        /* Transfert vers un site occupe */
        else if (s < nb_mach) {
                printf ("\n\n transfert vers site occupe \n\n");
                si = S[Ma];
                if (si > s) {
                        for (i = si; i > s; i--)
                                mS[i] = mS[i-1];
                        for (i = si; i > s; i--)
                                S[mS[i]] = i;
                        mS[s] = Ma;
                        S[Ma] = s;
                }
                else if (s > si) {
                        for (i = si; i < s; i++)
                                mS[i] = mS[i+1];
                        for (i = si; i < s; i++)
                                S[mS[i]] = i;
                        mS[s] = Ma;
                        S[Ma] = s;
                }
        }

/* Permutation de deux machines */
else if ( trans == 1 ) {
        printf ("\n\n permutation \n\n");
        /* Choix du couple de machines a permuter */
        /* 1ere machine */
        do {
                flag1 = 0;
                do {
                        do {X = drand48();}  while ( X > 1.0 );
                        m1 = (int)(X * (double)nb_mach );
                }
                while ( m1 > nb_mach - 1 );

                if (nb_m_def != 0) {
                        for ( i = 0; i < nb_m_def; i++) {
                                if (m1 == Mdef[i])      flag1 = 1;
                                if (flag1 == 1) break;
                        }
                }
        }
        while (flag1 == 1);

        /* 2eme machine */
        do {
                flag1 = 0;
                do {
                        do {X = drand48();}  while ( X > 1.0 );
                        m2 = (int)(X * (double)nb_mach );
                }
                while ( m2 > nb_mach - 1 );

                if (nb_m_def != 0) {
                        for ( i = 0; i < nb_m_def; i++) {
                                if (m2 == Mdef[i])      flag1 = 1;
                                if (flag1 == 1) break;
                        }
                }
        }
        while (flag1 == 1 || m2 == m1);

        /* Realisation de la permutation */

        K = mS[S[m1]]; mS[S[m1]] = mS[S[m2]]; mS[S[m2]] = K;
        K = S[m1]; S[m1] = S[m2]; S[m2] = K;
}
}

/* ########################################################################################## */

                /* ********************************************************* */
                /* *    Procedure d'ajustement apres une transformation    * */
                /* *                        elementaire                    * */
                /* ********************************************************* */
void ajustement()
{
        int     i, j, a, b;
        float   delta, del, depo, dep;
```

```
	/* Calcul du Delta */
	del = 0;
	for (i = 0; i < nb_mach; i++)
		if (lm[i] > del)	del = lm[i];
	delta = del;	del = dist_min;
	for (i = 0; i < nb_mach; i++)
		for (j = i + 1; j < nb_mach; j++)
			if ((cont[i][j] == 2) && (dis[i][j] > del))
				del = dis[i][j];
	delta = delta + del;


	/* Initialisation */
	for (i = 0; i < nb_mach; i++)
		X[mS[i]] = i * delta;

	/* Processus d'ajustement */
	for (i = 1; i < nb_mach; i++){
		a = mS[i];
		dep = 1.E+5;
		for (j = i - 1; j >= 0; j--) {
			b = mS[j];
			if (j == i - 1) {
				if (cont[b][a] == 2) {
					if (dis[b][a] >= dist_min)	depo = X[a] - X[b] - lm[b] - dis[b][a];
					else				depo = X[a] - X[b] - lm[b] - dist_min;
				}
				else	depo = X[a] - X[b] - lm[b] - dist_min;
			}
			else {
				if (cont[b][a] == 2)
					depo = X[a] - X[b] - lm[b] - dis[b][a];
			}
			if (depo <= dep)	dep = depo;
		}
		X[a] = X[a] - dep;
	}
}

	/* ########################################################################################## */

			/* ********************************************************** */
			/* *    Procedure de calcul des distances inter-machines    * */
			/* *                    apres ajustement                    * */
			/* ********************************************************** */
void calc_dist()
{
	int	i, j;

	for (i = 0; i < nb_mach - 1; i++) {
		for (j = i+1; j < nb_mach; j++)
			dist[mS[i]][mS[j]] = dist[mS[j]][mS[i]] = X[mS[j]] - lm[mS[i]] - X[mS[i]];
	}
}

	/* ########################################################################################## */

			/* ********************************************************** */
			/* *    Procedure de calcul du cout d'une configuration     * */
			/* ********************************************************** */
float calc_cuta()
{
	int	i, j;
	float	cout;

	cout = 0;
	for (i = 0; i < nb_mach; i++) {
		for (j = 0; j < nb_mach; j++) {
			if (i != j) {
				/* Cas ou les deux machines sont soumises a une contrainte de positionnement relatif */
				if ( (Couples == 1) && ((cont[i][j] == 1 && dist[i][j] > dis[i][j]) ||
							(cont[i][j] == 2 && dist[i][j] < dis[i][j])) )
					cout = cout + (F[i][j] + (0.5 * poids[i][j] * Fmax)) * (dist[i][j] + lm[i]/2 + lm[j]/2);
				/* Cas ou les deux machines ne sont pas soumises a une contrainte ou soumises a une contrainte respectee */
				else if ( (Couples == 0) || ((Couples == 1) && (cont[i][j] == 0)) ||
					((Couples == 1) && ((cont[i][j] == 1 && dist[i][j] <= dis[i][j])
					|| (cont[i][j] == 2 && dist[i][j] >= dis[i][j]))) )
					cout = cout + F[i][j] * dist[i][j];
			}
		}
	}
	return cout;
}

	/* ########################################################################################## */

			/* ********************************************************** */
			/* *      Procedure de conservation d'une configuration     * */
			/* *                 intermediaire acceptee                 * */
			/* ********************************************************** */
void cons_conf_int()
{
	int	i;


	for (i = 0; i < nb_mach; i++) {
		/* Le lieu */
		Sacc[i] = S[i];
		MSacc[i] = mS[i];
		/* La position */
		Xacc[i] = X[i];
```

```
	}
}

	/* ########################################################################################## */

			/* ************************************************************ */
			/* *	Procedure de conservation d'une configuration	* */
			/* *			qui ameliore le cout			* */
			/* ************************************************************ */
void cons_conf_fin()
{

	int	i;
	FILE	*pt_fich1, *fopen(), *fclose();
	char	conf_bon;

	pt_fich1 = fopen("conf_bon", "w");

	/*fprintf (pt_fich1, "\t\t\t ****   LES LIEUX ET POSITIONS DES MACHINES   ****\n\n\t\t");*/
	for (i = 0; i < nb_mach; i++)
		fprintf (pt_fich1, " [%d   <%f>]  ", MSm[i] + 1, Xm[MSm[i]]);
}

	/* ########################################################################################## */

			/* ************************************************************ */
			/* *    Verification des contraintes fortes de positionnement   * */
			/* *			entre les machines			* */
			/* ************************************************************ */
int verifi_cont()
{
	int i, j, a;

	a = 0;
	for (i = 0; i < nb_mach; i++) {
		for (j = 0; j < nb_mach; j++) {
			if (i != j) {
				if ( ( (cont[i][j] == 1) && (poids[i][j] >= 5) && (dist[i][j] > dis[i][j]) ) ||
					( (cont[i][j] == 2) && (poids[i][j] >= 5) && (dist[i][j] < dis[i][j]) ) )

					a = 1;
			}
			if ( a == 1)	break;
		}
		if ( a == 1)	break;
	}
	return a;
}

	/* ########################################################################################## */

		/* ********************************************************************** */
		/* *    Determination des positions de l'entree et de la sortie	* */
		/* ********************************************************************** */

void pos_ent_sor()
{
	int	i, j,
		F1, F2;

	printf ("\n\n\n\t POSITIONS DE L'ENTREE ET DE LA SORTIE");
	printf ("\n\t ************************************");
	F1 = F2 = 0;
	for (i = 0; i < nb_mach; i++)
		for ( j = i + 1; j < nb_mach; j++)
			if (mS[j] != -1) {
				F1 = F1 + F[mS[i]][mS[j]];
				F2 = F2 + F[mS[j]][mS[i]];
			}
	if (F1 >= F2)	printf ("\n\n\n\t\t L'entree est  gauche et la sortie  droite ");
	else printf ("\n\n\n\t\t L'entree est  droite et la sortie  gauche");

}
		/* ********************************** F I N ********************************** */
```

7. Programme de construction d'une configuration circulaire

```
                        /* ********************************************************** */
                        /* *    Sous-programme de placement de cellules sur une     * */
                        /* *       configuration en SIMPLE LIGNE CIRCULAIRE         * */
                        /* *    ----------------------------------------------     * */
                        /* *                      par T. HAMANN                     * */
                        /* ********************************************************** */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include "def.h"

extern int      Q[100],         /* Quantite a produire */
                U[100],         /* Taille du lot transporte */
                M[50],          /* Nombre de machines du routage */
                Ind[50],        /* Indices des machines */
                cont[50][50],   /* Type de contrainte */
                nb_mach,        /* Nombre de machine */
                nb_prod,        /* Nombre de produits */
                nb_m_def,       /* Nombre de machines instalees definitivement */
                Mdef[100],      /* Indice d'une machine placee definitivement */
                Sdef[20],       /* Indice d'un site occupe definitivement */
                M_def,          /* Indice d'existance de machines a positionnement definitif */
                Couples,        /* Indice d'existance de couples de machines a positionnement relatif */
                nb_couple,      /* Nombre de couples de machines */
                pres_cont,      /* Indice d'existance de contraintes */
                mS[50],         /* Indice de la machine se trouvant sur un site */
                MInsta[50],     /* Indice de la machine instalee sur un site */
                S[50];          /* Indice du site d'une machine */

extern float    dis[50][50],    /* Distance inter machine pour les contraintes */
                poids[50][50],  /* Poids de la contrainte entre deux machines */
                QsU[500],       /* Rapport Q / U */
                F[50][50],      /* Matrice des flux */
                pm[100],        /* Poids machine */
                wm[100],        /* Largeur machine */
                lm[100],        /* Longueur machine */
                Fmax,           /* Le flux maximum */
                dist_min;       /* Distance inter-machines minimale */

extern contrainte Cont[50];     /* Contraintes de positionnement relatif */

int             mS[50],         /* Indice de la machine se trouvant sur un site */
                MInsta[50],     /* Indice de la machine instalee sur un site */
                S[50];          /* Indice du site d'une machine */
                MSacc[100],     /* Indice de la machine se trouvant sur un site dans une configuration acceptee */
                Sacc[100],      /* Indice du site d'une machine dans une configuration acceptee */
                MSm[50];        /* Indice de la m/c se trouvant sur un site dans la meil. conf. */

float           dist[100][100], /* Distance inter machines */
                ALPHAacc[100],  /* Position d'une machine dans une configuration acceptee */
                ALPHA[100],     /* Abscisse d'une machine */
                ALPHAm[50],     /* Abscisse d'une machine dans la configurattion */
                RAY,            /* Rayon moyen de la cellule */
                delta;          /* Distance inter site maximale */

sim_lig_cir()
{
        int     i, j, k, temps,
                rep, nb_m_i,
                flag1, flag2,
                NbACC, NbREF,
                nb_acc, nb_ref,
                nb_iter, V;

        float   cout0, cout, delta_cout, coufi,
                proba, eps,
                tempi, tempfi, tempera,
                raigeo,
                ALPHAm[50],
                calc_cuta();

        double  drand48();

        char    repo, nom_fich[15];

        void    conf_init(),
                trans_elem(),

                calc_dist_cir(),
                ajustement_cir(),
                cons_conf_int(),
                cons_conf_fin(),
                cont_pos(),
                pos_ent_sor();

        FILE    *pt_fich, *fopen(), *fclose();

        printf ("\f\n\n\n\t\t\t\t ******************************************************");
        printf ("\n\t\t\t\t *   CONFIGURATION EN SIMPLE LIGNE CIRCULAIRE   *");
        printf ("\n\t\t\t\t ******************************************************");

        /* Acquisition des contraintes */
        cont_pos();

        /* transformation des contraintes de positionnement relatif */
        if (pres_cont == 1)
            for (i = 0; i < nb_mach; i++)
                for (j = i + 1; j < nb_mach; j++) {
                        cont[i][j] = 0;
                        dis[i][j] = 0;
```

```
                poids[i][j] = 0;
                if ( Couples == 1 )
                        for (k = 0; k < nb_couple; k++)
                                if ( Cont[k].M1 == i && Cont[k].M2 == j ) {
                                        cont[i][j] = Cont[k].type;
                                        dis[i][j] = Cont[k].Dmami;
                                        poids[i][j] = Cont[k].poids;
                                }
        }

/* Acquisition de donnees suplementaires */
printf ("\n\n\n\t\t Donnez le rayon moyen de la cellule : ");
scanf ("%f", &RAY);

/* Construction d'une configuration initiale */
printf ("\n\n\n\n\n\t\t Vous pouvez : ");
printf ("\n\n\t\t\t 1. Donnez une confuguration initiale ");
printf ("\n\n\t\t\t 2. Faire construire la configuration initiale par le systeme ");
printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
do {
        scanf("%d", &rep);
        if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
}
while ( rep != 1 && rep != 2 );
if ( rep == 1 ) {
        nb_m_i = 0;
        for (i = 0; i < nb_mach; i++)
                mS[i] = -1;
        /* Pour chaque site */
        for (i = 0; i < nb_mach; i++) {

                printf ("\n\n\t\t Donnez l'indice de la machine se trouvant sur le site %d : ", i + 1);
                do {
                        flag1 = flag2 = 0;
                        scanf ("%d", &mS[i]);

                        /* 1er test */
                        for (j = 0; j < nb_m_i; j++) {
                                if (mS[j] - 1 != -1)    flag1 = 1;
                                if ( flag1 = 1 ) break;
                        }

                        /* 2eme test */
                        if (mS[i] > nb_mach || mS[i] < 1)       flag2 = 1;

                        if ( flag1 == 1 || flag2 == 1 ) printf ("\n\n\t\t\t Donnez une autre reponse : ");
                }
                while ( flag1 == 1 || flag2 == 1 );
                nb_m_i++;
        }
        /* Operation d'ajustement de la configuration */
        ajustement_cir();

        /* Calcul des distances inter-machines */
        calc_dist_cir();

        /* Calcul de la valeur du cout */
        cout0 = calc_cuta();
}
else {
        /* Constuction d'une configuration initiale par le systeme */
        conf_init();

        /* Ajustement */
        ajustement_cir();

        /* Calcul des distances inter machines */
        calc_dist_cir();

        /* Calcul de la valeur du cout */
        cout0 = calc_cuta();
}

printf ("\n\n\n\t\t LE COUT DE CETTE CONFIGURATION EST : %f", cout0);

/* Visualisation de la configuration initiale */
printf ("\f\n\n\n\t\t LA CONFIGURATION INITIALE \n\n\n\t");
for (i = 0; i < nb_mach; i++)
        printf ("[%d  <%f>]   ", mS[i] + 1, ALPHA[mS[i]] * 180.0 / 3.14);

printf ("\n\n\n\t\t LE COUT DE CETTE CONFIGURATION EST : %f", cout0);

