THÈSE DE DOCTORAT

Présentée à

L'UNIVERSITÉ PAUL VERLAINE-METZ

Pour l'obtention du grade de

Docteur de l'Université Paul Verlaine de Metz

Spécialité : **Automatique**

par

Liya GU

Modèles Déterministe, Stochastique et Multicritère pour l'Equilibrage de Lignes d'Assemblage

Soutenue le 3 mars 2008 devant le jury composé de :

ABDELHAKIM ARTIBA	Professeur à Supméca-Paris (rapporteur)
LIONEL DUPONT	Professeur à l'Ecole des Mines d'Albi Carmaux (rapporteur)
JEAN-MICHEL HENRIOUD	Professeur à l'Université de Franche-Comté (examinateur)
JEAN CHARLES BILLAUT	Professeur à l'Ecole Polytechnique de l'Université de Tours (président de soutenance)
XIAOLAN XIE	Professeur à l'ENSM.SE (directeur de thèse)
SOPHIE HENNEQUIN	Maître de Conférences à l'ENIM (co-encadrant)
ALEXANDRE SAVA	Maître de Conférences à l'ENIM (co-encadrant)

REMERCIEMENTS

Les travaux présentés dans ce mémoire ont été effectués à l'ENI de Metz dans l'équipe des systèmes de production (SDP) du laboratoire de Génie Industriel et Production Mécanique (LGIPM).

J'exprime mon profond remerciement à mes rapporteurs, Monsieur Abdelhakim ARTIBA et Monsieur Lionel DUPONT qui ont porté un regard critique et pertinent sur mon manuscrit.

Je remercie très vivement mes examinateurs, Monsieur Jean-Michel HENRIOUD et Monsieur Jean Charles BILLAUT pour l'honneur qu'ils m'ont fait d'être membres du jury et d'apprécier mon travail.

Je voudrais également témoigner de ma reconnaissance à Monsieur Xiaolan XIE, mon directeur de thèse, Responsable du département Génie Industriel Hospitalier à l'école Nationale Supérieure des Mines de Saint-Etienne, pour m'avoir accueilli dans son équipe et pour toute sa confiance durant cette thèse. Je tiens à lui adresser mes vifs remerciements pour toute sa disponibilité et la façon dont il a partagé ses connaissances qui ont permis de faire progresser cette recherche.

Je remercie également Madame Sophie HENNEQUIN, Maître de Conférences à l'ENI de Metz, et Monsieur Alexandre SAVA, Maître de Conférences à l'ENI de Metz, mes encadrants, pour l'aide très importante qu'ils m'ont apportée tout au long de cette étude, tant sur les idées scientifiques que sur la vérification de mon travail et également sur ma vie quotidienne en France.

Je tiens à remercier l'ENI de Metz, son Directeur Monsieur Pierre PADILLA et le Conseil Régional de Lorraine pour le soutien financier de ma recherche.

Je suis très reconnaissant à toutes les personnes du LGIPM et de l'équipe SDP, pour leurs conseils techniques et scientifiques et le temps qu'ils m'ont accordé. J'exprime tous mes remerciements à Mademoiselle Kun YUAN, Messieurs Jie LI et Iyad MOURANI pour m'engager dans la recherche en algorithmique et en optimisation combinatoire.

Enfin, comment ne pas associer à mon parcours, mes parents, ma famille, mes amis qui m’ont toujours entouré de leur profonde affection et qui m’ont soutenu pendant toutes mes études en France ; comment ne pas remercier Guang YANG qui partage ma vie en France depuis 5 ans.

Table des matières

Liste des tableaux

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Liste des figures

Chapitre 1 :

Chapitre 2 :

Chapitre 3 :

Chapitre 4 :

Liste des notations

c	Temps de cycle
c_best	Plus petit temps de cycle obtenu
$\underline{c}$	Borne inférieure du temps de cycle initial
$\overline{c}$	Un temps de cycle maximum $\overline{c}=UC-1$
LC	Borne inférieure du temps de cycle
LLC_{μ}	Borne inférieure du temps de cycle pour un nœud μ utilisé dans les procédures par séparation et évaluation
UC	Borne supérieure du temps de cycle
t_i	Temps d'opération de l'opération i dans les modèles déterministes ou moyenne du temps d'opération de l'opération i dans les modèles stochastiques
t_{max}	Temps d'opération le plus grand
t_{sum}	Somme des temps d'opération
ST_k	Temps de stations pour la station k
TM_k	Temps mort de la station k
TML	Temps mort de la ligne
TR	Temps de retard
LST_k	Limite inférieure du temps de station pour la station k
m	Nombre de stations
n	Nombre d'opérations
N	Ensemble d'opérations
L_i	Station au plus tard pour l'opération i

E_i	Station au plus tôt pour l'opération *i*
SI_i	Intervalle entier entre E_i et L_i
S_k	Ensemble des opérations pouvant être exécutées sur la station *k*
F_i*	Ensemble des successeurs de l'opération *i*
P_i*	Ensemble des prédécesseurs de l'opération *i*
PW_i,	Poids de la position de l'opération *i* pour l'heuristique proposée
TS_k	Seuil de la station *k*
δ	Moyenne des temps d'opération pour les modèles déterministes
q^i	Charge de la particule électrique chargée *i* dans le mécanisme électromagnétique
F_i	Force totale du point *i* dans le mécanisme électromagnétique
R	Vecteur des clés aléatoires
D_i	*i*ème population de solutions dans la méthode d'estimation de distribution
D^{se}	Individus étalon dans la méthode d'estimation de distribution
$P_i(x)$	Distribution de probabilité décrivant la répartition de l'ensemble des solutions sélectionnées dans l'itération *i* pour la méthode d'estimation de distribution
W_k	Ensemble des opérations assignées à la station *k*
σ^2	Variance du temps d'opération
FL	Fiabilité de la ligne
IP	Probabilité d'incomplétude des opérations
Ω_k	Ensemble des colonnes de la station *k* composées des opérations *i* pour la génération de colonnes
Ω	Ensemble des colonnes pour la génération de colonnes

Introduction Générale

Les lignes d'assemblage, appelées aussi chaînes de montage, sont largement utilisées pour assembler rapidement un grand nombre de produits uniformes dans différents secteurs industriels comme par exemple l'industrie automobile, l'aéronautique, l'industrie pharmaceutique, les télécommunications, etc. Les produits concernés sont divers et variés allant des machines outils industrielles aux produits quotidiens comme les jouets et les produits électroniques. Le problème de l'équilibrage d'une ligne d'assemblage consiste à affecter les opérations d'assemblage aux stations de la ligne de façon à équilibrer les charges entre les stations tout en respectant des contraintes de production. Ce problème se pose lors de la conception préliminaire d'une nouvelle ligne, mais également au moment d'un changement important de la production. En effet, une mauvaise affectation des opérations aux stations peut entraîner un temps mort non justifié et donc des coûts supplémentaires inutiles pour chacune des pièces produites.