/* Acquisition des parametres de controle du recuit simule */
printf ("\n\n\n\t\t Vous pouvez : ");
printf ("\n\n\t\t\t 1. Donnez les parametres de controle du R.S ");
printf ("\n\n\t\t\t 2. Les prendre dans un fichier ");
printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
do {
        scanf("%d", &rep);
        if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
}
while ( rep != 1 && rep != 2 );
if ( rep == 1 ) {
        printf ("\n\n\t\t Voulez vous sauvegarder ses donnees dans un fichier : ");
        scanf ("%s", &repo);
        if (repo == 'o') {
                printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'paco...' : ");
                scanf ("%s", nom_fich);
                pt_fich = fopen(nom_fich, "w");

                printf ("\n\n\t\t Donnez la temperature initiale : ");
```

```
                scanf ("%f", &tempi);
                fprintf(pt_fich,"%f",tempi);

                printf ("\n\n\t\t Donnez la temperature finale : ");
                scanf ("%f", &tempfi);
                fprintf(pt_fich,"\n%f",tempfi);

                printf ("\n\n\t\t Donnez la raison geometrique : ");
                scanf ("%f", &raigeo);
                fprintf(pt_fich,"\n%f",raigeo);

                printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                scanf ("%d", &nb_acc);
                fprintf(pt_fich,"\n%d",nb_acc);

                printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                scanf ("%d", &nb_ref);
                fprintf(pt_fich,"\n%d",nb_ref);

                fclose (pt_fich);
        }
        else {
                printf ("\n\n\t\t Donnez la temperature initiale : ");
                scanf ("%f", &tempi);

                printf ("\n\n\t\t Donnez la temperature finale : ");
                scanf ("%f", &tempfi);

                printf ("\n\n\t\t Donnez la raison geometrique : ");
                scanf ("%f", &raigeo);

                printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                scanf ("%d", &nb_acc);

                printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                scanf ("%d", &nb_ref);
        }
}
else {
        printf ("\n\n\n\t\t Donnez le nom du fichier des parametres de controle du R.S. 'paco...' : ");
        scanf ("%s", nom_fich);
        pt_fich = fopen(nom_fich, "r");

        fscanf (pt_fich,"%f",&tempi);
        fscanf (pt_fich,"%f",&tempfi);
        fscanf (pt_fich,"%f",&raigeo);
        fscanf (pt_fich,"%d",&nb_acc);
        fscanf (pt_fich,"%d",&nb_ref);

        fclose (pt_fich);
}

/* Processus du recuit simule */
tempera = tempi;        NbACC = NbREF = nb_iter = 0;
temps = time(0);
coufi = cout = cout0;
do {
        do {
                /* Transformation elementaire */
                trans_elem();

                /* Ajustement */
                ajustement_cir();

                /* Calcul des distances */
                calc_dist_cir();

                /* Calcul de la valeur du cout */
                cout = calc_cuta();
                printf ("\n Iteration No %3d|\t Cout : %f|\t cout de la meilleur iter. : %f|\t temps ecoule : %d",
                        nb_iter, cout, coufi, time(0) - temps);

                /* Calcul de la variation du cout */
                delta_cout = cout - coufi;

                /* Test d'acceptation de Metropolis */
                if ( delta_cout <= 0 ) {
                        /* Conservation de cette configuration res. intermediaire */
                        cons_conf_int();
                        NbACC++;
                        /* Verification des contraintes fortes */
                        V = verifi_cont();
                        /* Conservation de cette configuration res. final */
                        cons_conf_fin();
                        /* Si l'une des cont. fortes sont verifiees : conservation de la conf. */
                        if ( V == 0 ) {
                                coufi = cout;
                                for (i = 0; i < nb_mach; i++) {
                                        MSm[i] = mS[i];         ALPHAm[i] = ALPHA[i];
                                }
                        }
                }
                else if (delta_cout > 0) {
                        srand48(time(0));
                        do eps = drand48(); while (eps > 1.0);
                        proba = exp(-delta_cout / tempera);
                        if (proba > eps) {
                                /* Conservation de cette configuration res. intermediaire */
                                cons_conf_int();
                                NbACC++;
                        }
```

```
                        else {
                                /* Reprise a partir de la derniere config. acceptee */
                                for (i = 0; i < nb_mach; i++) {
                                        /* Le lieu */
                                        S[i] = Sacc[i];
                                        mS[i] = MSacc[i];
                                        /* La position */
                                        ALPHA[i] = ALPHAacc[i];
                                }
                                NbREF++;
                        }
                }
                nb_iter++;
                /*printf ("\n Iteration No %3d|\t Cout : %f|\t cout de la meilleur iter. : %f|\t temps ecoule : %d",
                        nb_iter, cout, coufi, time(0) - temps);*/
        }
        while ((NbACC < nb_acc)  && (NbREF < nb_ref));
        tempera = tempera * raigeo;
}
/* Test d'arret */
while (tempera > tempfi);



/* Resultat */
printf ("\n\n\n\t\t Le nombre total d'iterations est : %d", nb_iter);
printf ("\n\n\t\t Le temps de traitement est : %d", time(0) - temps);
printf ("\n\n\t\t Le cout de la configuration initiale est : %f", cout0);
printf ("\n\n\t\t Le cout de la configuration finale est : %f", coufi);
printf ("\n\n\t\t Le taux d'amelioration est de : %5.2f pour-cent", (cout0 - coufi) * 100 / cout0 );
printf ("\n\n\t\t La sequence des machines de la configuration finale est la suivante :\n\n\t");
for (i = 0 ; i < nb_mach; i++)
        printf ("%d <%8.2f>   ", MSm[i] + 1, ALPHAm[MSm[i]] * 180.0 / 3.14);

/* Determination des positions de l'entree et de la sortie de la cellule */
pos_ent_sor();
}

/* ############################################################################################ */

                /* ********************************************************** */
                /* *    Procedure d'ajustement apres une transformation     * */
                /* *                    elementaire                         * */
                /* ********************************************************** */
void ajustement_cir()
{
        int     i, j, a, b;
        float   delta, del, depo, dep;

        /* Calcul du Delta */
        del = 0;
        for (i = 0; i < nb_mach; i++)
                if (lm[i] > del)        del = lm[i];
        delta = del;    del = dist_min;

        if ( pres_cont == 1 && Couples == 1)
                for (i = 0; i < nb_mach; i++)
                        for (j = i + 1; j < nb_mach; j++)
                                if (cont[i][j] == 2)
                                        if (dis[i][j] > del)    del = dis[i][j];
        delta = delta + del;


        /* Initialisation */
        for (i = 0; i < nb_mach; i++)
                ALPHA[mS[i]] = i * delta / RAY;

        /* Processus d'ajustement */
        for (i = 1; i < nb_mach; i++){
                a = mS[i];
                dep = 1.E+5;
                for (j = i - 1; j >= 0; j--) {
                        b = mS[j];
                        if (j == i - 1) {
                                if (cont[b][a] == 2) {
                                        if (dis[b][a] >= dist_min)      depo = ALPHA[a] - ALPHA[b] - (lm[b] + dis[b][a]) / RAY;
                                        else                            depo = ALPHA[a] - ALPHA[b] - (lm[b] + dist_min) / RAY;
                                }
                                else    depo = ALPHA[a] - ALPHA[b] - (lm[b] + dist_min) / RAY;
                        }
                        else {
                                if (cont[b][a] == 2)
                                        depo = ALPHA[a] - ALPHA[b] - (lm[b] + dis[b][a]) / RAY;
                        }
                        if (depo <= dep)        dep = depo;
                }
                ALPHA[a] = ALPHA[a] - dep;
        }
}

/* ############################################################################################ */

                /* ********************************************************** */
                /* *    Procedure de calcul des distances inter-machines    * */
                /* *                    apres ajustement                    * */
                /* ********************************************************** */
void calc_dist_cir()
{
        int     i, j;

        for (i = 0; i < nb_mach - 1; i++) {
```

```
                for (j = i+1; j < nb_mach; j++) {
                        dist[mS[i]][mS[j]] = dist[mS[j]][mS[i]] = RAY * (ALPHA[mS[j]] - ALPHA[mS[i]]) - lm[mS[i]];
                }
        }
}

        /* ########################################################################################## */

                        /* ************************************************************ */
                        /* *      Procedure de conservation d'une configuration       * */
                        /* *                    intermediaire acceptee                * */
                        /* ************************************************************ */
void cons_conf_int_cir()
{
        int     i;


        for (i = 0; i < nb_mach; i++) {
                /* Le lieu */
                Sacc[i] = S[i];
                MSacc[i] = mS[i];
                /* La position */
                ALPHAacc[i] = ALPHA[i];
        }
}

        /* ########################################################################################## */

                        /* ************************************************************ */
                        /* *      Procedure de conservation d'une configuration       * */
                        /* *                      qui ameliore le cout                * */
                        /* ************************************************************ */
void cons_conf_fin_cir()
{

        int     i;
        FILE    *pt_fich1, *fopen(), *fclose();
        char    conf_bon;

        pt_fich1 = fopen("conf_bon", "w");

        fprintf (pt_fich1, "\t\t\t ****   LES LIEUX ET POSITIONS DES MACHINES   ****\n\n\t\t");
        for (i = 0; i < nb_mach; i++)
                fprintf (pt_fich1, " [%d   <%f>]  ", MSm[i] + 1, ALPHAm[MSm[i]]);
}
                /* ********************************** F I N ************************************ */
```

8. Programme de construction d'une configuration en ligne double

```
                        /* ********************************************************** */
                        /* *    Sous-programme de placement de cellules sur une     * */
                        /* *       configuration en DOUBLE LIGNE RECTILIGNE         * */
                        /* *    ----------------------------------------------     * */
                        /* *                      par T. HAMANN                     * */
                        /* ********************************************************** */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include "def.h"

extern int      Q[50],          /* Quantite a produire */
                U[50],          /* Taille du lot transporte */
                M[50],          /* Nombre de machines du routage */
                Ind[50],        /* Indices des machines */
                nb_mach,        /* Nombre de machine */
                nb_prod,        /* Nombre de produits */
                pres_cont,      /* Indice d'existance de contraintes */
                Couples,        /* Indice d'existance de contraintes de positionnement relatif */
                nb_couple,      /* Nombre de couples de machines */
                M_def,          /* Indice d'existance de contraintes de positionnement definitif */
                nb_m_def,       /* Nombre de machines instalees definitivement */
                Mdef[50],       /* Indice d'une machine placee definitivement */
                Ldef[50],       /* Indice de la ligne d'une machine placee defini. */
                Sdef[50];       /* Indice d'un site occupe definitivement */

extern contrainte Cont[50];     /* Contraintes de positionnement relatif */

extern float    QsU[500],       /* Rapport Q / U */
                F[50][50],      /* Matrice des flux */
                pm[50],         /* Poids machine */
                wm[50],         /* Largeur machine */
                lm[50],         /* Longueur machine */
                Fmax,           /* Le flux maximum */
                dist_min;       /* Distance inter-machines minimale */

int             cont[50][50],   /* Type de contrainte */
                MS[2][50],      /* Indice de la machine se trouvant sur un site */
                MInsta[50],     /* Indice de la machine instalee sur un site */
                MSacc[2][50],   /* Indice de la m/c se trouvant sur un site dans une config. acceptee */
                Lacc[50],       /* Indice de la ligne contenant une machine dans une conf. acceptee */
                Sacc[50],       /* Indice du site d'une machine dans une configuration acceptee */
                MSm[2][50],     /* Indice de la m/c se trouvant sur un site dans la meil. conf. */
                L[50],          /* Numero de ligne d'une machine */
                S[50],          /* Indice du site d'une machine */
                mml,            /* Nombre maximum de machines par lignes */
                nml[2];         /* Nombre de machines par ligne */

float           dis[50][50],    /* Distance inter machine pour les contraintes */
                poids[50][50],  /* Poids de la contrainte entre deux machines */
                dist[50][50],   /* Distance inter machines */
                Xacc[50],       /* Position d'une machine dans une config. acceptee */
                X[50],          /* Abscisse d'une machine */
                Xm[50],         /* Abscisse d'une machine dans la configurattion */
                delta,          /* Distance inter site maximale */
                dil;            /* Distance inter-lignes */

dou_lig_lin()
{
        int     i, j, k, l, s, temps,
                rep, MM,
                NbACC, NbREF,
                nb_acc, nb_ref,
                nb_iter;

        float   cout0, cout, delta_cout, coufi,
                proba, eps,
                tempi, tempfi, tempera,
                raigeo,
                Xm[50],
                calc_cuta_dou();

        double  drand48();

        char    repo,
                nom_fich[15];

        void    conf_init_dou(),
                trans_elem_dou(),
                calc_dist_dou(),
                ajustement_dou(),
                cons_conf_int_dou(),
                cons_conf_fin_dou(),
                cont_pos();

        FILE    *pt_fich, *fopen(), *fclose();



        printf ("\f\n\n\n\t\t\t\t ***************************************");
        printf ("\n\t\t\t\t *   CONFIGURATION EN DOUBLE LIGNE   *");
        printf ("\n\t\t\t\t ***************************************");

        /* Acquisition des contraintes */
        cont_pos();

        /* Transformation des contraintes de positionnement relatif */
        if ( pres_cont == 1)
            for (i = 0; i < nb_mach; i++)
                for (j = 0; j < nb_mach; j++) {
```

```
                cont[i][j] = 0;
                dis[i][j] = 0;
                poids[i][j] = 0;
                if ( Couples == 1 )
                        for (k = 0; k < nb_couple; k++)
                                if ( Cont[k].M1 == i && Cont[k].M2 == j ) {
                                        cont[i][j] = Cont[k].type;
                                        dis[i][j] = Cont[k].Dmami;
                                        poids[i][j] = Cont[k].poids;
                                }
        }


/* Donnees suplementaires */
printf ("\n\n\n\t\t\t Donnez la distance entre les machines des deux lignes : ");
scanf ("%f", &dil);

printf ("\n\n\n\t\t\t Donnez le nombre maximal de machines par ligne : ");
do {
        scanf ("%d", &mml);
        if ( 2 * mml < nb_mach ) printf ("\n\n\t\t Ce nombre est insuffisant, donnez une autre reponse : ");
}
while (2 * mml < nb_mach);

/* Construction d'une configuration initiale */
printf ("\n\n\n\t\t Vous pouvez : ");
printf ("\n\n\t\t\t 1. Donnez une confuguration initiale ");
printf ("\n\n\t\t\t 2. Faire construire la configuration initiale par le systeme ");
printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
do {
        scanf("%d", &rep);
        if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
}
while ( rep != 1 && rep != 2 );
if ( rep == 1 ) {
        for (i = 0; i < 2; i++) {
                nml[i] = 0;
                for (j = 0; j < nb_mach; j++) MS[i][j] = -1;
        }
        /* Pour chaque machine */
        for (i = 0; i < nb_mach; i++) {
                printf ("\n\n\n\t\t Machine %d :", i+1);
                do {
                        printf ("\n\n\t\t\t Donnez son No de ligne : ");
                        do {
                                scanf ("%d", &l);       l = l;
                                if (l > 2) printf ("\n\n\t\t Donnez une autre reponse : ");
                                else if (nml[l] > mml)  printf ("\n\n\t\t Cette ligne est plaine, donnez une autre reponse : ");
                        }
                        while ( l > 2 );

                        printf ("\n\n\t\t\t Donnez son No de site sur la ligne %d : ", l+1);
                        do {
                                scanf ("%d", &s);       s = s;
                                if (s > mml) printf ("\n\n\t\t Donnez une autre reponse : ");
                        }
                        while ( s > mml );
                }
                /* Verification si le site est libre */
                while ( MS[l][s] != -1 );
                nml[l]++;
                MS[l][s] = i;
        }