Plusieurs modèles de lignes d'assemblage sont déclinés dans l'industrie. Les travaux que nous présentons dans ce mémoire concernent le problème d'équilibrage de lignes d'assemblage en série avec un nombre de stations fixé et des contraintes de précédence pour l'exécution des opérations. Ce type de problèmes est défini comme étant des problèmes d'équilibrage de lignes d'assemblage de type II ou SALBP-2. En fait, le problème SALBP-2 correspond à la mise en production d'un nouveau modèle de produits sur une ligne existante. Un seul type de produit est considéré avec une gamme de fabrication unique. La durée pour réaliser une opération peut être déterministe ou aléatoire. La performance que nous souhaitons optimiser est le temps de cycle. Le temps de cycle est défini comme étant l'intervalle de temps entre la production de deux produits dont la fabrication se succède sur la ligne. Dans le cas de durées des opérations déterministes, ce temps de cycle peut être déterminé par le temps le plus important qu'un produit passe sur une station de la ligne. Dans le cas où la durée des opérations est stochastique (cas des lignes manuelles ou semi-automatisées ou encore lorsque

les pannes des machines sont considérées), le temps de cycle est défini sous la contrainte d'une bonne fiabilité de la ligne. Autrement dit, notre objectif est d'affecter les opérations sur les machines d'une ligne d'assemblage afin de maximiser la productivité de cette ligne.

Le plan de ce mémoire est le suivant. Le premier chapitre vise à établir un état de l'art des travaux existants dans la littérature et traitant le problème de l'équilibrage des lignes d'assemblage. Ainsi, nous donnons d'abord quelques définitions sur les lignes d'assemblage. Ensuite, nous introduisons une classification pour les problèmes d'équilibrage de lignes d'assemblage. Puis, nous présentons les approches existantes qui traitent les problèmes d'équilibrage de lignes d'assemblage de type II (SALBP-2). Enfin, nous situons nos travaux de recherche par rapport aux travaux existants dans la littérature.

Par la suite, dans le deuxième chapitre, nous considérons le problème d'équilibrage des lignes d'assemblage où la durée des opérations est déterministe. Tout d'abord, nous proposons une heuristique spécifique aux problèmes d'équilibrage de lignes d'assemblage de type II. Cette heuristique est composée de deux étapes. La première étape permet de générer une solution initiale qui respecte les contraintes de précédence des opérations et le nombre de stations donné pour la ligne. Ensuite, cette solution est améliorée par un mécanisme de transfert et d'échange. Des résultats numériques permettent de comparer notre heuristique avec ceux d'une heuristique bi-directionnelle. Puis, deux méta-heuristiques sont utilisées pour résoudre ce même problème. Le premier algorithme est basé sur le mécanisme d'attraction - répulsion de particules chargées d'électricité (electromagnetism-like mechanism - EM). Le deuxième algorithme, appelé estimation de distribution (ED) est inspiré des algorithmes génétiques. Dans ce cas, les nouvelles populations de solutions ne sont pas générées par mutation et croisement. Des résultats numériques sont donnés et comparés avec ceux obtenus par la méthode du recuit simulé (SA).

Le troisième chapitre traite des problèmes d'équilibrage de lignes d'assemblage pour lesquels les durées opératoires sont aléatoires. Dans ce cas, il est possible que la durée de traitement d'un produit sur une machine dépasse le temps de cycle. La probabilité que les opérations ne puissent pas être complétées dans les stations (la probabilité d'incomplétude des opérations) est une performance importante de la ligne d'assemblage car elle a une influence considérable sur sa productivité. Deux types de problèmes sont alors considérés : les problèmes avec un seul objectif et des problèmes multi-objectif. Pour les problèmes avec un seul objectif, le

temps de cycle est minimisé sous la contrainte d'une fiabilité donnée. Pour les problèmes multi-objectif, le temps de cycle et la probabilité d'incomplétude des opérations sont optimisés simultanément. L'objectif des problèmes multi-objectif est de déterminer un ensemble de solutions Pareto-optimales. L'approche que nous proposons pour résoudre chacun des problèmes est basée sur la méthode EM. Dans le cas multi-objectif, nous modifions le mécanisme EM de façon à utiliser l'efficience de cette méthode pour notre problème.

Le choix de la solution initiale joue un rôle important dans la vitesse de convergence d'un algorithme d'optimisation. Par conséquent, dans le chapitre 4, nous proposons une méthode de calcul pour la borne inférieure du temps de cycle ce qui nous permettra de définir ensuite une bonne solution initiale pour notre problème. Cette méthode consiste à représenter le problème d'équilibrage sous forme de programmation linéaire et de le résoudre à l'aide de la méthode de génération des colonnes. Cette méthode est efficace pour des problèmes de programmation linéaire de grande taille.

Enfin, des conclusions terminent le contexte de ce mémoire en dressant un bilan des travaux présentés et en ouvrant de nouvelles perspectives de développement.

Chapitre 1.

Equilibrage de lignes d’assemblage

Dans ce premier chapitre nous présentons les problèmes d’équilibrage des lignes d’assemblage. Ils visent à équilibrer au mieux une ligne de production de façon à obtenir un taux de production élevé et par conséquent de réduire les coûts liés à la fabrication des produits. Nous introduisons d'abord les définitions de base de l'équilibrage des lignes. Nous présentons ensuite quelques classifications des modèles existants, ainsi que les approches de modélisation et de résolution proposées dans la littérature. Nous terminons ce chapitre par une synthèse des travaux existants et nous introduisons notre projet de recherche.

1.1 Introduction

Une ligne d'assemblage, appelée aussi une *chaîne de montage*, a été mise en place pour la première fois dans la production industrielle en 1913 par Henry Ford. Cette ligne d'assemblage utilisée pour la fabrication des véhicules Ford dans une usine du Michigan a permis d'augmenter par dix la quantité produite ainsi que la rentabilité, et de réduire largement les coûts de fabrication. Depuis son apparition, les lignes d'assemblage ont été largement employées pour assembler rapidement un grand nombre de produits uniformes, comme dans l'industrie automobile, l'aéronautique, l'industrie pharmaceutique, les télécommunications, etc. pour une grande variété de produits allant des machines outils industrielles aux produits quotidiens comme les jouets et les produits électroniques.

Une ligne d'assemblage est un ensemble de stations de travail (postes de travail) interconnectées entre elles à l'aide d'un moyen de transfert mécanisé (comme un convoyeur, un tapis roulant, un bol vibrant etc.). Le produit est transféré sur les stations pour subir les diverses opérations d'assemblage nécessaires. A la fin de la ligne, le produit assemblé sort de la ligne. Un exemple d'une ligne d'assemblage est montré dans la Fig. 1. 1. Dans l'exemple, les pièces des produits ***P*** passent par les quatre stations sur la ligne pour exécuter les opérations.

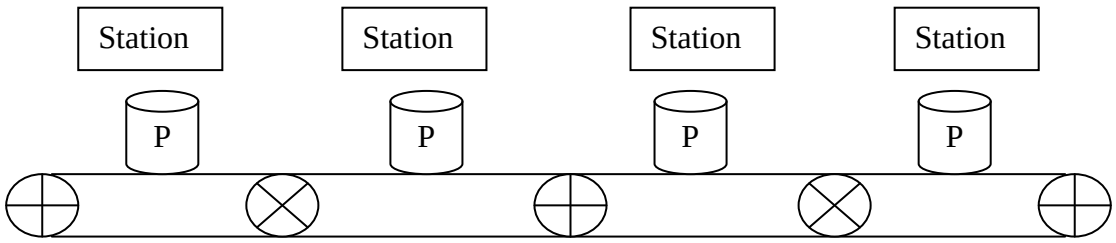


Fig. 1. 1 Une ligne d'assemblage

Dans ce qui suit, nous donnons quelques définitions de lignes d'assemblage qui seront utilisées dans la suite du mémoire et qui sont présentées dans [ABD 95].