        /* Elimination des sites non occupes */
        MM = mml;
        for (i = 0; i < 2; i++)
                for (j = 0; j < MM; j++)
                        if ((MS[i][j] == -1) && (j != MM - 1))
                                for (k = j + 1; k < MM; k++) {
                                        MS[i][k-1] = MS[i][k];
                                        MM--;
                                }

        /* Operation d'ajustement de la configuration */
        ajustement_dou();

        /* Calcul des distances inter-machines */
        calc_dist_dou();

        /* Calcul de la valeur du cout */
        cout0 = calc_cuta_dou();
}
else {
        /* Constuction d'une configuration initiale par le system */
        conf_init_dou();

        /* Ajustement */
        ajustement_dou();

        /* Calcul des distances inter machines */
        calc_dist_dou();

        /* Calcul de la valeur du cout */
        cout0 = calc_cuta_dou();

        printf ("\n\n\n\t\t LE COUT DE CETTE CONFIGURATION EST : %f", cout0);
}
/* Visualisation de la configuration initiale */
printf ("\f\n\n\n\t\t LA CONFIGURATION INITIALE \n\n\n\t");
for(i = 0; i < 2; i++) {
```

```
        printf ("\n\n\t Ligne %d : ", i+1);
        for (j = 0; j < nml[i]; j++)
                printf ("[%d  <%f>]   ", MS[i][j] + 1, X[MS[i][j]]);
}
/* Acquisition des parametres de controle du recuit simule */
printf ("\n\n\n\t\t Vous pouvez : ");
printf ("\n\n\t\t\t 1. Donnez les parametres de controle du R.S ");
printf ("\n\n\t\t\t 2. Les prendre dans un fichier ");
printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
do {
        scanf("%d", &rep);
        if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
}
while ( rep != 1 && rep != 2 );
if ( rep == 1 ) {
        printf ("\n\n\t\t Voulez vous sauvegarder ses donnees dans un fichier : ");
        scanf ("%s", &repo);
        if (repo == 'o') {
                printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'paco...' : ");
                scanf ("%s", nom_fich);
                pt_fich = fopen(nom_fich, "w");

                printf ("\n\n\t\t Donnez la temperature initiale : ");
                scanf ("%f", &tempi);
                fprintf(pt_fich,"%f",tempi);

                printf ("\n\n\t\t Donnez la temperature finale : ");
                scanf ("%f", &tempfi);
                fprintf(pt_fich,"\n%f",tempfi);

                printf ("\n\n\t\t Donnez la raison geometrique : ");
                scanf ("%f", &raigeo);
                fprintf(pt_fich,"\n%f",raigeo);

                printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                scanf ("%d", &nb_acc);
                fprintf(pt_fich,"\n%d",nb_acc);

                printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                scanf ("%d", &nb_ref);
                fprintf(pt_fich,"\n%d",nb_ref);

                fclose (pt_fich);
        }
        else {
                printf ("\n\n\t\t Donnez la temperature initiale : ");
                scanf ("%f", &tempi);

                printf ("\n\n\t\t Donnez la temperature finale : ");
                scanf ("%f", &tempfi);

                printf ("\n\n\t\t Donnez la raison geometrique : ");
                scanf ("%f", &raigeo);

                printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                scanf ("%d", &nb_acc);

                printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                scanf ("%d", &nb_ref);
        }
}
else {
        printf ("\n\n\n\t\t Donnez le nom du fichier des parametres de controle du R.S. 'paco...' : ");
        scanf ("%s", nom_fich);
        pt_fich = fopen(nom_fich, "r");

        fscanf (pt_fich,"%f",&tempi);
        fscanf (pt_fich,"%f",&tempfi);
        fscanf (pt_fich,"%f",&raigeo);
        fscanf (pt_fich,"%d",&nb_acc);
        fscanf (pt_fich,"%d",&nb_ref);

        fclose (pt_fich);
}

/* Processus du recuit simule */
tempera = tempi;        NbACC = NbREF = nb_iter = 0;
temps = time(0);
coufi = cout = cout0;
do {
        do {
                /* Transformation elementaire */
                trans_elem_dou();

                /* Ajustement */
                ajustement_dou();

                /* Calcul des distances */
                calc_dist_dou();

                /* Calcul de la valeur du cout */
                cout = calc_cuta_dou();

                /* Calcul de la variation du cout */
                delta_cout = cout - coufi;

                /* Test d'acceptation de Metropolis */
                if ( delta_cout <= 0 ) {
                        /* Conservation de cette configuration res. intermediaire */
                        cons_conf_int_dou();
                        NbACC++;
```

```
                        /* Conservation de cette configuration res. final */
                        cons_conf_fin_dou();
                        coufi = cout;
                        for (i = 0; i < 2; i++) {
                                for (j = 0; j < nml[i]; j++) {
                                        MSm[i][j] = MS[i][j];
                                        Xm[MS[i][j]] = X[MS[i][j]];
                                }
                        }
                }
                else if (delta_cout > 0) {
                        srand48(time(0));
                        do eps = drand48(); while (eps > 1.0);
                        proba = exp(-delta_cout / tempera);
                        if (proba > eps) {
                                /* Conservation de cette config. res. intermediaire */
                                cons_conf_int_dou();
                                NbACC++;
                        }
                        else NbREF++;
                }
                nb_iter++;
                printf ("\n Iteration No %3d|\t Cout : %f|\t cout de la meilleur iter. : %f|\t temps ecoule : %d",
                        nb_iter, cout, coufi, time(0) - temps);
        }
        while ((NbACC < nb_acc)  && (NbREF < nb_ref));
        tempera = tempera * raigeo;
    }
    /* Test d'arret */
    while (tempera > tempfi);



    /* Resultat */
    printf ("\n\n\n\t\t Le nombre total d'iterations est : %d", nb_iter);
    printf ("\n\n\t\t Le temps de traitement est : %d", time(0) - temps);
    printf ("\n\n\t\t Le cout de la configuration initiale est : %f", cout0);
    printf ("\n\n\t\t Le cout de la configuration finale est : %f", coufi);
    printf ("\n\n\t\t Le taux d'amelioration est de : %5.2f pour-cent", (cout0 - coufi) * 100 / cout0 );
    printf ("\n\n\t\t La sequence des machines de la configuration finale est la suivante :\n\n\t");
    for (i = 0 ; i < 2; i++) {
            printf ("\n\n\t Ligne %d : ", i+1);
            for (j = 0 ; j < nml[i]; j++)
                    printf (" [%d <%8.2f>]   ", MSm[i][j] + 1, Xm[MSm[i][j]]);
    }
}

    /* ########################################################################################## */

                    /* ********************************************************** */
                    /* *    Procedure de calcul du cout d'une configuration     * */
                    /* ********************************************************** */
float calc_cuta_dou()
{
    int     i, j;
    float   cout;

    cout = 0;
    for (i = 0; i < nb_mach; i++) {
            for (j = 0; j < nb_mach; j++) {
                    if (i != j) {
                            /* Cas ou les deux machines sont soumises a une contrainte de positionnement relatif */
                            if ( (Couples == 1) && ((cont[i][j] == 1 && dist[i][j] > dis[i][j]) ||
                                                    (cont[i][j] == 2 && dist[i][j] < dis[i][j])) ) {
                                    if (L[i] == L[j])
                                            cout = cout + F[i][j] * dist[i][j] + (0.5 * poids[i][j] * Fmax) * (dist[i][j] + lm[i]/2 + lm[j]/2);
                                    else if (L[i] != L[j])
                                            cout = cout + F[i][j] * dist[i][j] + (0.5 * poids[i][j] * Fmax) * sqrt((X[i]-X[j]+(lm[i]-lm[j])/2)*
]-X[j]+(lm[i]-lm[j])/2) + dil*dil);
                            }
                            /* Cas ou les deux machines ne sont pas soumises a une contrainte ou soumises a une contrainte respectee */
                            else if ( (Couples == 0) || ((Couples == 1) && (cont[i][j] == 0)) ||
                                            ((Couples == 1) && ((cont[i][j] == 1 && dist[i][j] <= dis[i][j])
                                            || (cont[i][j] == 2 && dist[i][j] >= dis[i][j]))) )
                                    cout = cout + F[i][j] * dist[i][j];
                    }
            }
    }
    return cout;
}

    /* ########################################################################################## */


                    /* ********************************************************** */
                    /* *    Procedure de construction d'une configuration       * */
                    /* *                        initiale                        * */
                    /* ********************************************************** */
void conf_init_dou()
{
    int     i, j, k, l, s, flag, nb_m_insta, MM;
    double  Xdr, drand48();

    srand48(time(0));

    /* Initialisation */
    nb_m_insta = 0;
    for (i = 0; i < 2; i++) {
            nml[i] = 0;
            for (j = 0; j < nml; j++)
                    MS[i][j] = -1;
```

```
	}
	for (i = 0; i < nb_mach; i++) {
		MInsta[i] = -1;
		L[i] = -1;
		S[i] = -1;
	}

	/* Affectation des machines a position definitive vers leur sites */
	if (pres_cont == 1 && M_def == 1)
		for (i = 0; i < nb_m_def; i++) {
			L[Mdef[i]] = Ldef[Mdef[i]];
			S[Mdef[i]] = Sdef[Mdef[i]];
			MS[L[Mdef[i]]][S[Mdef[i]]] = Mdef[i];
			nml[L[Mdef[i]]]++;
			MInsta[i] = Mdef[i];
			nb_m_insta++;
		}

	/* Affectation aleatoire des machines restantes */
	for (i = 0; i < nb_mach; i++) {
		if ( S[i] == -1 ) {
			/* La ligne */
			do {
				Xdr = drand48();
				l = (int)(Xdr * 2.0 );
			}
			while ( (l > 1) || (nml[l] > mml) );
			L[i] = l;

			/* La position sur la ligne */
			do {
				Xdr = drand48();
				s = (int)(Xdr * mml);
			}
			while ( (s > mml) || (MS[l][s] != -1) );
			S[i] = s;

			nml[L[i]]++;
			MS[L[i]][S[i]] = i;
			MInsta[nb_m_insta] = i;
			nb_m_insta++;
		}
	}

	for (i = 0; i < 2; i++) {
		printf ("\n\n\t Ligne %d : ", i + 1);
		for (j = 0; j < mml; j++)	printf (" %d  ", MS[i][j]);
	}

	/* Elimination des sites non occupes */
	for (i = 0; i < 2; i++) {
		MM = mml;
		do {
		   flag = 0;
		   for (j = 0; j < MM; j++) {
			if (MS[i][j] == -1) {
				for (k = j + 1; k < MM; k++)
					MS[i][k-1] = MS[i][k];
				MM--;
				flag == 1;
			}
			if (flag == 1)  break;
		   }
		}
		while ( MM > nml[i] );
		for (j = 0; j < nml[i]; j++)
			S[MS[i][j]] = j;
	}

	for (i = 0; i < 2; i++) {
		printf ("\n\n\t\t Ligne %d : ", i + 1);
		for (j = 0; j < nml[i]; j++)	printf (" %d  ", MS[i][j]);
	}
}

		/* ############################################################################ */

			/* ********************************************************* */
			/* *    Procedure de realisation d'une transformation      * */
			/* *            elementaire de reafectation                * */
			/* ********************************************************* */

void trans_elem_dou()
{

	int	trans, flag, Ma, i, K,
		l, l1, l2, m1, m2,
		s, s1, flag1;

	double  X, drand48();

	/* Choix de la transformation elementaire */
	do {
		X = drand48();
		trans = (int)(X * 2.0);
	}
	while (trans > 1);

	/* Transfert */
	if ( trans == 0 && (nb_mach < 2 * mml) ) {
	    flag = 0;
```

```
        do {
            /* Tirrage aleatoire d'une machine a transferer */
            do {
                    X = drand48();
                    Ma = (int)(X * (float)nb_mach);
            }
            while (Ma >= nb_mach);

            /* Tirrage aleatoire d'une ligne de destination */
            do {
                    X = drand48();
                    l = (int)(X * 2.0);
            }
            while ( (l > 1) || (nml[l] >= mml) );

            /* Transfert sur la meme ligne */
            if (l == L[Ma]) {
                    /* Choix aleatoire du site sur cette ligne */
                    do {
                            X = drand48();
                            s = (int)(X * (float)mml);
                    }
                    while ((s >= mml) || (s == S[Ma]));

                    /* Transfert vers un site libre */
                    if ( (nml[l] < mml) && (s >= nml[l]) ) {
                            for (i = S[Ma] + 1; i < nml[l]; i++)
                                    MS[l][i-1] = MS[l][i];
                            for (i = S[Ma]; i < nml[l]; i++)
                                    S[MS[l][i]]--;

                            S[Ma] = nml[l]-1;
                            MS[l][nml[l]-1] = Ma;
                    }

                    /* Transfert vers un site occupe */
                    else if (s < nml[l]) {
                            si = S[Ma];
                            if (si > s) {
                                    for (i = si; i > s; i--)
                                            MS[l][i] = MS[l][i-1];
                                    for (i = si; i > s; i--)
                                            S[MS[l][i]] = i;
                                    MS[l][s] = Ma;
                                    S[Ma] = s;
                            }
                            else if (s > si) {
                                    for (i = si; i < s; i++)
                                            MS[l][i] = MS[l][i+1];
                                    for (i = si; i < s; i++)
                                            S[MS[l][i]] = i;
                                    MS[l][s] = Ma;
                                    S[Ma] = s;
                            }
                    }
            }

            /* Transfert vers une autre ligne */
            else {
                    l1 = L[Ma];
                    l2 = l;
                    if (nml[l2] < mml) {
                            L[Ma] = l2;
                            /* Extraction de la m/c de sa ligne initiale */
                            s = S[Ma];
                            for (i = s + 1; i < nml[l1]; i++)
                                    MS[l1][i-1] = MS[l1][i];
                            for (i = s; i < nml[l1]; i++)
                                    S[MS[l1][i]]--;
                            nml[l1]--;

                            /* Placement de la m/a sur se nouvelle ligne */
                            /* ***************************************** */
                            /* selection aleatoire d'un site sur la nouvelle ligne */
                            do {
                                    X = drand48();
                                    s = (int)(X * (float)mml);
                            }
                            while (s >= mml);
                            /* Cas d'un transfert vers un site libre */
                            if ((nml[l2] < mml) && (s >= nml[l2])) {
                                    S[Ma] = nml[l2];
                                    MS[l2][nml[l2]] = Ma;
                            }
                            /* Cas d'un transfert vers un site occupe */
                            else if (s < nml[l2]) {
                                    for (i = nml[l2]; i > s; i--)
                                            MS[l2][i] = MS[l2][i-1];
                                    for (i = nml[l2]; i > s; i--)
                                            S[MS[l2][i]] = i;
                                    S[Ma] = s;
                                    MS[l2][s] = Ma;
                            }
                            nml[l2]++;
                    }
                    else    flag = 1;
            }
            if ( flag == 0) printf ("\n Transfert de la Machine %d vers le Site %d de la Ligne %d", Ma+1, S[Ma]+1, L[Ma]+1);
        }
        while ( flag == 1 );
    }
```

```
	/* Permutation */
	else {
		/* Tirrage aleatoire des deux machines a permuter */