Opération (ou tâche) : Une opération est la plus petite unité de travail. Elle est pratiquement indivisible. Un travail à réaliser sur un produit peut être divisé en plusieurs opérations indivisibles. A chaque opération sont associés au moins deux types de paramètres qui correspondent à sa durée et à des contraintes. La durée nécessaire pour compléter une

opération est dite temps d'opération ou temps opératoire. Nous présentons les contraintes ultérieurement dans les sections 1.1 et 1.2.2.

Station : Une station est un élément de la ligne où les opérations sont réalisées sur un produit. La totalité du travail à réaliser sur une station est appelée la *charge* de cette station. La durée d'exécution de la charge d'une station s'appelle le *temps de station*.

Temps de cycle : temps compris entre la production de deux unités sur une ligne d'assemblage. Généralement le temps de cycle est donné par la station qui a le temps de station le plus grand (station goulet). Il correspond à la productivité de la ligne d'assemblage. Remarquons qu'on le confond souvent avec le temps de réponse ou temps du système qui est le temps de présence d'un produit dans la ligne.

Temps mort : Le temps mort d'une station correspond à la différence entre son temps de station et le temps de cycle de la ligne. Le temps mort représente un gaspillage des ressources et par conséquent augmente le coût de production.

Contraintes de précédence : Les contraintes de précédence présentent l'ordre partiel de réalisation des opérations, et ce en raison des restrictions technologiques (ex. dans une ligne d'assemblage des ordinateurs, la carte mère doit être fixée dans le boîtier d'ordinateur avant d'installer les cartes sur la carte mère). Les contraintes de précédence sont illustrées par un graphe appelé *graphe de précédence* [PRE 64]. Un exemple de graphe de précédence est donné dans la Fig. 1. 2.

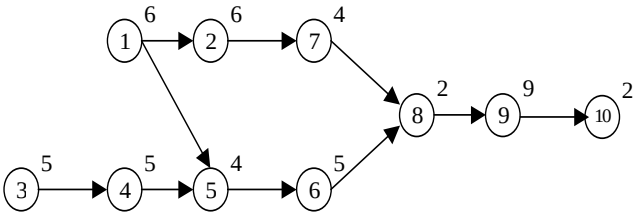


Fig. 1. 2 Graphe de précédences

Dans ce graphe, chaque opération est représentée par un nœud. Ces opérations sont numérotées topologiquement selon leur position dans le graphe. Le nombre associé à une opération est toujours plus grand que celui associé à une opération précédente. Les temps

d'opération sont indiqués par un poids associé au nœud correspondant. Un arc (*i*, *h*) indique la relation de précédence entre l'opération *i* et l'opération *h*. Si un arc (*i*, *h*) existe, l'opération *i* doit être accomplie avant que l'opération *h* puisse commencer. L'opération *i* (resp. opération *h*) est alors un prédécesseur direct (resp. un successeur direct) de l'opération *h* (resp. *i*). Cette relation de précédence définit aussi des prédécesseurs indirects qui sont des prédécesseurs des prédécesseurs et des successeurs indirects qui sont des successeurs des successeurs.

1.2 Problèmes d'équilibrage de lignes d'assemblage

L'équilibrage d'une ligne d'assemblage est un problème d'optimisation combinatoire qui a été formulé pour la première fois par M. E. Salveson [SAL 1955]. Il s'agit d'affecter les opérations aux stations tout en respectant les différentes contraintes de façon à optimiser un critère d'efficacité donné. Ce problème se pose lors de la conception préliminaire d'une nouvelle ligne, mais également au moment d'un changement important de la production. Une mauvaise répartition des opérations aux stations peut entraîner un temps mort non justifié et des coûts supplémentaires inutiles pour chacune des pièces produites.

Une solution de ce problème est ce qu'on appelle une *ligne équilibrée* qui cherche à minimiser les temps morts des stations afin d'optimiser le taux d'utilisation des ressources. Pour le système de production présenté dans la Fig. 1. 2, si nous définissons que le temps de cycle est égal à 11 et le nombre de stations dans la ligne est 5, une ligne équilibrée (une solution du problème) est donnée dans la Fig. 1. 3.

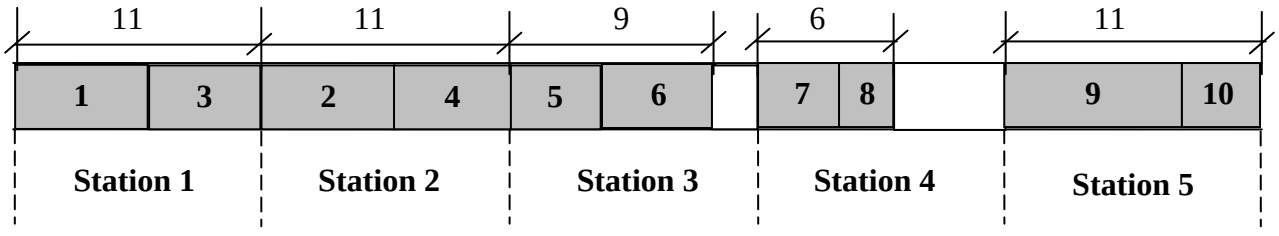


Fig. 1. 3 Une ligne équilibrée

Dans la Fig. 1. 3, les opérations sont représentées par des rectangles gris. Dans cette ligne, les opérations 1 et 3 sont affectées à la station 1 et les opérations 2 et 4 sont affectées à la station 2 etc. Les temps de stations sont présentés par les chiffres au-dessus des rectangles. Alors qu'aucun temps mort ne se produit dans les stations 1, 2 et 5, des temps mort apparaissent dans les stations 3 et 4, temps mort qui sont respectivement égaux à 2 et 5. Par conséquent, le temps mort total de la ligne est égal à 7.

1.2.1 Classification et modélisation des lignes d'assemblage

Suite à l'importance croissante de ce problème dans la production depuis 60 ans, beaucoup de travaux de recherche ont été menés. Ils sont classés précisément selon différents critères comme le montre la Fig. 1. 4 ([ARM 99], [REK 02], [FIN 04], [CHR 05], [DOL 06], [NIL 06a] et [NIL 06b]).