		/* 1ere machine */
		do {
			flag1 = 0;
			do {
				do {X = drand48();}  while ( X > 1.0 );
				m1 = (int)(X * (double)nb_mach);
			}
			while ( m1 > nb_mach - 1 );

			if (nb_m_def != 0)
				for ( i = 0; i < nb_m_def; i++) {
					if (m1 == Mdef[i])	flag1 = 1;
					if (flag1 == 1) break;
				}
		}
		while (flag1 == 1);

		/* 2eme machine */
		do {
			flag1 = 0;
			do {
				do {X = drand48();}  while ( X > 1.0 );
				m2 = (int)(X * (double)nb_mach );
			}
			while ( m2 > nb_mach - 1 );

			if (nb_m_def != 0)
				for ( i = 0; i < nb_m_def; i++) {
					if (m2 == Mdef[i])	flag1 = 1;
					if (flag1 == 1) break;
				}
		}
		while (flag1 == 1 || m2 == m1);

		/* Cas d'une permutation sur une meme ligne */
		if ( L[m1] == L[m2] ) {
			K = MS[L[m1]][S[m1]]; MS[L[m1]][S[m1]] = MS[L[m2]][S[m2]]; MS[L[m2]][S[m2]] = K;
			K = S[m1]; S[m1] = S[m2]; S[m2] = K;
		}

		/* Cas d'une permutation d'une ligne a une autre */
		else if ( L[m1] != L[m2] ) {
			K = MS[L[m1]][S[m1]]; MS[L[m1]][S[m1]] = MS[L[m2]][S[m2]]; MS[L[m2]][S[m2]] = K;
			K = L[m1]; L[m1] = L[m2]; L[m2] = K;
			K = S[m1]; S[m1] = S[m2]; S[m2] = K;
		}
		printf ("\n\n Permutation des machines %d et %d.", m1+1, m2+1);
	}
}

		/* ########################################################################## */

			/* ********************************************************** */
			/* *    Procedure d'ajustement apres une transformation     * */
			/* *                        elementaire                     * */
			/* ********************************************************** */

void ajustement_dou()
{
	int	i, j, k, a, b;
	float	delta, del, depo, dep;

	/* Calcul du Delta */
	del = 0;
	for (i = 0; i < nb_mach; i++)
		if (lm[i] > del)		del = lm[i];
	delta = del;
	if (dist_min >= dil)		del = dist_min;
	else if (dil > dist_min)	del = dil;
	if ( Couples == 1 )
		for (i = 0; i < nb_mach; i++)
			for (j = i + 1; j < nb_mach; j++)
				if (cont[i][j] == 2)
					if (dis[i][j] > del)	del = dis[i][j];
	delta = delta + del;

	/* Initialisation */
	for (i = 0; i < 2; i++)
		for (j = 0; j < nml[i]; j++)
			X[MS[i][j]] = j * delta;

	/* Processus d'ajustement */
	for (k = 0; k < 2; k++)
		for (i = 1; i < nml[k]; i++){
			a = MS[k][i];
			dep = 1.E+5;
			for (j = i - 1; j >= 0; j--) {
				b = MS[k][j];
				if (j == i - 1) {
					if (cont[b][a] == 2) {
						if (dis[b][a] >= dist_min)	depo = X[a] - X[b] - lm[b] - dis[b][a];
						else				depo = X[a] - X[b] - lm[b] - dist_min;
					}
					else	depo = X[a] - X[b] - lm[b] - dist_min;
				}
				else {
```

```
                                if (cont[b][a] == 2)
                                depo = X[a] - X[b] - lm[b] - dis[b][a];
                        }
                        if (depo <= dep)        dep = depo;
                }
                X[a] = X[a] - dep;
        }
}


        /* ########################################################################### */

                /* ********************************************************** */
                /* *    Procedure de calcul des distances inter-machines    * */
                /* *                    apres ajustement                    * */
                /* ********************************************************** */


void calc_dist_dou()
{
        int     i, j, k;

        /* Distances entre les machines d'une meme ligne */
        for (k = 0; k < 2; k++) {
                for (i = 0; i < nml[k] - 1; i++)
                        for (j = i+1; j < nml[k]; j++)
                                dist[MS[k][i]][MS[k][j]] = dist[MS[k][j]][MS[k][i]] = (X[MS[k][j]] - X[MS[k][i]])
                                        + ( lm[MS[k][j]] - lm[MS[k][i]] ) / 2 + dil;
        }

        /* Distance entre les machines de lignes differentes */
        for (i = 0; i < nml[0]; i++)
                for (j = 0; j < nml[1]; j++) {
                        if (X[MS[0][i]] >= X[MS[1][j]])
                                dist[MS[0][i]][MS[1][j]] = dist[MS[1][j]][MS[0][i]] = (X[MS[0][i]] - X[MS[1][j]])
                                        + (lm[MS[0][i]] - lm[MS[1][j]]) / 2 + dil;
                        else if (X[MS[1][j]] > X[MS[0][i]])
                                dist[MS[0][i]][MS[1][j]] = dist[MS[1][j]][MS[0][i]] = (X[MS[1][j]] - X[MS[0][i]])
                                        + (lm[MS[1][j]] - lm[MS[0][i]]) / 2 + dil;
                }
}

        /* ########################################################################### */

                /* ********************************************************** */
                /* *      Procedure de conservation d'une configuration     * */
                /* *                  intermediaire acceptee                * */
                /* ********************************************************** */

void cons_conf_int_dou()
{
        int     i, k;


        for (i = 0; i < nb_mach; i++) {
                /* Le lieu */
                Lacc[i] = L[i];
                Sacc[i] = S[i];
                /* La position */
                Xacc[i] = X[i];
        }
        for (k = 0; k < 2; k++) {
                for ( i = 0; i < nml[k]; i++)
                        MSacc[k][i] = MS[k][i];
        }
}

        /* ########################################################################### */

                /* ********************************************************** */
                /* *      Procedure de conservation d'une configuration     * */
                /* *                    qui ameliore le cout                * */
                /* ********************************************************** */

void cons_conf_fin_dou()
{

        int     i, k;
        FILE    *pt_fich1, *fopen(), *fclose();
        char    conf_bon;

        pt_fich1 = fopen("conf_bon", "w");

        fprintf (pt_fich1, "\t\t\t ****   LES LIEUX ET POSITIONS DES MACHINES   ****\n\n");
        for (k = 0; k < 2; k++) {
                fprintf (pt_fich1, "\n Ligne %d : \t", k + 1);
                for (i = 0; i < nb_mach; i++)
                        MSm[k][i]++;
                        fprintf (pt_fich1, " [%d   <%f>]  ", MSm[k][i], Xm[MSm[k][i]]);
        }

}

        /* ***************************   Fin   ***************************** */
```

9. Programme de construction d'une configuration multi-lignes

```
                    /* ********************************************************** */
                    /* *    Sous-programme de placement de cellules sur une     * */
                    /* *              configuration en MULTI-LIGNES             * */
                    /* *    ---------------------------------------------     * */
                    /* *                       par T. HAMANN                  * */
                    /* ********************************************************** */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include "def.h"

extern int      Q[50],          /* Quantite a produire */
                U[50],          /* Taille du lot transporte */
                M[50],          /* Nombre de machines du routage */
                Ind[50],        /* Indices des machines */
                nb_mach,        /* Nombre de machine */
                nb_prod,        /* Nombre de produits */
                pres_cont,      /* Indice d'existance de contraintes */
                Couples,        /* Indice d'existance de contraintes de positionnement relatif */
                nb_couple,      /* Nombre de couples de machines */
                M_def,          /* Indice d'existance de contraintes de positionnement definitif */
                nb_m_def,       /* Nombre de machines instalees definitivement */
                Mdef[50],       /* Indice d'une machine placee definitivement */
                Ldef[50],       /* Indice de la ligne d'une machine placee defini. */
                Sdef[50];       /* Indice d'un site occupe definitivement */

extern contrainte Cont[50];     /* Contraintes de positionnement relatif */

extern float    QsU[500],       /* Rapport Q / U */
                F[50][50],      /* Matrice des flux */
                pm[50],         /* Poids machine */
                wm[50],         /* Largeur machine */
                lm[50],         /* Longueur machine */
                Fmax,           /* Le flux maximum */
                lmax, wmax,     /* Plus grande longueur et plus grande largeur */
                dist_min;       /* Distance inter-machines minimale */

int             cont[50][50],   /* Type de contrainte */
                MS[20][50],     /* Indice de la machine se trouvant sur un site */
                MInsta[50],     /* Indice de la machine instalee sur un site */
                MSacc[20][50],  /* Indice de la m/c se trouvant sur un site dans une config. acceptee */
                Lacc[50],       /* Indice de la ligne contenant une machine dans une conf. acceptee */
                Sacc[50],       /* Indice du site d'une machine dans une configuration acceptee */
                MSm[20][50],    /* Indice de la m/c se trouvant sur un site dans la meil. conf. */
                L[50],          /* Numero de ligne d'une machine */
                S[50],          /* Indice du site d'une machine */
                mml,            /* Nombre maximum de machines par lignes */
                nml[20],        /* Nombre de machines par ligne */
                nb_ligne,       /* Nombre de lignes */
                massa;          /* type de distance utilisee */

float           dis[50][50],    /* Distance inter machines sous contraintes */
                poids[50][50],  /* Poids de la contrainte entre deux machines */
                dist[50][50],   /* Distance inter machines */
                Xacc[50],       /* Position d'une machine dans une config. acceptee */
                X[50],          /* Abscisse d'une machine */
                Xm[50],         /* Abscisse d'une machine dans la configurattion */
                delta,          /* Distance inter site maximale */
                dil,            /* Distance inter-lignes */
                Dili,           /* Distance entre deux lignes ceparees par le vois du transporteur */
                dili,           /* Distance entre deux lignes non ceparees par le vois du transporteur */
                VAG;            /* Largeur du couloire de l'AGV */


mul_lig()
{
        int     i, j, k, l, s, temps,
                rep, MM,
                NbACC, NbREF,
                nb_acc, nb_ref,
                nb_iter;

        float   cout0, cout, delta_cout, coufi,
                proba, eps,
                tempi, tempfi, tempera,
                raigeo,
                Xm[50],
                calc_cuta_mul();                /* Procedure de calcul du cout */

        double  drand48();

        char    repo,
                nom_fich[15];

        void    conf_init_mul(),        /* Procedure de determination d'une configuration initiale */
                trans_elem_mul(),       /* Procedure de selection et de realisation d'une transformation elementaire */
                calc_dist_mul(),        /* Procedure de calcul des distances minimales entre les ma chines */
                ajustement_mul(),       /* Procedure d'ajustement des machines sur les lignes en respectant les contraintes */
                cons_conf_int_mul(),    /* Conservation d'une configuration intermediaire */
                cons_conf_fin_mul(),    /* Conservation d'une configuration finale */
                cont_pos();             /* Contraintes de positionnement des machines */

        FILE    *pt_fich, *fopen(), *fclose();


        printf ("\f\n\n\n\t\t\t\t ***************************************");
        printf ("\n\t\t\t\t *   CONFIGURATION EN MULTI-LIGNE    *");
        printf ("\n\t\t\t\t ***************************************");

        /* Acquisition des contraintes */
```

```
	cont_pos();

	/* Transformation des contraintes de positionnement relatif */
	if ( pres_cont == 1)
	    for (i = 0; i < nb_mach; i++)
	        for (j = i + 1; j < nb_mach; j++) {
	                cont[i][j] = 0;
	                dis[i][j] = 0;
	                poids[i][j] = 0;
	                if ( Couples == 1 )
	                        for (k = 0; k < nb_couple; k++)
	                                if ( Cont[k].M1 == i && Cont[k].M2 == j ) {
	                                        cont[i][j] = Cont[k].type;
	                                        dis[i][j] = Cont[k].Dmami;
	                                        poids[i][j] = Cont[k].poids;
	                                }
	        }


	/* Donnees suplementaires */
	printf ("\n\n\n\t\t\t Donnez le nombre de lignes : ");
	scanf ("%d", &nb_ligne);


	printf ("\n\n\n\t\t\t Donnez le nombre maximal de machines par ligne : ");
	do {
	        scanf ("%d", &mml);
	        if ( nb_ligne * mml < nb_mach ) printf ("\n\n\t\t Ce nombre est insuffisant, donnez une autre reponse : ");
	}
	while (nb_ligne * mml < nb_mach);

	printf ("\n\n\n\t\t\t Le moyen de transport est un : ");
	printf ("\n\n\t\t 1. Pont Roulant 'distance euclidienne',");
	printf ("\n\n\t\t 2. Vehicule Auto-Guide 'distance de Manhattan'");
	printf ("\n\n\t\t 3. Chariot ou Convoyeur 'chemins fixes autours des lignes'");
	printf ("\n\n\t\t\t Donnez votre reponse : ");
	do {
	        scanf ("%d", &massa);
	        if (massa != 1 && massa != 2 && massa != 3) printf ("\n\n\t\t Donnez une autre reponse : ");
	}
	while (massa != 1 && massa != 2 && massa != 3);

	if (massa == 3) {
	        printf ("\n\n\n\t\t Donnez la distance entre deux lignes separees par une voie du transporteur,");
	        printf ("\n\t\t tenez compte de la largeur de la machine la plus large et celle du transporteur : ");
	        do {
	                scanf("%f", &Dili);
	                if (Dili <= wmax) printf ("\n\n\t\t Insuffisant, donnez une autre reponse : ");
	        }
	        while (Dili <= wmax);

	        printf ("\n\n\n\t\t Donnez la distance minimale entre deux machines suivant l'axe des Y : ");
	        do {
	                scanf("%f", &dili);
	                if (dili <= 0) printf ("\n\n\t\t Insuffisant, donnez une autre reponse : ");
	        }
	        while (dili <= 0);
	        dili = dili + 2 * wmax;
	}
	else if (massa == 1 || massa == 2){
	        printf ("\n\n\n\t\t\t Donnez la distance entre les axes des lignes successives : ");
	        do {
	                scanf("%f", &dil);
	                if (dil <= wmax) printf ("\n\n\t\t Insuffisant, donnez une autre reponse : ");
	        }
	        while (dil <= wmax);
	        if (massa == 2) {
	                printf ("\n\n\t\t Donnez la largeure du couloire de passage d'AGV : ");
	                scanf ("%f", &VAG);
	        }
	}

	/* Construction d'une configuration initiale */
	printf ("\n\n\n\t\t Vous pouvez : ");
	printf ("\n\n\t\t\t 1. Donnez une confuguration initiale ");
	printf ("\n\n\t\t\t 2. Faire construire la configuration initiale par le systeme ");
	printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
	do {
	        scanf("%d", &rep);
	        if ( rep != 1 && rep != 2 )      printf ("\n\n\t\t Repondez par 1 ou 2 : ");
	}
	while ( rep != 1 && rep != 2 );
	if ( rep == 1 ) {
	        for (i = 0; i < nb_ligne; i++) {
	                nml[i] = 0;
	                for (j = 0; j < nb_mach; j++) S[j] = -1;
	                for (j = 0; j < nb_mach; j++) MS[i][j] = -1;
	        }
	        /* Pour chaque machine */
	        for (i = 0; i < nb_mach; i++) {
	                printf ("\n\n\n\t\t Machine %d :", i+1);
	                do {
	                        printf ("\n\n\t\t\t Donnez son No de ligne : ");
	                        do {
	                                scanf ("%d", &l);       l = l;
	                                if (l > nb_ligne) printf ("\n\n\t\t Donnez une autre reponse : ");
	                                else if (nml[l] > mml)  printf ("\n\n\t\t Cette ligne est plaine, donnez une autre reponse : ");
	                        }
	                        while (l > nb_ligne);

	                        printf ("\n\n\t\t\t Donnez son No de site sur la ligne %d : ", l+1);
```

```
                        do {
                                scanf ("%d", &s);       s - 1;
                                if (s > nml) printf ("\n\n\t\t Donnez une autre reponse : ");
                        }
                        while (s > nml);
                }
                /* Verification si le site est libre */
                while ( MS[l][s] != -1 );
                nml[l]++;
                MS[l][s] = i;
                S[i] = s;
        }

        /* Elimination des sites non occupes */
        MM = nml;
        for (i = 0; i < nb_ligne; i++)
                for (j = 0; j < MM; j++)
                        if ((MS[i][j] == -1) && (j != MM - 1))
                                for (k = j + 1; k < MM; k++) {
                                        MS[i][k-1] = MS[i][k];
                                        MM--;
                                }

        /* Operation d'ajustement de la configuration */
        ajustement_mul();

        /* Calcul des distances inter-machines */
        calc_dist_mul();

        /* Calcul de la valeur du cout */
        cout0 = calc_cuta_mul();
}
else {
        /* Constuction d'une configuration initiale par le system */
        conf_init_mul();

        /* Ajustement */
        ajustement_mul();

        /* Calcul des distances inter machines */
        calc_dist_mul();

        /* Calcul de la valeur du cout */
        cout0 = calc_cuta_mul();

        printf ("\n\n\n\t\t LE COUT DE CETTE CONFIGURATION EST : %f", cout0);
}
/* Visualisation de la configuration initiale */
printf ("\f\n\n\n\t\t LA CONFIGURATION INITIALE \n\n\n\t");
for(i = 0; i < nb_ligne; i++) {
        printf ("\n\n\t Ligne %d : ", i+1);
        for (j = 0; j < nml[i]; j++)
                printf ("[%d  <%f>]   ", MS[i][j] + 1, X[MS[i][j]]);
}
/* Acquisition des parametres de controle du recuit simule */
printf ("\n\n\n\t\t Vous pouvez : ");
printf ("\n\n\t\t\t 1. Donnez les parametres de controle du R.S ");
printf ("\n\n\t\t\t 2. Les prendre dans un fichier ");
printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
do {
        scanf("%d", &rep);
        if ( rep != 1 && rep != 2 )     printf ("\n\n\t\t Repondez par 1 ou 2 : ");
}
while ( rep != 1 && rep != 2 );
if ( rep == 1 ) {
        printf ("\n\n\t\t Voulez vous sauvegarder ses donnees dans un fichier : ");
        scanf ("%s", &repo);
        if (repo == 'o') {
                printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'paco...' : ");
                scanf ("%s", nom_fich);
                pt_fich = fopen(nom_fich, "w");

                printf ("\n\n\t\t Donnez la temperature initiale : ");
                scanf ("%f", &tempi);
                fprintf(pt_fich,"%f",tempi);

                printf ("\n\n\t\t Donnez la temperature finale : ");
                scanf ("%f", &tempfi);
                fprintf(pt_fich,"\n%f",tempfi);

                printf ("\n\n\t\t Donnez la raison geometrique : ");
                scanf ("%f", &raigeo);
                fprintf(pt_fich,"\n%f",raigeo);

                printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                scanf ("%d", &nb_acc);
                fprintf(pt_fich,"\n%d",nb_acc);

                printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                scanf ("%d", &nb_ref);
                fprintf(pt_fich,"\n%d",nb_ref);

                fclose (pt_fich);
        }
        else {
                printf ("\n\n\t\t Donnez la temperature initiale : ");
                scanf ("%f", &tempi);

                printf ("\n\n\t\t Donnez la temperature finale : ");
                scanf ("%f", &tempfi);
```

```
                printf ("\n\n\t\t Donnez la raison geometrique : ");
                scanf ("%f", &raigeo);

                printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
                scanf ("%d", &nb_acc);

                printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
                scanf ("%d", &nb_ref);
        }
    }
    else {
        printf ("\n\n\n\t\t Donnez le nom du fichier des parametres de controle du R.S. 'paco...' : ");
        scanf ("%s", nom_fich);
        pt_fich = fopen(nom_fich, "r");

        fscanf (pt_fich,"%f",&tempi);
        fscanf (pt_fich,"%f",&tempfi);
        fscanf (pt_fich,"%f",&raigeo);
        fscanf (pt_fich,"%d",&nb_acc);
        fscanf (pt_fich,"%d",&nb_ref);

        fclose (pt_fich);
    }

    /* Processus du recuit simule */
    tempera = tempi;        NbACC = NbREF = nb_iter = 0;
    temps = time(0);
    coufi = cout = cout0;
    do {
        do {
            /* Transformation elementaire */
            trans_elem_mul();

            /* Ajustement */
            ajustement_mul();

            /* Calcul des distances */
            calc_dist_mul();

            /* Calcul de la valeur du cout */
            cout = calc_cuta_mul();

            /* Calcul de la variation du cout */
            delta_cout = cout - coufi;

            /* Test d'acceptation de Metropolis */
            if ( delta_cout <= 0 ) {
                /* Conservation de cette configuration res. intermediaire */
                cons_conf_int_mul();
                NbACC++;
                /* Conservation de cette configuration res. final */
                cons_conf_fin_mul();
                coufi = cout;
                for (i = 0; i < nb_ligne; i++) {
                    for (j = 0; j < nml[i]; j++) {
                        MSm[i][j] = MS[i][j];
                        Xm[MS[i][j]] = X[MS[i][j]];
                    }
                }
            }
            else if (delta_cout > 0) {
                srand48(time(0));
                do eps = drand48(); while (eps > 1.0);
                proba = exp(-delta_cout / tempera);
                if (proba > eps) {
                    /* Conservation de cette config. res. intermediaire */
                    cons_conf_int_dou();
                    NbACC++;
                }
                else NbREF++;
            }
            nb_iter++;
            printf ("\n Iteration No %3d|\t Cout : %f|\t cout de la meilleur iter. : %f|\t temps ecoule : %d",
                nb_iter, cout, coufi, time(0) - temps);
        }
        while ((NbACC < nb_acc)  && (NbREF < nb_ref));
        tempera = tempera * raigeo;
    }
    /* Test d'arret */
    while (tempera > tempfi);


    /* Resultat */
    printf ("\n\n\n\t\t Le nombre total d'iterations est : %d", nb_iter);
    printf ("\n\n\t\t Le temps de traitement est : %d", time(0) - temps);
    printf ("\n\n\t\t Le cout de la configuration initiale est : %f", cout0);
    printf ("\n\n\t\t Le cout de la configuration finale est : %f", coufi);
    printf ("\n\n\t\t Le taux d'amelioration est de : %5.2f pour-cent", (cout0 - coufi) * 100 / cout0 );
    printf ("\n\n\t\t La sequence des machines de la configuration finale est la suivante :\n\n\t");
    for (i = 0 ; i < nb_ligne; i++) {
        printf ("\n\n\t Ligne %d : ", i+1);
        for (j = 0 ; j < nml[i]; j++)
            printf (" [%d <%8.2f>]   ", MSm[i][j] + 1, Xm[MSm[i][j]]);
    }
}

/* ########################################################################################## */

/* ********************************************************** */
```

```
                        /* *    Procedure de construction d'une configuration      * */
                        /* *   initiale pour une configuration en multi lignes     * */
                        /* ********************************************************** */
void conf_init_mul()
{
        int     i, j, k, l, s, flag, nb_m_insta, MM, cont;
        double  Xdr, drand48();

        srand48(time(0));

        /* Initialisation */
        nb_m_insta = 0;
        for (i = 0; i < nb_ligne; i++) {
                nml[i] = 0;
                for (j = 0; j < mml; j++)
                        MS[i][j] = -1;
        }
        for (i = 0; i < nb_mach; i++) {
                MInsta[i] = -1;
                L[i] = -1;
                S[i] = -1;
        }

        /* Affectation des machines a position definitive vers leur sites */
        if (pres_cont == 1 && M_def == 1)
                for (i = 0; i < nb_m_def; i++) {
                        L[Mdef[i]] = Ldef[Mdef[i]];
                        S[Mdef[i]] = Sdef[Mdef[i]];
                        MS[L[Mdef[i]]][S[Mdef[i]]] = Mdef[i];
                        nml[L[Mdef[i]]]++;
                        MInsta[i] = Mdef[i];
                        nb_m_insta++;
                }

        /* Affectation aleatoire des machines restantes */
        for (i = 0; i < nb_mach; i++) {
                if ( S[i] == -1 ) {
                        /* La ligne */
                        do {
                                Xdr = drand48();
                                l = (int)(Xdr * (float)nb_ligne );
                                cont = 0;
                                /* La position sur la ligne */
                                do {
                                        Xdr = drand48();
                                        s = (int)(Xdr * (mml+1));
                                        cont++;
                                }
                                while ( (s > mml) || (cont < 5 && MS[l][s] != -1) );
                        }
                        while ( (MS[l][s] != -1) || (l >= nb_ligne) || (nml[l] > mml) );
                        L[i] = l;
                        S[i] = s;

                        nml[L[i]]++;
                        MS[L[i]][S[i]] = i;
                        MInsta[nb_m_insta] = i;
                        nb_m_insta++;
                }
        }

        for (i = 0; i < nb_ligne; i++) {
                printf ("\n\n\t Ligne %d : ", i + 1);
                for (j = 0; j < mml; j++)       printf (" %d  ", MS[i][j]);
        }

        /* Elimination des sites non occupes */
        for (i = 0; i < nb_ligne; i++) {
                MM = mml;
                do {
                   flag = 0;
                   for (j = 0; j < MM; j++) {
                        if (MS[i][j] == -1) {
                                for (k = j + 1; k < MM; k++)
                                        MS[i][k-1] = MS[i][k];
                                MM--;
                                flag == 1;
                        }
                        if (flag == 1)  break;
                   }
                }
                while ( MM > nml[i] );
                for (j = 0; j < nml[i]; j++)
                        S[MS[i][j]] = j;
        }

        for (i = 0; i < nb_ligne; i++) {
                printf ("\n\n\t\t Ligne %d : ", i + 1);
                for (j = 0; j < nml[i]; j++)    printf (" %d  ", MS[i][j]);
        }
}

                /* ########################################################################### */

                        /* ********************************************************** */
                        /* *    Procedure de realisation d'une transformation       * */
                        /* *            elementaire de reafectation                 * */
                        /* ********************************************************** */

void trans_elem_mul()
{
```

```
int     trans, flag, Ma, i, K,
        l, l1, l2, ml, m2,
        s, si, flag1;

double  X, drand48();

/* Choix de la transformation elementaire */
do {
        X = drand48();
        trans = (int)(X * 2.0);
}
while (trans > 1);

/* Transfert */
if ( trans == 0 && (nb_mach < nb_ligne * mml) ) {
    flag = 0;
    do {
        /* Tirrage aleatoire d'une machine a transferer */
        do {
                X = drand48();
                Ma = (int)(X * (float)nb_mach);
        }
        while (Ma >= nb_mach);

        /* Tirrage aleatoire d'une ligne de destination */
        do {
                X = drand48();
                l = (int)(X * (float)nb_ligne);
        }
        while ( (l >= nb_ligne) || (nml[l] >= mml) );

        /* Transfert sur la meme ligne */
        if (l == L[Ma]) {
                /* Choix aleatoire du site sur cette ligne */
                do {
                        X = drand48();
                        s = (int)(X * (float)mml);
                }
                while ((s >= mml) || (s == S[Ma]));

                /* Transfert vers un site libre */
                if ( (nml[l] < mml) && (s >= nml[l]) ) {
                        for (i = S[Ma] + 1; i < nml[l]; i++)
                                MS[l][i-1] = MS[l][i];
                        for (i = S[Ma]; i < nml[l]; i++)
                                S[MS[l][i]]--;

                        S[Ma] = nml[l]-1;
                        MS[l][nml[l]-1] = Ma;
                }

                /* Transfert vers un site occupe */
                else if (s < nml[l]) {
                        si = S[Ma];
                        if (si > s) {
                                for (i = si; i > s; i--)
                                        MS[l][i] = MS[l][i-1];
                                for (i = si; i > s; i--)
                                        S[MS[l][i]] = i;
                                MS[l][s] = Ma;
                                S[Ma] = s;
                        }
                        else if (s > si) {
                                for (i = si; i < s; i++)
                                        MS[l][i] = MS[l][i+1];
                                for (i = si; i < s; i++)
                                        S[MS[l][i]] = i;
                                MS[l][s] = Ma;
                                S[Ma] = s;
                        }
                }
        }

        /* Transfert vers une autre ligne */
        else {
                l1 = L[Ma];
                l2 = l;
                if (nml[l2] < mml) {
                        L[Ma] = l2;
                        /* Extraction de la m/c de sa ligne initiale */
                        s = S[Ma];
                        for (i = s + 1; i < nml[l1]; i++)
                                MS[l1][i-1] = MS[l1][i];
                        for (i = s; i < nml[l1]; i++)
                                S[MS[l1][i]]--;
                        nml[l1]--;

                        /* Placement de la m/a sur se nouvelle ligne */
                        /* ***************************************** */
                        /* selection aleatoire d'un site sur la nouvelle ligne */
                        do {
                                X = drand48();
                                s = (int)(X * (float)mml);
                        }
                        while (s >= mml);
                        /* Cas d'un transfert vers un site libre */
                        if ((nml[l2] < mml) && (s >= nml[l2])) {
                                S[Ma] = nml[l2];
                                MS[l2][nml[l2]] = Ma;
                        }
```

```
                        /* Cas d'un transfert vers un site occupe */
                        else if (s < nml[l2]) {
                                for (i = nml[l2]; i > s; i--)
                                        MS[l2][i] = MS[l2][i-1];
                                for (i = nml[l2]; i > s; i--)
                                        S[MS[l2][i]] = i;
                                S[Ma] = s;
                                MS[l2][s] = Ma;
                        }
                        nml[l2]++;
                }
                else    flag = 1;
        }
        if ( flag == 0) printf ("\n Transfert de la Machine %d vers le Site %d de la Ligne %d", Ma+1, S[Ma]+1, L[Ma]+1);
    }
    while ( flag == 1 );
}

/* Permutation */
else {
        /* Tirrage aleatoire des deux machines a permuter */

        /* 1ere machine */
        do {
                flag1 = 0;
                do {
                        do {X = drand48();}  while ( X > 1.0 );
                        m1 = (int)(X * (double)nb_mach);
                }
                while ( m1 > nb_mach - 1 );

                if (nb_m_def != 0)
                        for ( i = 0; i < nb_m_def; i++) {
                                if (m1 == Mdef[i])      flag1 = 1;
                                if (flag1 == 1) break;
                        }
        }
        while (flag1 == 1);

        /* 2eme machine */
        do {
                flag1 = 0;
                do {
                        do {X = drand48();}  while ( X > 1.0 );
                        m2 = (int)(X * (double)nb_mach );
                }
                while ( m2 > nb_mach - 1 );

                if (nb_m_def != 0)
                        for ( i = 0; i < nb_m_def; i++) {
                                if (m2 == Mdef[i])      flag1 = 1;
                                if (flag1 == 1) break;
                        }
        }
        while (flag1 == 1 || m2 == m1);