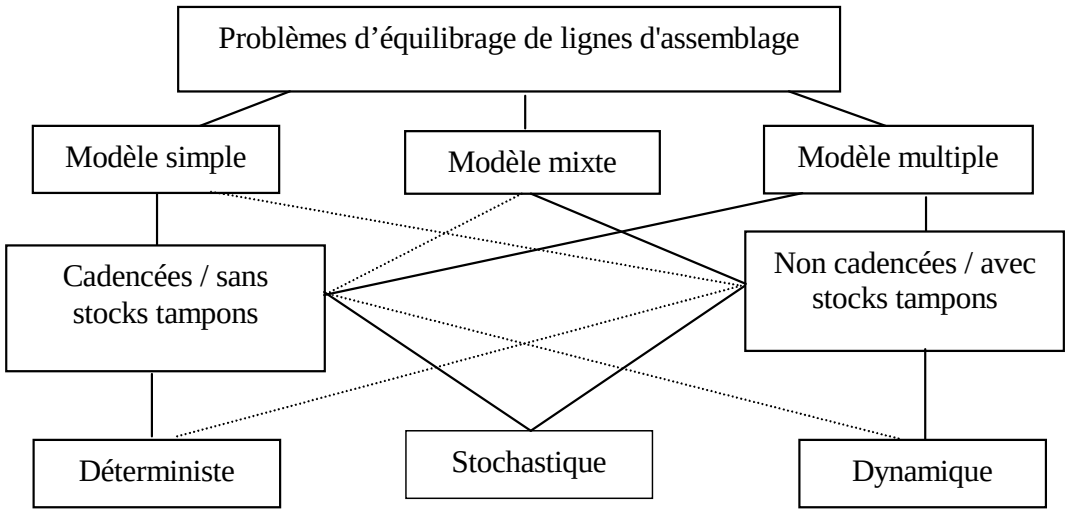


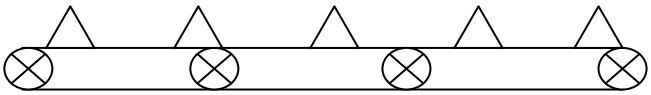
Fig. 1. 4 Classification des problèmes d'équilibrage de lignes d'assemblage [ARM 99]

Une première classification peut se faire selon le nombre de types de produits différents fabriqués sur la ligne. Dans ce sens trois grands modèles peuvent être définis (Fig. 1. 5) :

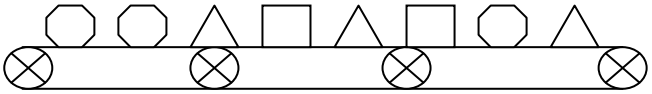
- le modèle simple pour lequel un seul type de produit est assemblé par ligne;
- le modèle mixte où plusieurs types de produits sont fabriqués sur la ligne. Les produits de différents types sont lancés sur la ligne en respectant des ratios donnés, mais sans regroupement obligatoire par type, c'est-à-dire qu'à chaque instant un

mélange de produits de différents types peut être présent sur la ligne, voir ([[JIN 03], [JOS 02a] et [NOO 05]);

- le modèle multiple pour lequel plusieurs types de produits sont fabriqués sur la ligne. Les différents types de produits sont lancés sur la ligne par lots. Pour le modèle multiple, deux problèmes se posent en même temps, le problème d'équilibrage de la ligne pour chaque lot et le problème de séquençage des lots, voir ([RAF 02] et [GRA 88]).



Ligne pour un modèle simple



Ligne pour un modèle mixte

Ligne pour un modèle multiple

avec△ □ ○ : trois sortes de produits différents
: mis en place

Fig. 1. 5 Trois modèles de ligne

Les lignes d'assemblage peuvent être classées également selon le cadencement ou non :

- *ligne cadencée* : toutes les stations dans la ligne ont des temps limités identiques (temps de cycle) pour exécuter les opérations assignées. Dans une ligne cadencée, il n'y a pas de stock tampon. La plupart des travaux sont proposés pour ce type de lignes, voir ([SUB 99] et [JUN 97]).
- *ligne non cadencée* : les stations sont séparées par des stocks tampons où les pièces assemblées sont déposées lorsque la station suivante est occupée par la fabrication d'une pièce. Si le stock tampon est plein, ce qui correspond au fait que le temps de station de la station en amont du stock est plus petit que celui de la station en aval du stock, alors la ligne est bloquée. Dans le cas contraire, la ligne peut être affamée.

La variabilité des temps d'opération peut également être considérée pour classer les lignes d'assemblage. Dans ce cas trois grands types de modèles peuvent être définis selon l'évolution du système en fonction du temps :

- *modèle déterministe* : la plupart des modèles d'équilibrage de lignes d'assemblage sont des modèles déterministes avec des temps d'opération constants et connus. Ce modèle est une simplification du modèle stochastique quand la variation du temps est suffisamment petite.
- *modèle stochastique* : dans une ligne manuelle, la performance de la ligne de production est forcément dépendante du travail des opérateurs. Le temps d'opération peuvent être très différents en raison des compétences et des motivations différentes des opérateurs. Même si le temps d'opération est presque constant dans une ligne automatisée, une variation des taux de production des stations peut se produire à cause des pannes des machines. Dans la littérature, les temps des opérations d'une ligne manuelle sont souvent donnés selon une loi normale ([SUB 94] et [SUB 99]), tandis que pour la ligne automatisée, d'autres lois de distributions comme la loi exponentielle sont utilisées ([RAM 70], [KAO 79] et [SNI 81]).
- *modèle dynamique* : les temps d'opération peuvent être réduits en raison d'effets d'apprentissage ou d'améliorations successives du procédé de production. Cela se produit essentiellement quand une nouvelle ligne est installée, car les opérateurs prennent plus de temps pour exécuter des opérations jusqu'à atteindre un temps relativement constant de réalisation des opérations [GRA 95].

Une classification importante est également définie selon la *disposition du système de production.* Ainsi on peut définir :

- *ligne en série* : c'est une ligne classique sur laquelle la plupart des travaux sont proposés. Dans ce type de lignes, les stations sont organisées selon une ligne droite le long d'un convoyeur.
- *ligne en forme de U* : les deux extrémités de cette ligne forment un U étroit (Fig. 1. 6). Les stations peuvent travailler selon deux segments face à face dans la ligne simultanément. Rivalisant avec la ligne en série, une ligne en forme de U permet de réduire la longueur de la ligne et le nombre de stations dans la ligne ([AJE 98], [TIM 98]). [GER 98] prouve que la ligne en forme de *U* peut augmenter le taux de production. Elle peut parfois diminuer le nombre de stations ou construire une ligne

plus équilibrée que la ligne en série. Dans cette configuration certains opérateurs doivent bouger dans la ligne pour exécuter des opérations différentes.

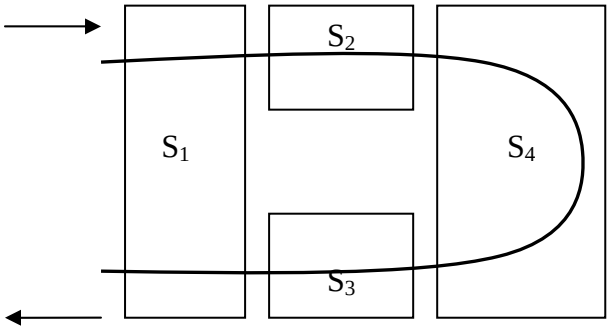


Fig. 1. 6 La ligne en forme-U

- *Ligne parallèle (station parallèle)* : plusieurs stations mises en parallèle sont installées dans une ligne pour fabriquer un ou plusieurs produits. Cette configuration augmente la flexibilité et diminue l'impact des pannes des machines du système. Autrement dit, une ligne parallèle permet de mieux satisfaire les changements de la demande, car le nombre de stations utilisées peut changer en fonction de l'évolution de la demande ([MCM 06] et [GOK 06]).
- *Ligne à deux cotés* : les opérateurs peuvent être positionnés sur les deux cotés d'une ligne et plusieurs opérateurs peuvent travailler simultanément sur le même produit en exécutant des opérations différentes. Ce type de ligne est utilisé quand le produit à assembler est très grand et n'est pas facile à déplacer, c'est à dire comme pour une ligne d'assemblage d'automobiles, les quatre pneus peuvent être installés en même temps sur les deux cotés de la ligne ([KIM 00]).