        /* Cas d'une permutation sur une meme ligne */
        if ( L[m1] == L[m2] ) {
                K = MS[L[m1]][S[m1]]; MS[L[m1]][S[m1]] = MS[L[m2]][S[m2]]; MS[L[m2]][S[m2]] = K;
                K = S[m1]; S[m1] = S[m2]; S[m2] = K;
        }

        /* Cas d'une permutation d'une ligne a une autre */
        else if ( L[m1] != L[m2] ) {
                K = MS[L[m1]][S[m1]]; MS[L[m1]][S[m1]] = MS[L[m2]][S[m2]]; MS[L[m2]][S[m2]] = K;
                K = L[m1]; L[m1] = L[m2]; L[m2] = K;
                K = S[m1]; S[m1] = S[m2]; S[m2] = K;
        }
        printf ("\n\n Permutation des machines %d et %d.", m1+1, m2+1);
}
}


        /* ############################################################################ */

                /* ************************************************************ */
                /* *    Procedure d'ajustement apres une transformation       * */
                /* *                        elementaire                       * */
                /* ************************************************************ */

void ajustement_mul()
{
        int     i, j, k, a, b;
        float   delta, del, depo, dep, dimi;

        /* Calcul du Delta */
        del = 0;
        for (i = 0; i < nb_mach; i++)
                if (lm[i] > del)        del = lm[i];
        delta = del;
        if (dist_min >= dil)            del = dist_min;
        else if (dil > dist_min)        del = dil;
        if ( Couples == 1 )
                for (i = 0; i < nb_mach; i++)
                        for (j = i + 1; j < nb_mach; j++)
                                if (cont[i][j] == 2)
                                        if (dis[i][j] > del)    del = dis[i][j];
        delta = delta + del;

        /* Initialisation */
        for (i = 0; i < nb_ligne; i++)
```

```
                for (j = 0; j < nml[i]; j++)
                        X[MS[i][j]] = j * delta;

        if (dist_min <= VAG && massa == 2) dimi = VAG;
        else dimi = dist_min;

        /* Processus d'ajustement */
        for (k = 0; k < nb_ligne; k++)
                for (i = 1; i < nml[k]; i++){
                        a = MS[k][i];
                        dep = 1.E+5;
                        for (j = i - 1; j >= 0; j--) {
                                b = MS[k][j];
                                if (j == i - 1) {
                                        if ( (Couples == 1) && (cont[b][a] == 2) ) {
                                                if (dis[b][a] >= dimi)
                                                        depo = X[a] - X[b] - lm[b] - dis[b][a];
                                                else    depo = X[a] - X[b] - lm[b] - dimi;
                                        }
                                        else    depo = X[a] - X[b] - lm[b] - dimi;
                                }
                                else {
                                        if ( (Couples == 1) && (cont[b][a] == 2) )
                                                depo = X[a] - X[b] - lm[b] - dis[b][a];
                                }
                                if (depo <= dep)        dep = depo;
                        }
                        X[a] = X[a] - dep;
                }
}


                /* ########################################################################### */

                        /* ********************************************************** */
                        /* *    Procedure de calcul des distances inter-machines    * */
                        /* *                       apres ajustement                 * */
                        /* ********************************************************** */


void calc_dist_mul()  /* Il y a du travail ici : pour l'instant on suppose qu'il n'y a pas de contournements */
{
        int     i, j, k, k1, k2, K1, K2;
        float   deltaX, deltaY, LC, long_cell, DD1, DD2;

        /* Distances entre les machines d'une meme ligne */
        /* ********************************************* */
        for (k = 0; k < nb_ligne; k++) {
                        for (i = 0; i < nml[k] - 1; i++)
                                for (j = i+1; j < nml[k]; j++)
                                        dist[MS[k][i]][MS[k][j]] = dist[MS[k][j]][MS[k][i]] = (X[MS[k][j]] - X[MS[k][i]])
                                                                + ( lm[MS[k][j]] - lm[MS[k][i]] ) / 2 + dil;
        }

        /* Distance entre les machines de lignes differentes */
        /* ************************************************* */
        if (massa == 1 || massa == 2) {
           for ( k1 = 0; k1 < nb_ligne - 1; k1++)
                for (k2 = k1 + 1; k2 < nb_ligne; k2++)
                        for (i = 0; i < nml[k1]; i++)
                                for (j = 0; j < nml[k2]; j++) {
                                   if (F[i][j] != 0 || F[j][i] != 0) {
                                        if (X[MS[k1][i]] >= X[MS[k2][j]]) {
                                                deltaX = (X[MS[k1][i]] - X[MS[k2][j]]) + (lm[MS[k1][i]] - lm[MS[k2][j]]) / 2;
                                                deltaY = (k2 - k1) * dil;
                                                if (massa == 2)
                                                        dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = deltaX + deltaY;
                                                else if (massa == 1)
                                                        dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = deltaX * deltaX
                                                                                                + deltaY * deltaY;
                                        }
                                        else if (X[MS[k2][j]] > X[MS[k1][i]]) {
                                                deltaX = (X[MS[k2][j]] - X[MS[k1][i]]) + (lm[MS[k2][j]] - lm[MS[k1][i]]) / 2 ;
                                                deltaY = (k2 - k1) * dil ;
                                                if (massa == 2)
                                                        dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = deltaX + deltaY;
                                                else if (massa == 1)
                                                        dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = deltaX * deltaX
                                                                                                + deltaY * deltaY;
                                        }
                                   }
                                   else dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = 0;
                                }
        }
        else if (massa == 3) {
                /* Calcul de la dimension de la cellule le long de l'axe X */
                long_cell = 0;
                for (i = 0; i < nb_ligne; i++) {
                        LC = X[MS[i][nml[i]-1]] + lm[MS[i][nml[i]-1]];
                        if ( LC > long_cell )   long_cell = LC;
                }

                /* Calcul des longeurs des chemins entre les cellules */
                for ( k1 = 0; k1 < nb_ligne - 1; k1++)
                    for (k2 = k1 + 1; k2 < nb_ligne; k2++)
                        for (i = 0; i < nml[k1]; i++)
                            for (j = 0; j < nml[k2]; j++) {
                                if (F[i][j] != 0 || F[j][i] != 0) {

                                        /* Calcul de delta Y */
                                        if ( fmod((float)k1, 2.0) == 0 ) K1 = k1 / 2;
```

```
                                else K1 = (k1 + 1) / 2;
                                if ( fmod((float)k2, 2.0) == 0 ) K2 = k2 / 2;
                                else K2 = (k2 + 1) / 2;
                                deltaY = fabs((float)(K1 - K2)) * (Dili + dili);

                                /* Calcul du chemin le plus court */
                                DD1 = deltaY + (X[MS[k2][j]] + X[MS[k1][i]]) + (lm[MS[k2][j]] + lm[MS[k1][i]]) / 2 + Dili;
                                DD2 = deltaY + 2 * long_cell - (X[MS[k2][j]] + X[MS[k1][i]]) -
                                        (lm[MS[k2][j]] + lm[MS[k1][i]]) / 2 + Dili;
                                if (DD1 > DD2) dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = DD2;
                                else dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = DD1;
                        }
                        else dist[MS[k1][i]][MS[k2][j]] = dist[MS[k2][j]][MS[k1][i]] = 0;
                }
        }

}

        /* ########################################################################## */

                /* ********************************************************** */
                /* *      Procedure de conservation d'une configuration     * */
                /* *                  intermediaire acceptee                * */
                /* ********************************************************** */

void cons_conf_int_mul()
{
        int     i, k;


        for (i = 0; i < nb_mach; i++) {
                /* Le lieu */
                Lacc[i] = L[i];
                Sacc[i] = S[i];
                /* La position */
                Xacc[i] = X[i];
        }
        for (k = 0; k < nb_ligne; k++) {
                for ( i = 0; i < nml[k]; i++)
                        MSacc[k][i] = MS[k][i];
        }
}

        /* ########################################################################## */

                /* ********************************************************** */
                /* *      Procedure de conservation d'une configuration     * */
                /* *                  qui ameliore le cout                  * */
                /* ********************************************************** */

void cons_conf_fin_mul()
{

        int     i, k;
        FILE    *pt_fich1, *fopen(), *fclose();

        pt_fich1 = fopen("conf_bon", "w");

        fprintf (pt_fich1, "\t\t\t ****   LES LIEUX ET POSITIONS DES MACHINES   ****\n\n");
        for (k = 0; k < nb_ligne; k++) {
                fprintf (pt_fich1, "\n Ligne %d : \t", k + 1);
                for (i = 0; i < nb_mach; i++)
                        MSm[k][i]++;
                        fprintf (pt_fich1, " [%d   <%f>]  ", MSm[k][i], Xm[MSm[k][i]]);
        }

}
        /* ########################################################################## */
                /* ********************************************************** */
                /* *    Procedure de calcul du cout d'une configuration     * */
                /* ********************************************************** */

float calc_cuta_mul()
{
        int     i, j;
        float   cout, disou,
                Dx, Dy;

        cout = 0;
        for (i = 0; i < nb_mach; i++) {
                for (j = 0; j < nb_mach; j++) {
                        if (i != j) {
                                /* Calcul de la distance euclidienne */
                                Dx = fabs(X[S[i]] - X[S[j]]);
                                Dy = fabs(L[i] - L[j]) * dil;
                                disou = sqrt(Dx * Dx + Dy * Dy);
                                /* Cas ou les deux machines sont soumises a une contrainte de positionnement relatif */
                                if ( (Couples == 1) && ((cont[i][j] == 1 && disou > dis[i][j]) ||
                                                        (cont[i][j] == 2 && disou < dis[i][j])) )
                                        cout = cout + F[i][j] * dist[i][j] + (0.5 * poids[i][j] * Fmax) * disou ;

                                /* Cas ou les deux machines ne sont pas soumises a une contrainte */
                                else if (Couples == 0 || ((Couples == 1) && (cont[i][j] == 0)) ||
                                        ((Couples == 1) && ( (cont[i][j] == 1 && disou <= dis[i][j])
                                        || (cont[i][j] == 2 && disou >= dis[i][j]) )) )
                                        cout = cout + F[i][j] * dist[i][j];
                        }
                }
        }
        return cout;
}               /* ***************************   Fin   **************************** */
```

10. Programme de construction d'une configuration quelconque imposée par l'utilisateur

```
                    /* ********************************************************** */
                    /* *    Sous-programme de placement de cellules sur une     * */
                    /* *          configuration en FREE-CONFIGURATION           * */
                    /* *    ---------------------------------------------     * */
                    /* *                              par  T. HAMANN            * */
                    /* ********************************************************** */


#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include "def.h"

extern int      Q[50],          /* Quantite a produire */
                U[50],          /* Taille du lot transporte */
                M[50],          /* Nombre de machines du routage */
                Ind[50],        /* Indices des machines */
                nb_mach,        /* Nombre de machine */
                nb_prod,        /* Nombre de produits */
                nb_m_def,       /* Nombre de machines instalees definitivement */
                Mdef[20],       /* Indice d'une machine a placee definitivement */
                Sdef[20],       /* Indice d'un site occupe definitivement */
                pres_cont,      /* Indice d'existance des contraintes */
                M_def,          /* Indice d'existance des contraintes de positionnement definitif */
                Couples,        /* Indice d'existance des contraintes de positionnement relatif */
                nb_couple;      /* nombre de couples de machines sous contrainte */

extern float    QsU[50],        /* Rapport Q / U */
                F[50][50],      /* Matrice des flux */
                pm[50],         /* Poids machine */
                wm[50],         /* Largeur machine */
                lm[50],         /* Longueur machine */
                Fmax,           /* Le flux maximum */
                wmax,           /* largeur de la cellule la plus large */
                lmax,           /* longueur de la cellule la plus longue */
                dist_min;       /* Distance inter-machines minimale */

extern contrainte Cont[50];     /* Contrainte de positionnement relatif entre les cellules */

int             mS[50],         /* Indice de la machine se trouvant sur un site */
                MInsta[50],     /* Indice de la machine instalee sur un site */
                MSacc[50],      /* Indice de la machine se trouvant sur un site dans une configuration acceptee */
                Sacc[50],       /* Indice du site d'une machine dans une configuration acceptee */
                MSm[50],        /* Indice de la m/c se trouvant sur un site dans la meilleure configuration */
                cont[50][50],   /* Type de contrainte entre deux cellules lorsqu'elle existe */
                ori[50],        /* Orientation d'une machine */
                transpo,        /* Type de moyen de transport utilise */
                nb_site,        /* Nombre de sites de la cellules */
                S[50];          /* Indice du site d'une machine */

float           dist[50][50],   /* Distance inter-machines */
                Xacc[50],       /* Abscisse d'une machine dans une configuration acceptee */
                Yacc[50],       /* Ordonnee d'une machine dans une configuration acceptee */
                X[50],          /* Abscisse du centre d'une machine */
                Y[50],          /* Ordonnee du centre d'une machine */
                Xm[50],         /* Abscisse d'une machine dans la meilleure configuration */
                Ym[50],         /* Ordonnee d'une machine dans la meilleure configuration */
                Xsite[50],      /* Abscisse d'un site */
                Ysite[50],      /* Ordonnee d'un site */
                dis[50][50],    /* Distance max/min entre deux machines sous contrainte */
                poids[50][50];  /* Poids d'une contrainte de positionnement relatif */

free_lig()
{
        int     i, j, k, temps,
                rep, s,
                flag,
                NbACC, NbREF,
                nb_acc, nb_ref,
                nb_iter, orio,
                nb_m_insta,
                verifi_cont();

        float   cout0, cout, delta_cout, coufi,
                proba, eps,
                tempi, tempfi, tempera,
                raigeo,
                calc_cuta_free();

        double  drand48();

        char    repo,
                nom_fich[15];

        void    acq_do_sup(),
                conf_init_free(),
                trans_elem_free(),
                calc_dist_free(),
                cons_conf_int_fr(),
                cons_conf_fin_fr(),
                cont_pos();

        FILE    *pt_fich, *fopen(), *fclose();


        printf ("\f\n\n\n\t\t\t\t ******************************");
        printf ("\n\t\t\t\t *    CONFIGURATION LIBRE     *");
        printf ("\n\t\t\t\t ******************************");

        /* Acquisition des donnees suplementaires */
```

```
	acq_do_sup();

	/* Acquisition des contraintes */
	cont_pos();

	/* Transformation des contraintes de positionnement relatif */
	for (i = 0; i < nb_mach; i++)
		for (j = i + 1; j < nb_mach; j++) {
			cont[i][j] = 0;
			dis[i][j] = 0;
			poids[i][j] = 0;
			if ( Couples == 1 )
				for (k = 0; k < nb_couple; k++)
					if ( Cont[k].M1 == i && Cont[k].M2 == j ) {
						cont[i][j] = Cont[k].type;
						dis[i][j] = Cont[k].Dmani;
						poids[i][j] = Cont[k].poids;
					}
		}