1.2.2 Contraintes supplémentaires

D'autres contraintes en dehors des contraintes de précédences peuvent être définies. Ces contraintes sont présentées dans ([ARM 99], [BOY 07a], [CHR 05]).

Contraintes liées aux opérations : dans certaines situations, des limites sur la position de deux (ou plusieurs) opérations existent, c'est à dire que par exemple deux opérations doivent être exécutées sur une même station ([DEC 89]) ou qu'elles ne doivent pas être réalisées sur une même station ([BAU 00]). Le premier cas (exécution d'au moins deux opérations dans

une même station) correspond aux opérations qui nécessitent certaines particularités au niveau de la production, comme par exemple une température identique, etc. Scholl et al. [ARM 03] proposent une approche basée sur le concept de groupes d'opérations. Ces groupes d'opérations correspondent aux opérations qui doivent impérativement être réalisées sur une même station.

Contraintes liées aux stations : si des machines et des équipements spéciaux sont obligatoires sur une station pour exécuter une certaine opération qui ne peut être déplacée sur une autre station sans ajouter des coûts prohibitifs, alors dans ce cas là, des contraintes entre l'opération et la station existent (en anglais : must-do-task, fixed facility restriction, [KIL 61b]. Bartholdi met en application une version modifiée d'un heuristique basé sur les règles de priorité [ARM 03] qui permet de résoudre des problèmes fixant des opérations à des stations spécifiques.

Contraintes liées à la position : dans le cas d'une pièce trop grande et trop lourde, un espace à coté de la pièce concernée peut être nécessaire pour exécuter des opérations [JOA 07]. Comme il n'est pas possible ou économiquement inintéressant de déplacer cette pièce, des opérations sur cette pièce doivent être groupées dans une même partie de la ligne.

Contraintes liées aux opérateurs : les opérations ont des complexités techniques différentes. La qualification d'un opérateur dépend de l'opération la plus complexe à réaliser sur la station où se trouve l'opérateur. De plus, les opérateurs sont payés selon l'opération la plus complexe qu'ils exécutent. Donc, il peut être intéressant d'exécuter des opérations avec une complexité similaire par un opérateur [JOH 83].

Une fois le modèle de la ligne d'assemblage clairement défini, les critères qui vont permettre de concevoir ou reconcevoir au mieux la ligne doivent être définis. Nous présentons dans ce qui suit les différents critères d'optimisation qui sont proposés dans la littérature.

1.2.3 Critères d'optimisation

Plusieurs critères sont proposés pour l'équilibrage de lignes d'assemblage. Les critères peuvent correspondre à la minimisation des temps de production [LAP 06], minimisation des

coûts [BEL 06] ou maximisation du profit [GRA 95]. L'objectif peut également être la sélection des équipements utilisés sur la ligne [STE 87].

Capacité (temps) : dans la littérature, la plupart des problèmes d'équilibrage de lignes d'assemblage sont liés à des objectifs de capacité (temps) pour évaluer les solutions obtenues.

Cas des problèmes déterministes :

1. Maximiser l'efficience de la ligne (pourcentage de temps de production dans la ligne)
2. Minimiser le temps mort total par pièce
3. Minimiser le pourcentage de temps mort
4. Minimiser le temps de cycle
5. Equilibrer les taux d'utilisation des différentes stations
6. Minimiser le temps d'écoulement (intervalle de temps entre le lancement d'un produit au début de la ligne et la sortie du produit fini de cette ligne)

Cas des problèmes stochastiques :

7. Maximiser la fiabilité de la ligne (qui correspond au produit de la fiabilité de chaque station)
8. Maximiser la fiabilité de la station la moins fiable
9. Minimiser la probabilité d'arrêt de la ligne

Coût : le coût de production est un indicateur important pour les entreprises. En effet, quand une ligne est installée, un investissement en équipements peut être estimé. Les objectifs liés aux coûts de production étudiés dans la littérature sont les suivants :

1. Minimiser les coûts d'équipements (minimiser le nombre de stations)
2. Minimiser les coûts salariaux
3. Minimiser les coûts du matériel
4. Minimiser les coûts engendrés par le temps mort
5. Minimiser les coûts engendrés par les produits mal achevés
6. Minimiser les coûts de stockage

Profit : les objectifs de profit sont plutôt évolués par rapport aux objectifs de coût. Les objectifs de profit considèrent le taux de production (temps de cycle) et le coût de production comme des variables de décision, tandis que les objectifs de coût considèrent le volume de la production et le coût de production donnés.

L’objectif est généralement de maximiser le profit par unité ou par période de temps.

Sélection d’équipements : les équipements sont considérés comme étant donnés dans les modèles que nous avons présentés ci-dessus. Cependant, la sélection des équipements est liée aux conditions induites par les opérations assignées à une station. En outre, la manière de fabriquer un produit peut dépendre de ces équipements. Cet objectif apparaît plutôt dans le problème de la conception de lignes d’assemblage ([JOS 00] et [GRA 88]).

Les objectifs sont définis très clairement pour satisfaire les besoins réels des entreprises. Dans la littérature, la plupart des travaux définissent un seul objectif à optimiser, tandis que les autres objectifs sont considérés comme des contraintes. Des travaux plus complexes permettent d’optimiser plusieurs objectifs ensembles, dans ce cas on parle de problèmes d’équilibrage multi-objectif ([CEL 99], [GRA 88], [MCM 88] et [MCM06]).

Par la suite, nous présentons la résolution de problèmes d’équilibrage de lignes d’assemblage simple de type II (SALBP-2) qui considère du temps de cycle de la ligne d’assemblage cadencées sans stock tampons.

1.3 Résolution de problèmes SALBP

Les *principales hypothèses* de ce type de problème sont énoncées par [BAY 86] et [BRA 02] :

1. les temps d’opération sont déterministes,
2. une opération est définie comme étant la plus petite tâche possible et ne peut par conséquent pas être divisée sur deux stations,
3. il existe des contraintes de précédence,
4. toutes les opérations doivent être réalisées,
5. n’importe quelle station peut faire n’importe quelle opération,
6. le temps opératoire d'une opération ne dépend pas de la station qui l'exécute,
7. n’importe quelle opération peut être affectée à n’importe quelle station,
8. les stations sont disposées en série,
9. la ligne est conçue pour un seul type de produit et elle a un seul mode de fonctionnement.

Enfin, seulement une des deux hypothèses suivantes est utilisées à la fois :

10a le temps de cycle *CT* est donné et fixe,

10b le nombre de stations *M* est donné et fixe.

Plusieurs versions de ce problème sont proposées dans la littérature selon les objectifs à optimiser. Nous les précisons dans le Tableau 1. 1. SALBP-F est un problème qui vérifie si une ligne d'équilibrage faisable existe ou non pour une combinaison donnée du temps de cycle et du nombre de stations de la ligne. SALBP-1 et SALBP-2 sont deux classiques duaux. SALBP-1 cherche à minimiser le nombre de stations tout en respectant un temps de cycle souhaité, alors que SALBP-2 permet de minimiser le temps de cycle avec un nombre de stations fixé. SALBP-E cherche à maximiser l'efficacité de la ligne, c'est à dire minimiser le temps de cycle et le nombre de stations simultanément en considérant leur corrélation [KIL 61a].

Temps de cycle (*c*) / Nombre de stations (*m*)	Connu	A minimiser
Connu	SALBP-F	SALBP-2
A minimiser	SALBP-1	SALBP-E

Tableau 1. 1 Version de SALBP

Pour la suite, nous présentons essentiellement des formulations du problème SALBP-2 et des publications qui traitent le problème SALBP-2 que nous avons considéré dans nos travaux. Ce problème est utilisé pour regrouper les opérations sur les stations quand un nouveau type de produit est fabriqué sur la ligne.

1.3.1 Formulations mathématiques d'un problème SALBP-2

Plusieurs formulations sont proposées dans la littérature ([BAY 86], [ARM 99]). Nous allons détailler différents modèles mathématiques pour SALBP-2. Dans ces modèles, *c* désigne le temps de cycle, *m* le nombre de stations, *n* le nombre d'opérations et t_i le temps d'opération pour l'opération *i*.

Formulation 1 : Baybars [BAY 86] propose une formulation en programmation linéaire en nombres entiers :

$$x_{ik}=\left\{\begin{array}{ll} 1 & \text{si l'opération } i \text{ est affectée ? la station } k \\ 0 & \text{sinon} \end{array}\right\} \qquad 1.\ 1$$

pour $i = 1,\ldots, n$ et $k \in SI_i$

où SI_i est l'intervalle entier entre E_i et L_i, i.e. = { E_i , E_i +1, ..., L_i,},

L_i est la station au plus tard pour l'opération *i*, c'est-à-dire la dernière station où elle peut être exécutée en respectant les contraintes de précédences,

E_i est la station au plus tôt pour l'opération *i*, c'est-à-dire la première station où elle peut être exécutée en respectant les contraintes de précédences.

La fonction objective est la minimisation du temps de cycle *c* et le problème s'écrit donc :

$$\text{Min } c \qquad 1.\ 2(a)$$

sous les contraintes :

$$\sum_{k\in SI_j} x_{ik}=1 \quad \text{pour } i = 1,\ldots,n \qquad 1.\ 3$$

$$c \geq \sum_{i\in S_k} t_i * x_{ik} \quad \text{pour} \quad k=1,\ldots, m \qquad 1.\ 4$$

$$\sum_{k\in SI_h} k*x_{hk} \leq \sum_{k\in SI_i} k*x_{ik} \quad \text{pour } h \in P_i* \text{ et } L_h \geq E_i \qquad 1.\ 5$$

$$x_{ik} \in \{0,1\} \quad \text{pour } j =1,\ldots,n \qquad 1.\ 6$$

où S_i est l'ensemble des opérations pouvant être exécutées sur la station *i* et P_i^* est l'ensemble des prédécesseurs de l'opération *i*.

La contrainte (1. 3) correspond aux contraintes d'occurrence qui assurent que chaque opération n'est affectée qu'une seule fois. La contrainte (1. 4) présente la définition du temps de cycle qui doit être supérieur ou égal à tous les temps de station de la ligne. La contrainte (1. 5) impose les relations de précédence entre opérations. Si l'opération *h* est à réaliser avant l'opération *i*, l'indice de station où elle est exécutée ne doit pas être supérieur à celui de

l'opération *i*. La contrainte (1. 6) est une contrainte d'intégrité des variables de décision. Dans cette formulation, l'objectif (1.2 a) peut être décrit par :

$$\text{Min Max}\left\{ \sum_{i \in S_k(c)} t_i * x_{ik} \,\middle|\, k = 1,...,m \right\} \qquad \text{1.2 (b)}$$

Dans ce cas là, la formulation ne représente plus un programme linéaire en nombre entiers. Pour que le problème soit plus facile à résoudre, la formule 1.2(b) est utilisée.

Formulation 2 : Scholl [ARM 99] présente une nouvelle formulation similaire à la première. Elle utilise des variables y_{ik} qui sont reliées aux x_{ik} par $x_{ik} = y_{ik} - y_{i,k-1}$, avec $y_{i,Ei-1}=0 \; \forall i \in \{1,\ldots, n\}$.

$$y_{ik} = \left\{ \begin{array}{ll} 1 & \text{si l'opération } i \text{ est affectée à la station } k \text{ ou des stations précédentes} \\ 0 & \text{si non} \end{array} \right\} \qquad \text{1. 7}$$

Chaque vecteur ($y_{i,Ei}$,…, y_{ik},…, $y_{i,Li}$) contient un bloc (ensemble) de 0 suivi par un bloc de 1. Le deuxième bloc se produit pour la station *k* où l'opération *i* est exécutée. Par exemple, un vecteur (0,0,0,1,1) implique qu'une opération est assignée à la station 4, à condition que cette opération soit potentiellement assignable aux stations de 1 à 5. La fonction objective est (1.2 a) et les contraintes sont présentées par :

$$y_{ik} - y_{i,k-1} \geq 0 \qquad \text{pour } i = 1,\ldots, n \text{ et } k \in SI_i \qquad \text{1. 8}$$

$$\sum_{i \in S_k} t_i * (y_{ik} - y_{i,k-1}) \leq c \qquad \text{pour } k = i,\ldots, m \qquad \text{1. 9}$$

$$y_{hk} - y_{i,k} \geq 0 \qquad \text{pour } h \in P_i^* \text{et } L_h \geq E_i \qquad \text{1. 10}$$

où P_i^* est l'ensemble des prédécesseurs (directs ou indirects) de l'opération *i*.

$$y_{i,E_i-1}=0, \quad y_{i,L_i}=1 \qquad \text{pour } i = 1, \ldots, n \qquad \text{1. 11}$$

$$y_{i,k} \in \{0,1\} \qquad \text{pour } i = 1,\ldots, n \text{ et } k \in SI_i \qquad \text{1. 12}$$

L’ensemble des contraintes (1. 8) et (1. 12) assure que chaque opération est assignée dans une seule station. La contrainte (1. 9) permet de respecter le temps de cycle de la ligne. La contrainte (1. 10) représente les contraintes de précédence. La contrainte (1. 11) garantit que chaque opération est assignée dans une station comprise entre la station au plus tôt et la station au plus tard. Evidement, dans la formulation 2, les contraintes sont plus nombreuses que dans la formulation 1. L’avantage de la formulation 2 réside dans le fait que les coefficients des contraintes, excepté celui de la contrainte (1. 9), sont égaux à 1, -1 ou 0.