	/* Construction d'une configuration initiale */
	printf ("\f\n\n\n\n\t CONFIGURATION INITIALE");
	printf ("\n\t ************************");

	printf ("\n\n\n\t\t Vous pouvez : ");
	printf ("\n\n\t\t\t 1. Donnez une confuguration initiale ");
	printf ("\n\n\t\t\t 2. Faire construire la configuration initiale par le systeme ");
	printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
	do {
		scanf("%d", &rep);
		if ( rep != 1 && rep != 2 )	printf ("\n\n\t\t Repondez par 1 ou 2 : ");
	}
	while ( rep != 1 && rep != 2 );

	if ( rep == 1 ) {
			/* Donner une configuration initiale manuellement */
		/* Initialisation */
		for (i = 0; i < nb_site; i++)
			mS[i] = -1;
		for (i = 0; i < nb_mach; i++)
			S[i] = -1;

		/* Affectation des machines a position definitive vers leur sites */
		if (pres_cont == 1 && M_def == 1)
			for (i = 0; i < nb_m_def; i++) {
				S[Mdef[i]] = Sdef[Mdef[i]];
				mS[S[Mdef[i]]] = Mdef[i];
				MInsta[i] = Mdef[i];
				nb_m_insta++;
			}

		printf ("\n\n\n\n REMARQUES:");
		printf ("\n\n\t Rq. 1 - Tenir compte de contraintes de positionnement definitif et relatif, si elles existent.");
		printf ("\n\n\t Rq. 2 - L'orientation de la machine est definie par rapport a son orientation initiale dans les donnees,");
		printf ("\n\t Repondez par: \n\t\t 0. pas de rotation \n\n\t\t 1. pour une rotation de 90 deg.");
		printf ("\n\n\t\t 2. pour une rotation de 180 deg.");
		printf ("\n\n\t\t 3. pour une rotation de 270 deg.");
		printf ("\n\n\t Rq. 3 - Eviter les chevauchements.\n\n\n\n");

		for (i = 0; i < nb_mach; i++) {
			printf ("\n\n\t Machine %d : ", i + 1);
			flag = 0;
			/* Le site de la machine */
			if ( S[i] == -1 ) {
				printf ("\n\n\t\t Donnez l'indice de son site : ");
				do {
					scanf ("%d", &s);
					s--;
					if ( (s > nb_site) || (s < -1) ) printf ("\n\n\t\t\t Donnez une autre reponse : ");
				}
				while ( (s > nb_site) || (s < -1) );
				S[i] = s;
				mS[s] = i;
				nb_m_insta++;
				flag = 1;
			}
			/* L'orientation de la machine */
			if ( flag == 0 ) printf ("\n\n\t\t Cette machine est instalee definitivement donnez son orientation : ");
			else if ( flag == 1 ) printf ("\n\n\t\t Donnez son orientation : ");
			do {
				scanf ("%d", &orio);
				if ( (orio != 0) && (orio != 1) && (orio != 2) && (orio != 3) )
					printf ("\n\t\t repondez par 0, 1, 2 ou 3 : ");
			}
			while ( (orio != 0) && (orio != 1) && (orio != 2) && (orio != 3) );
			ori[i] = orio; /* Il devrait y avoir ici une verification du non chevauchement */
		}
	}
	else
		/* Constuction d'une configuration initiale par le system */
		conf_init_free();

	/* Calcul des distances inter machines */
	calc_dist_free();

	/* Calcul de la valeur du cout */
	cout0 = calc_cuta_free();


	/* Visualisation de la configuration initiale */
	printf ("\f\n\n\n\t\t LA CONFIGURATION INITIALE  EST :\n");
```

```
	printf ("\n\n\t\t Machine		Site		X		Y		Orientation");
	printf ("\n\t\t ------------------------------------------------------------------------");
	for (i = 0; i < nb_mach; i++)
		printf ("\n\n\t\t	%3d		%3d		%5.2f		%5.2f		%1d",
					i + 1, S[i] + 1, Xsite[S[i]], Ysite[S[i]], ori[i]);

	printf ("\n\n\n\t\t LE COUT DE CETTE CONFIGURATION EST : %f", cout0);

	/* Acquisition des parametres de controle du recuit simule */
	/* ******************************************************* */

	printf ("\n\n\n\t\t LES PARAMETRES DE CONTROLE DU RECUIT SIMULE:");
	printf ("\n\n\t\t Vous pouvez : ");
	printf ("\n\n\t\t\t 1. Donnez les parametres de controle du R.S ");
	printf ("\n\n\t\t\t 2. Les prendre dans un fichier ");
	printf ("\n\n\t\t Faites votre choix '1 ou 2' : ");
	do {
		scanf("%d", &rep);
		if ( rep != 1 && rep != 2 )	printf ("\n\n\t\t Repondez par 1 ou 2 : ");
	}
	while ( rep != 1 && rep != 2 );
	if ( rep == 1 ) {
		printf ("\n\n\t\t Voulez vous sauvegarder ses donnees dans un fichier : ");
		scanf ("%s", &repo);
		if (repo == 'o') {
			printf ("\n\n\n\t\t Donnez le nom du fichier de donnees 'paco...' : ");
			scanf ("%s", nom_fich);
			pt_fich = fopen(nom_fich, "w");

			printf ("\n\n\t\t Donnez la temperature initiale : ");
			scanf ("%f", &tempi);
			fprintf(pt_fich,"%f",tempi);

			printf ("\n\n\t\t Donnez la temperature finale : ");
			scanf ("%f", &tempfi);
			fprintf(pt_fich,"\n%f",tempfi);

			printf ("\n\n\t\t Donnez la raison geometrique : ");
			scanf ("%f", &raigeo);
			fprintf(pt_fich,"\n%f",raigeo);

			printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
			scanf ("%d", &nb_acc);
			fprintf(pt_fich,"\n%d",nb_acc);

			printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
			scanf ("%d", &nb_ref);
			fprintf(pt_fich,"\n%d",nb_ref);

			fclose (pt_fich);
		}
		else {
			printf ("\n\n\t\t Donnez la temperature initiale : ");
			scanf ("%f", &tempi);

			printf ("\n\n\t\t Donnez la temperature finale : ");
			scanf ("%f", &tempfi);

			printf ("\n\n\t\t Donnez la raison geometrique : ");
			scanf ("%f", &raigeo);

			printf ("\n\n\t\t Donnez le nombre d'acceptations par palier de temperature : ");
			scanf ("%d", &nb_acc);

			printf ("\n\n\t\t Donnez le nombre de refus par palier de temperature : ");
			scanf ("%d", &nb_ref);
		}
	}
	else {
		printf ("\n\n\n\t\t Donnez le nom du fichier des parametres de controle du R.S. 'paco...' : ");
		scanf ("%s", nom_fich);
		pt_fich = fopen(nom_fich, "r");

		fscanf (pt_fich,"%f",&tempi);
		fscanf (pt_fich,"%f",&tempfi);
		fscanf (pt_fich,"%f",&raigeo);
		fscanf (pt_fich,"%d",&nb_acc);
		fscanf (pt_fich,"%d",&nb_ref);

		fclose (pt_fich);
	}

	/* Processus du recuit simule */
	tempera = tempi;		NbACC = NbREF = nb_iter = 0;
	temps = time(0);
	coufi = cout = cout0;
	do {
		do {
			/* Transformation elementaire */
			trans_elem_free();

			/* Calcul des distances */
			calc_dist_free();

			/* Calcul de la valeur du cout */
			cout = calc_cuta_free();

			/* Calcul de la variation du cout */
			delta_cout = cout - coufi;

			/* Test d'acceptation de Metropolis */
```

```
                        if ( delta_cout <= 0 ) {
                                /* Conservation de cette configuration comme res. intermediaire */
                                cons_conf_int_fr();
                                NbACC++;
                                /* Verification si l'une des contraintes tres fortes n'est pas verifiees
                                V = verifi_cont_fr();*/
                                /* Conservation de cette configuration res. final
                                cons_conf_fin();*/
                                /* Si toutes les cont. fortes sont verifiees : conservation de la configuration comme bonne
                                if ( V == 0 ) {*/
                                        coufi = cout;
                                        for (i = 0; i < nb_mach; i++) {
                                                MSm[i] = mS[i]; Xm[i] = X[i]; Ym[i] = Y[i];
                                        }
                                /*}*/
                        }
                        else if (delta_cout > 0) {
                                srand48(time(0));
                                do eps = drand48(); while (eps > 1.0);
                                proba = exp(-delta_cout / tempera);
                                if (proba > eps) {
                                        /* Conservation de cette configuration res. intermediaire */
                                        cons_conf_int_fr();
                                        NbACC++;
                                }
                                else {
                                        /* Reprise a partir de le derniere configuration acceptee */
                                        for (i = 0; i < nb_mach; i++) {
                                                /* Le lieu */
                                                S[i] = Sacc[i];
                                                mS[i] = MSacc[i];
                                                /* La position */
                                                X[i] = Xacc[i]; Y[i] = Yacc[i];
                                        }
                                        NbREF++;
                                }
                        }

                        nb_iter++;
                        printf ("\n\n Iteration No %3d|\t Cout : %f|\t cout de la meilleur iter. : %f|\t temps ecoule : %d\n",
                                 nb_iter, cout, coufi, time(0) - temps);
                        /*for (i = 0; i < nb_mach; i++)
                                printf ("[%d <%8.2f, %8.2f>]   ", mS[i] + 1, X[mS[i]], Y[mS[i]]);*/
                }
                while ((NbACC < nb_acc)  && (NbREF < nb_ref));
                tempera = tempera * raigeo;
        }
        /* Test d'arret */
        while (tempera > tempfi);

        /* Resultat */
        printf ("\n\n\n\t\t Le nombre total d'iterations est : %d", nb_iter);
        printf ("\n\n\t\t Le temps de traitement est : %d", time(0) - temps);
        printf ("\n\n\t\t Le cout de la configuration initiale est : %f", cout0);
        printf ("\n\n\t\t Le cout de la configuration finale est : %f", coufi);
        printf ("\n\n\t\t Le taux d'amelioration est de : %5.2f pour-cent", (cout0 - coufi) * 100 / cout0 );
        printf ("\n\n\t\t La sequence des machines de la configuration finale est la suivante :\n\n\t");
        for (i = 0 ; i < nb_mach; i++)
                printf ("%d <%8.2f, %8.2f>   ", MSm[i] + 1, Xm[MSm[i]], Ym[MSm[i]]);

}

        /* ########################################################################################### */

                        /* ********************************************** */
                        /* *    Procedure d'acquisition des donnees     * */
                        /* *                suplementaires              * */
                        /* ********************************************** */

void acq_do_sup()
{
        int     i, rep, repi;
        char    nom_fich[10], repo;
        FILE    *pt_fich, *fopen(), *fclose();


        printf ("\n\n\n\t DONNEES SUPLEMENTAIRES ");
        printf ("\n\t ********************** ");

        printf ("\n\n\n\t\t Vous pouvez : \n\n\t\t\t 1. Fournir les donnees manuellement,");
        printf ("\n\n\t\t\t 2. Prendre les donnees dans un fichier.");
        printf ("\n\n\t\t Donnez votre reponse : ");
        do {
                scanf ("%d", &repi);
                if ( repi != 1 && repi != 2 ) printf ("\n\n Repondez par 1 ou 2 : ");
        }
        while ( repi != 1 && repi != 2 );

        if (repi == 1) {

                printf ("\n\n\n\t Donnez le nombre de sites de la cellule : ");
                scanf ("%d", &nb_site);

                printf ("\n\n\n\t Relativement a un repere orthonorme [dont l'axe des X est dirige de gauche a droite,");
                printf ("\n et l'axe des Y de haut en bas], donnez les coordonnees des sites : ");
                for (i = 0; i < nb_site; i++) {
                        printf ("\n\n\t\t Site No %d :\n\t\t\t Abscisse : ", i + 1);
                        scanf ("%f", &Xsite[i]);
                        printf ("\n\t\t\t Ordonnee : ");
                        scanf ("%f", &Ysite[i]);
                }
```

```
		printf ("\n\n\n\t\t Selectionnez le moyen de transport utilise : ");
		printf ("\n\n\t 1. Pont roulant \n\n\t 2. Vehicule guide, convoyeur et chariot");
		printf ("\n\n Donnez votre reponse : ");
		do {
			scanf ("%d", &rep);
			if (rep != 1 && rep != 2) printf ("\n\n Donnez une autre reponse 1 ou 2 : ");
		}
		while ( rep != 1 && rep != 2);

		if (rep == 1) transpo = 1;
		else if (rep == 2) transpo = 2;

		printf ("\n\n\n Voulez vous sauvegarder ses donnees dans un fichier o/n ? : ");
		do {
			scanf ("%s", &repo);
			if ( repo != 'o' && repo != 'n' ) printf ("\n\t Repondez par o ou n : ");
		}
		while ( repo != 'o' && repo != 'n' );

		if ( repo == 'o' ) {
			printf ("\n\n\n\t Donnez un nom au fichier de sauvegarde 'dosuli...' : ");
			scanf ("%s", nom_fich);

			pt_fich = fopen(nom_fich, "w");

			/* Nombre de sites */
			fprintf (pt_fich,"\n%d", nb_site);

			/* Coordonnees des sites */
			for (i = 0; i < nb_site; i++) {
				fprintf (pt_fich, "\n%f", Xsite[i]);
				fprintf (pt_fich, "\n%f", Ysite[i]);
			}

			/* Type de moyen de transport */
			fprintf (pt_fich, "\n%d", transpo);
			fclose(pt_fich);
		}
	}

	else if ( repi == 2 ) {
		printf ("\n\n\n\t\t Donnez le nom de fichier contenant ses donnees, 'dosuli...' : ");
		scanf ("%s", nom_fich);

		pt_fich = fopen (nom_fich, "r");

		/* Nombre de sites */
		fscanf (pt_fich,"%d", &nb_site);

		/* Coordonnees des sites */
		for (i = 0; i < nb_site; i++) {
			fscanf (pt_fich, "%f", &Xsite[i]);
			fscanf (pt_fich, "%f", &Ysite[i]);
		}

		/* Type de moyen de transport */
		fscanf (pt_fich, "%d", &transpo);
		fclose(pt_fich);
	}
}
	/* ############################################################################################ */

			/* ********************************************************* */
			/* *    Procedure de construction d'une configuration      * */
			/* *       initiale pour une configuration libre           * */
			/* ********************************************************* */
void conf_init_free()
{
	int	i, j, nb_m_insta,
		flag, M, V,
		vrif_che();

	float	Xdr, vg;


	srand48(time(0));

	/* Initialisation */
	nb_m_insta = 0;
	for (i = 0; i < nb_mach; i++) {
		mS[i] = -1;
		MInsta[i] = -1;
	}

	/* Affectation des machines a position definitive vers leur sites */
	/* Dans ce qui suit nous ne fixons que la position l'orientation peut etre changee */
	if (pres_cont == 1 && M_def == 1)
		for (i = 0; i < nb_m_def; i++) {
			S[Mdef[i]] = Sdef[Mdef[i]];
			mS[S[Mdef[i]]] = Mdef[i];
			MInsta[i] = Mdef[i];
			X[i] = Xsite[S[Mdef[i]]];
			Y[i] = Ysite[S[Mdef[i]]];
			nb_m_insta++;
		}

	/* Affectation aleatoire des machines restantes */
	for (i = 0; i < nb_site; i++) {
		/* Si le site n'est pas occupe */
		if ( mS[i] == -1 ) {
```