Formulation 3 : Une autre formulation existe, qui est proposée également par Scholl [ARM 99]. C’est une formulation de programmation linéaire en variables mixtes (MIP). Elle combine les idées de Bowman [BOW 60] avec celles de Talbot et Patterson [TAL 84]. Dans le modèle de Scholl, la variable entière z_i représente le numéro de la station à laquelle l'opération i est affectée et la variable réelle y_i correspond à la date de début de l'opération i dans chaque cycle (l'intervalle de temps entre le lancement d'une pièce sur la ligne et l'instant où l'opération i commence). En outre, une variable binaire est utilisée :

$$w_{ih} = \begin{cases} 1 & \text{si l'opération } i \text{ est effectuée avant l'opération } h \\ 0 & \text{sinon} \end{cases} \qquad 1.\ 13$$

pour $i < h$ et $i \notin A_h^*$ et $SI_i \cap SI_h = \varnothing$

La fonction objectif est spécifiée sous la forme (1.2 a) et les contraintes sont données par :

$$y_i \geq c\ (z_i - 1), \qquad 1.\ 14$$

$$y_i + t_i \leq c\ z_i, \qquad 1.\ 15$$

$$(1 - w_{ih})\ T_t + y_h \geq y_i + t_i, \text{ pour } i < h,\ i \notin A_h^* \text{ et } SI_i \cap SI_h = \varnothing, \qquad 1.\ 16$$

$$w_{ih}\ T_t + y_i \geq y_h + t_h, \text{ pour } i < h,\ i \notin A_h^* \text{ et } SI_i \cap SI_h = \varnothing, \qquad 1.\ 17$$

$$y_h + t_h \leq y_i, \text{ pour } i \notin A_h^* \text{ et } L_i \geq E_h, \qquad 1.\ 18$$

$$E_i \leq z_i, \text{ et } z_i \leq L_i, \qquad 1.\ 19$$

$$z_i \geq 0 \text{ entière, et } y_i \geq 0, \text{ réelle}, \qquad 1.\ 20$$

$w_{ih} \in \{0, 1\}$, pour $i < h$, $i \notin A_h^*$ et $SI_i \cap SI_h = \varnothing$, 1. 21

où $T_t = m^* c$,

et $i, h \in N$, ici N est l'ensemble d'opérations

Les contraintes (1. 14) et (1. 15) assurent qu'une opération est exécutée sur une et une seule station. Les contraintes (1. **16**) et (1. **17**) précisent que si deux opérations n'ont pas de relation de précédence, alors si l'une d'elle est commencée avant l'autre, elle est terminée avant l'autre. La contrainte (1. **18**) assure que les relations de précédence sont respectées. La contrainte (1. **19**) permet de restreindre les intervalles d'assignation des opérations aux stations et les contraintes (1. **20**) et (1. **21**) définissent les variables de décision. La formulation 3 utilise plus de variables et de contraintes que les autres formulations mais elle est intéressante quand les contraintes de précédence sont fortes. En effet, dans ce cas, le nombre de variables et de contraintes dépend du nombre des opérations qui ne sont pas liées par des contraintes de précédence assemblées par paire.

Parmi les trois formulations mathématiques, la formulation 1 utilise le plus petit nombre de contraintes et la formulation 3 utilise le plus grand nombre de contraintes. L'avantage de la formulation 3 apparaît quand le nombre de contraintes de précédence est grand. Plus le nombre de contraintes de précédence est grand, moins nombre de variables de décision est grand utilisées. Dans la formulation 2, les coefficients des contraintes, excepté celui de la contrainte du temps de cycle, sont égaux à 1, -1 ou 0.

Ces formulations sont utilisées par certaines méthodes exactes, comme la méthode du Simplexe, etc..

1.3.2 Méthodes exactes

Les Procédures par Séparation et Evaluation (PSE) sont beaucoup plus utilisées pour résoudre le SALBP de type I ([BAY 86], [HOF 92] et [JOS 02b]). Dans ce cas, FABLE ([JOH 88]) et EUREKA ([HOF 92]) sont deux approches efficaces. Ces procédures peuvent être utilisées pour le SALBP de type II à l'aide de la procédure basée sur des itérations (voir les paragraphes suivants). Dans chaque itération, un nombre de stations est déterminé par ces procédures avec un temps de cycle d'essai compris entre une borne inférieure LC et une borne supérieure UC. Des méthodes pour le calcul des LC et UC seront présentées dans la section

1.3.3.3. Le temps de cycle minimum correspondant au nombre de stations *m* est obtenu après quelques itérations.

Klein et Scholl comparent une approche SALOM-2 (**S**imple **A**ssembly **L**ine Balancing **O**ptimization **M**ethod de type 2) avec TBB2 (Task oriented Branch and Bound procedure) [ROB 96]. Ces deux méthode sont basées sur des PSE. Klein et Scholl ont montré expérimentalement que SALOM-2 est plus efficace que TBB-2. Ceci est dû au fait que le processus SALOM-2 garde toujours la borne inférieure locale au lieu de la borne supérieure pour TBB-2. D'autre part, TBB-2 assigne une opération dans chaque branchement donc il a besoin de plus de branchements comparativement à SALOM-2 qui construit une station dans chaque branchement.

TBB-2 : La méthode TBB-2 est une procédure **orientée opération**, c'est à dire que la construction d'un nouveau nœud (branchement) se fait par ajout d'une opération candidate (qui n'a plus de prédécesseurs non assignés) à une station. Un nœud utilisé par TBB-2 présente une étape d'assignation d'une opération dans une station.
Cette méthode commence par déterminer une borne inférieure *LC* et une borne supérieure *UC* du temps de cycle de la ligne. Puis un temps de cycle maximum est obtenu par $\bar{c}=UC-1$. Pour chaque opération *i*, les stations au plus tôt $E_i(\bar{c})$ et au plus tard $L_i(\bar{c})$ basées sur $\bar{c}$ sont déterminées par les équations (1. 28) et (1. 29). Si $E_j(\bar{c}) = L_i(\bar{c})$, l'opération *i* est alors assignée dans la station $E_i(\bar{c})$. Dans chaque nœud, une opération *i* est choisie d'après des règles prioritaires définies (ex : *la règle du temps d'opération maximum, la règle de contraintes de précédence, etc.*). Ensuite, une station suivante $k \in SI_i(\bar{c})$ est choisie selon l'ordre croissant et dans une seule direction ou dans deux directions. L'opération *i* est alors assignée à la station *k*. Des nouvelles stations au plus tôt et au plus tard des opérations restantes sont actualisées. Une fois qu'une opération est assignée à une station, un nouveau nœud de TBB-2 est ajouté. Pour chaque nœud μ de TBB-2, une borne inférieure locale LLC_μ est calculée par :

$$LLC_\mu = \mathrm{Max}\left\{LLC_\sigma, \mathrm{Max}\left\{ST_k \mid k = 1, \cdots, m\right\}\right\} \qquad 1.\ 22$$

LLC_σ est la borne inférieure locale du nœud précédent.
Si $LLC \geq UC$ la branche est abandonnée. Quand toutes les opérations sont assignées une

branche est construite et une borne supérieure locale est donnée par :

$$UCL = \text{Min}\left\{UC, \text{Max}\left\{ST_k \,|k = 1,\ldots,m\right\}\right\} \quad \text{si } UCL<UC \qquad 1.\ 23$$

SALOME-2 : La méthode Salome-2 est proposée par [ROB 96]). Dans SALOME-2, l'énumération est **orientée station**, c'est-à-dire qu'aux branches, nous cherchons des combinaisons d'opérations réalisables formant une station. Nous en choisissons une et affectons toutes les opérations appartenant à cette combinaison d'un coup à la nouvelle station pour former un nouveau nœud d'arbre de l'énumération. A chaque nœud, une borne inférieure locale du temps de cycle est définie. LLC_k est utilisée comme borne inférieure locale du nœud au niveau k qui représente une solution partielle avec des stations 1,…,k chargées. Ensuite, le temps mort d'une station k est donné par $TM_k = LLC_k - ST_k$. Le temps mort de la ligne est la somme des temps mort de toutes les stations et il ne doit pas dépasser le temps de retard noté par: $TR = LLC * m - t_{sum.}$. La différence entre le temps mort de la ligne et le temps de retard est notée par RI :

$$RI_k = TR - \sum_{h=1}^{k} TM_h \quad \text{pour k=0,…,}m\text{-1} \qquad 1.\ 24$$

Un temps de cycle peut être réalisé seulement si les temps de station pour les stations k+1,…,m sont plus grands qu'une limite inférieure LST_k et plus petits que LLC :

$$LST_k = \text{Max}\left\{0, LLC_k - RI_k\right\} \quad \text{pour k=0,…,}m\text{-1} \qquad 1.\ 25$$

La méthode commence par déterminer les bornes supérieure (UC) et inférieure (LC) globales de la ligne. Elles sont calculées par les méthodes présentées dans la partie précédente. La borne inférieure locale du nœud 0 (LLC_0) est égale à la borne globale LC..