```
                        do {
                                do {Xdr = drand48();}  while ( Xdr > 1.0 );
                                do {
                                        M = (int)(Xdr * (double)nb_mach );
                                }
                                while ( M > nb_mach - 1 );
                                flag = 0;
                                for (j = 0; j < nb_m_insta; j++) {
                                        if ( M == MInsta[j] ) flag = 1;
                                        if (flag == 1) break;
                                }
                                /* Verification du non chevauchement */
                                if (flag == 0) V = verif_che(M, i);
                                if (V == 1) {
                                        /* Rotation de la machine */
                                        vg = lm[M]; lm[M] = wm[M]; wm[M] = vg;
                                        /* Verification du non chevauchement */
                                        V = verif_che(M, i);
                                }
                                if ( V == 1 ) {
                                        /* Remise de la machine a sa position initiale */
                                        vg = lm[M]; lm[M] = wm[M]; wm[M] = vg;
                                        flag = 1;
                                }
                        }
                        while ( flag == 1 );
                        S[M] = i;
                        mS[i] = M;
                        nb_m_insta++;
                        MInsta[nb_m_insta - 1] = M;
                }
                if ( nb_m_insta == nb_mach ) break;
        }
}

        /* ########################################################################################## */

                        /* ******************************************************* */
                        /* *     Procedure de verification du chevauchement      * */
                        /* *   renvoie 0 s'il y a chevauchement, et 1 si non     * */
                        /* ******************************************************* */

int verif_che(m, s)
int s, m;
{
        int     i, j, k, chev, flag,
                SR[50], sr, ml;

        float   R, DD, DX, DY,
                x1[4], y1[4], x2[4], y2[4];

        /* Calcul du rayon de la zone a risque */

        R = sqrt(wmax * wmax + lmax * lmax);

        /* Determination des sites a risque */
        k = 0;
        for (i = 0; i < nb_site; i++)
                if (i != s) {
                        DX = fabs (Xsite[i] - Xsite[s]);
                        DY = fabs (Ysite[i] - Ysite[s]);
                        DD = sqrt(DX * DX + DY * DY);

                        if ( DD < R )   { SR[k] = i;    k++;}
                }
        /* Verification du non chevauchement avec les machines se trouvant sur ces sites */
        x1[0] = x1[2] = Xsite[s] - lm[m] / 2;   y1[0] = y1[1] = Ysite[s] - lm[m] / 2;
        x1[1] = x1[3] = Xsite[s] - wm[m] / 2;   y1[2] = y1[3] = Ysite[s] + wm[m] / 2;

        flag = 0;
        for ( i = 0; i < k; i++) {
                sr = SR[i];
                ml = mS[sr];
                if ( mS[sr] != -1 ) {
                        x2[0] = x2[2] = Xsite[sr] - lm[ml] / 2; y2[0] = y2[1] = Ysite[sr] - lm[ml] / 2;
                        x2[1] = x2[3] = Xsite[sr] - wm[ml] / 2; y2[2] = y2[3] = Ysite[sr] + wm[ml] / 2;

                        /* Verification si l'un des 4 points de de la 1ere machine se trouve dans la deuxieme cellule */
                        for ( j = 0; j < 4; j++) {
                                if (x1[j] >= x2[0] && x1[j] <= x2[1] && y1[j] <= y2[2] && y1[j] >= y2[0])       flag = 1;
                                if (flag == 1)  break;
                        }
                        if (flag == 1)  break;

                        /* Verification si l'un des 4 points de de la 1ere machine se trouve dans la deuxieme cellule */
                        for ( j = 0; j < 4; j++) {
                                if (x2[j] >= x1[0] && x2[j] <= x1[1] && y2[j] <= y1[2] && y2[j] >= y1[0])       flag = 1;
                                if (flag == 1)  break;
                        }
                        if (flag == 1)  break;

                        /* Verification si les deux machines ne se croisent pas */
                        if ( (x1[0] <= x2[0] && x1[1] >= x2[1]) || (x2[0] <= x1[0] && x2[1] >= x1[1]) )         flag = 1;
                        if (flag == 1)  break;
                }
                if ( flag == 1 )        break;
        }
        if ( flag == 1 )        chev = 1;
        else if ( flag == 0 )   chev = 0;

        return (chev);
```

```
}
        /* ########################################################################################## */

                        /* ******************************************************* */
                        /* *    Procedure de realisation des transformations     * */
                        /* *                      elementaires                   * */
                        /* ******************************************************* */

void trans_elem_free()
{
        int     flag1, i, Ma,
                m1, m2, s, s1, s2,
                chev,trans, V, m, rot1, rot2;

        double  Xx, drand48(), vg;

        srand48(time(0));
    do {
        chev = 0;
        /* Choix de la transformation */
        do {
                Xx = drand48();
                trans = (int)(Xx * 3.0);
        }
        while (trans > 2);

        /* Transfert */
        if ( trans == 0 ) {
                /* Tirrage aleatoire d'une machine a transferer */
                do {
                        Xx = drand48();
                        Ma = (int)(Xx * (float)nb_mach);
                }
                while (Ma >= nb_mach);

                /* Choix aleatoire d'un site libre */
                do {
                        Xx = drand48();
                        s = (int)(Xx * (float)nb_mach);
                }
                while ((s > nb_site) || (s == S[Ma]) || (mS[s] != -1));

                /* Verification du non chevauchement */
                V = verif_che(Ma, s);
                if (V == 0) {
                        /* Transfert */
                        mS[S[Ma]] = -1;
                        S[Ma] = s;
                        mS[s] = Ma;
                        X[Ma] = Xsite[s];       Y[Ma] = Ysite[s];
                }
                else if (V == 1) {
                        /* Rotation de la machine */
                        vg = lm[Ma]; lm[Ma] = wm[Ma]; wm[Ma] = vg;

                        /* Verification du non chevauchement */
                        V = verif_che(Ma, s);
                        if (V == 0) {
                                /* Transfert */
                                mS[S[Ma]] = -1;
                                S[Ma] = s;
                                mS[s] = Ma;
                                X[Ma] = Xsite[s];       Y[Ma] = Ysite[s];
                        }
                        else if (V == 1)
                                /* Remise de la machine a son orientation initiale */
                                vg = lm[Ma]; lm[Ma] = wm[Ma]; wm[Ma] = vg;
                                chev = 1;
                }
        }

        /* Permutation de deux machines */
        /* **************************** */
        else if ( trans == 1 ) {
                /* Choix du couple de machines a permuter */
                /* 1ere machine */
                do {
                        flag1 = 0;
                        do {
                                do {Xx = drand48();}  while ( Xx > 1.0 );
                                m1 = (int)(Xx * (double)nb_mach );
                        }
                        while ( m1 > nb_mach - 1 );

                        if (nb_m_def != 0) {
                                for ( i = 0; i < nb_m_def; i++) {
                                        if (m1 == Mdef[i])      flag1 = 1;
                                        if (flag1 == 1) break;
                                }
                        }
                }
                while (flag1 == 1);

                /* 2eme machine */
                do {
                        flag1 = 0;
                        do {
                                do {Xx = drand48();}  while ( Xx > 1.0 );
                                m2 = (int)(Xx * (double)nb_mach );
                        }
                        while ( m2 > nb_mach - 1 );
```

```
                if (nb_m_def != 0) {
                        for ( i = 0; i < nb_m_def; i++) {
                                if (m2 == Mdef[i])      flag1 = 1;
                                if (flag1 == 1) break;
                        }
                }
        }
        while (flag1 == 1 || m2 == m1);

        /* Enlever les deux machines de leurs sites */
        s1 = S[m1]; mS[s1] = -1;
        s2 = S[m2]; mS[s2] = -1;
        rot1 = rot2 = 0;

        /* Placer m1 sur le site de m2 */
        X[m1] = Xsite[s2];      Y[m1] = Ysite[s2];
        /* Test de chevauchement */
        V = verif_che(m1, s2);
        if ( V == 1 ) {
                /* Rotation de la machine m1 */
                vg = lm[m1]; lm[m1] = wm[m1]; wm[m1] = vg; rot1 = 1;
                /* Test de chevauchement */
                V = verif_che(m1, s2);
                if ( V == 1 ) {
                        /* Permutation impossible: Remise m1 a sa place avec son orientation */
                        X[m1] = Xsite[s1];      Y[m1] = Ysite[s1];
                        vg = lm[m1]; lm[m1] = wm[m1]; wm[m1] = vg; rot1 = 0;
                        mS[s1] = m1; mS[s2] = m2;
                        chev = 1;
                }
        }
        if ( V == 0 ) {
                /* Placer la machine m2 sur le site de m1 */
                X[m2] = Xsite[s1];      Y[m2] = Ysite[s1];
                /* test de chevauchement */
                V = verif_che(m2, s1);
                if ( V == 1 ) {
                        /* Rotation de m2 */
                        vg = lm[m2]; lm[m2] = wm[m2]; wm[m2] = vg; rot2 = 1;
                        /* Test de chevauchement */
                        V = verif_che(m2, s1);
                        if ( V == 1 ) {
                                /* Rotation de m1 */
                                vg = lm[m1]; lm[m1] = wm[m1]; wm[m1] = vg;
                                if ( rot1 == 0 ) rot1 = 1;
                                else rot1 = 0;
                                /* Test du chevauchement avec m1 */
                                V = verif_che(m1, s2);
                                if (V == 0) {
                                        /* test du chevauchement avec m2 */
                                        V = verif_che(m2, s1);
                                        if ( V == 1 ) {
                                                /* Impossible: Remise de m1 et m2 a leur place dans la meme orientation */
                                                X[m1] = Xsite[s1];      Y[m1] = Ysite[s1];
                                                X[m2] = Xsite[s2];      Y[m2] = Ysite[s2];
                                                if (rot1 == 1)  {vg = lm[m1]; lm[m1] = wm[m1]; wm[m1] = vg;}
                                                if (rot2 == 1)  {vg = lm[m2]; lm[m2] = wm[m2]; wm[m2] = vg;}
                                                chev = 1;
                                        }
                                }
                        }
                }
        }
        if ( V == 0 ) {
                mS[s1] = m2;    mS[s2] = m1;
                S[m1] = s2;     S[m2] = s1;
                chev = 0;
        }
}
else if ( trans == 2 ) {
        /* Choix de la machine a tourner */
        do {
                flag1 = 0;
                do {
                        do {Xx = drand48();}  while ( Xx > 1.0 );
                        m = (int)(Xx * (double)nb_mach);
                }
                while ( m > nb_mach - 1 );

                if (nb_m_def != 0) {
                        for ( i = 0; i < nb_m_def; i++) {
                                if (m == Mdef[i])       flag1 = 1;
                                if (flag1 == 1) break;
                        }
                }
        }
        while (flag1 == 1);

        /* Rotation fictive de la machine */
        vg = lm[m]; lm[m] = wm[m]; wm[m] = vg;

        /* Verification du non chevauchement */
        V = verif_che(m, S[m]);

        if (V == 1) {
                /* Retour a l'orientation originale */
                vg = lm[m]; lm[m] = wm[m]; wm[m] = vg;
                chev = 1;
        }
}
```

```
    }
    while (chev == 1);
}

        /* ########################################################################################## */

                        /* ********************************************** */
                        /* *    Procedure de calcul des distances       * */
                        /* *        inter-machines minimales            * */
                        /* ********************************************** */
void calc_dist_free()
{
        int i, j;
        float Dx, Dy;

        for (i = 0; i < nb_mach - 1; i++)
                for (j = i+1; j < nb_mach; j++) {
                        Dx = fabs(X[i] - X[j]);
                        Dy = fabs(Y[i] - Y[j]);
                        if (transpo == 1)
                                dist[mS[i]][mS[j]] = dist[mS[j]][mS[i]] = sqrt( Dx * Dx + Dy * Dy);
                        else if (transpo == 2)
                                dist[mS[i]][mS[j]] = dist[mS[j]][mS[i]] = Dx + Dy;

                }
}

        /* ########################################################################################## */

                        /* ********************************************************** */
                        /* *      Procedure de conservation d'une configuration     * */
                        /* *                 intermediaire acceptee                 * */
                        /* ********************************************************** */
void cons_conf_int_fr()
{
        int     i;


        for (i = 0; i < nb_mach; i++) {
                /* Le lieu */
                Sacc[i] = S[i];
                MSacc[i] = mS[i];
                /* La position */
                Xacc[i] = X[i];
                Yacc[i] = Y[i];
        }
}

        /* ########################################################################################## */

                        /* ********************************************************** */
                        /* *      Procedure de conservation d'une configuration     * */
                        /* *                   qui ameliore le cout                 * */
                        /* ********************************************************** */
void cons_conf_fin_fr()
{

        int     i;
        FILE    *pt_fich1, *fopen(), *fclose();

        pt_fich1 = fopen("conf_bon", "w");

        /*fprintf (pt_fich1, "\t\t\t ****   LES LIEUX ET POSITIONS DES MACHINES   ****\n\n\t\t");*/
        for (i = 0; i < nb_mach; i++)
                fprintf (pt_fich1, " [%d   <%f, %f>]  ", MSm[i] + 1, Xm[MSm[i]], Ym[MSm[i]]);
}

        /* ########################################################################################## */

                        /* ********************************************************** */
                        /* *    Procedure de calcul du cout d'une configuration     * */
                        /* ********************************************************** */
float calc_cuta_free()
{
        int     i, j;
        float   cout, disou,
                Dx, Dy;

        cout = 0;
        for (i = 0; i < nb_mach; i++) {
                for (j = 0; j < nb_mach; j++) {
                        if (i != j) {
                                /* Calcul de la distance euclidienne */
                                Dx = fabs(X[S[i]] - X[S[j]]);
                                Dy = fabs(Y[S[i]] - Y[S[j]]);
                                disou = sqrt(Dx * Dx + Dy * Dy);
                                /* Cas ou les deux machines sont soumises a une contrainte de positionnement relatif */
                                if ( (Couples == 1) && ((cont[i][j] == 1 && disou > dis[i][j]) ||
                                                        (cont[i][j] == 2 && disou < dis[i][j])) )
                                        cout = cout + F[i][j] * dist[i][j] + (0.5 * poids[i][j] * Fmax) * disou ;

                                /* Cas ou les deux machines ne sont pas soumises a une contrainte */
                                else if (Couples == 0 || ((Couples == 1) && (cont[i][j] == 0)) ||
                                        ((Couples == 1) && ( (cont[i][j] == 1 && disou <= dis[i][j])
                                        || (cont[i][j] == 2 && disou >= dis[i][j]) )) )
                                        cout = cout + F[i][j] * dist[i][j];
                        }
                }
        }
        return cout;
```

```
}
        /* ########################################################################################## */
                        /* ********************************************************************** */
                        /* *    Verification des contraintes fortes de positionnement           * */
                        /* *                        entre les machines                          * */
                        /* ********************************************************************** */
int verifi_cont_fr()
{
        int i, j, a;

        a = 0;
        for (i = 0; i < nb_mach; i++) {
                for (j = 0; j < nb_mach; j++) {
                        if (i != j) {
                                if ( ( (cont[i][j] == 1) && (poids[i][j] >= 5) && (dist[i][j] > dis[i][j]) ) ||
                                                ( (cont[i][j] == 2) && (poids[i][j] >= 5) && (dist[i][j] < dis[i][j]) ) )

                                        a = 1;
                        }
                        if ( a == 1)     break;
                }
                if ( a == 1)     break;
        }
        return a;
}
                /* ************************************ F I N ************************************** */
```