Une branche est supprimée quand $LLC_k > UC$. Pour limiter le nombre d'énumérations, elle emploie des règles de réductions (ex. la règle de charge maximale : l'opération qui minimise le temps mort d'une station est assignée dans cette station en priorité).

1.3.3 Méthodes heuristiques pour SALBP de type II

Les heuristiques trouvent leur place dans les algorithmes qui nécessitent l'exploration d'un grand nombre de cas, car elles permettent de réduire leur complexité moyenne en examinant d'abord les cas qui ont le plus de chances de donner la solution. Il est aussi parfois possible de prouver que le résultat fourni par l'heuristique ne s'éloigne pas trop de la solution optimale, on parle alors de garantie de performance.

1.3.3.1 Bornes inférieure et supérieure

Une borne inférieure du temps de cycle (en anglais, Lower Bound of Cycle Time – *LC*) nous permet de définir le temps de cycle minimum possible d'une ligne. Nous ne sommes jamais sûrs que nous puissions obtenir une solution faisable avec *LC*, mais nous savons qu'il n'est pas possible d'obtenir moins. Par ailleurs, une solution avec le temps de cycle égal à *LC* est une solution optimale.

Dans la littérature, plusieurs formulations pour les bornes inférieures et supérieures sont présentées pour le problème SALBP-2 ([MCN 59], [TAL 84], [HAC 89], et [ARM 99]). Ces bornes sont souvent utilisées dans les procédures par séparation et évaluation (PSE). Elles travaillent aussi avec quelques heuristiques qui réalisent répétitivement des heuristiques pour le problème de type I avec des temps de cycle différents. Dans certaines méthodes méta-heuristiques, les bornes de temps de cycle peuvent être utilisées pour limiter le nombre de solutions évaluées.

Borne inférieure

LC1 : La borne inférieure la plus simple et la plus fréquente est la suivante [MCN 59] :

$$LC1 := \mathrm{Max}\left\{t_{\max}, \left\lceil t_{sum} / m \right\rceil\right\} \qquad 1.\ 26$$

où $\lceil x \rceil$ est le plus petit entier supérieur ou égal à x, t_{max} est le temps d'opération le plus grand et t_{sum} est la somme des temps d'opération.

LC2 : Elle ordonne les opérations dans l'ordre décroissant des temps d'opération, $t_i \geq t_{i+1}$ pour i=1,..., n. Un problème réduit de m+1 opérations est alors considéré. Une borne inférieure du temps de cycle de ce problème réduit est donc égale à $t_m + t_{m+1}$ parce que au plus deux

opérations sont exécutées sur une même station. De même, pour un problème de 2*m*+1 opérations, $t_{2m-1}+t_{2m}+t_{2m+1}$ est une borne inférieure. Généralement, LC2 est défini par :

$$LC2 = \text{Max}\left\{\sum_{i=0}^{k} t_{k.m+1-i} \middle| k = 1,\ldots,\lfloor (n-1)/m \rfloor\right\} \qquad 1.\ 27$$

où $\lfloor x \rfloor$ est le plus grand entier inférieur ou égal à *x*.

LC3 : Cette borne inférieure est basée sur les stations au plus tôt et au plus tard de chaque opération et l'amélioration successive de la borne inférieure. Nous définissons une borne inférieure du temps de cycle initial $\underline{c}$, les stations au plus tôt et au plus tard pour l'opération *i* sont données par :

$$E_i(\underline{c}) := \left\lceil (t_i + \sum_{h \in p_i^*} t_h)/\underline{c} \right\rceil \qquad 1.\ 28$$

$$L_i(\underline{c}) := m+1-\left\lceil (t_i + \sum_{h \in F_i^*} t_h)/\underline{c} \right\rceil \text{ pour } i=1,\ldots..,\ n, \qquad 1.\ 29$$

où F_i^* (P_i^*) est l'ensemble de successeurs (prédécesseurs) de l'opération *i*
LC3 est calculée en augmentant $\underline{c}$ jusqu'à ce que $E_i(\underline{c}) \le L_i(\underline{c})$ soit obtenue pour toutes les opérations *i*=1,...,*n*.

LC4 : Les stations [1,*m*] sur la ligne sont subdivisées en deux stations virtuelles [1,*k*] et [*k*+1,*m*] ([ARM 99] et [ARM 06]). Pour une borne inférieure du temps de cycle initial $\underline{c}$, les deux stations virtuelles obtiennent des temps de cycle $k^*\underline{c}$ et $(m-k)^*\underline{c}$, respectivement. Les opérations respectant la contrainte $L_i(\underline{c}) \le k$ sont assignées dans la première station virtuelle et celles respectant la contrainte $E_i(\underline{c}) > k$ sont regroupées dans la deuxième station virtuelle. Les opérations restantes et les temps mort des deux stations virtuelles constituent un problème résiduel. Ce problème est un problème de sac à dos pour répartir les opérations restantes aux deux sacs avec des capacités égales aux temps mort de ces deux stations virtuelles. Si une solution faisable existe, la borne inférieure est trouvée. Si la solution faisable n'existe pas, $\underline{c}$ augmente et le processus se répète jusqu'à ce qu'une solution faisable soit trouvée.

LC5 : Cette borne inférieure est similaire à celle du SALBP-1 proposée par [TAL 84] Talbot et Patterson. Elle est donnée par :

$$LC5 := \mathrm{Min}\left\{ c \,\middle|\, \sum_{k=1}^{m} TM_k(c) \le TML(c) \right\} \qquad 1.30$$

où $TML(c) = c*m-t_{sum}$ et $TM_k(c)$ est le temps mort de la station k au regard du temps de cycle d'essai c qui est un temps de cycle estimé par des expériences. La valeur de $TM_k(c)$ est calculée en considérant un problème de sac à dos [PLA 85] avec la capacité c pour toutes les stations $k = 1,...,m$, c'est à dire que nous regroupons les opérations de façon à maximiser les temps de stations qui ne doivent pas dépasser le temps de cycle c sans considérer les contraintes de précédence.

Les bornes inférieures présentées sont faciles à calculer même pour des problèmes de grandes dimensions. Mais aucune méthode ne peut assurer la précision de cette borne inférieure. C'est à dire qu'elles ne peuvent pas assurer que la borne inférieure est près du temps de cycle optimal pour un problème quelconque.

Borne supérieure

La borne supérieure du temps de cycle (en anglais, Upper Bound of Cycle Time– *UC*) est utilisée comme un temps de cycle initial pour le processus d'optimisation.

UC1 : la borne supérieure la plus simple [HAC 89] est donnée par :

$$UC1 := \mathrm{Max}\left\{ t_{\max}, 2 * \left\lceil t_{sum} / m \right\rceil \right\} \qquad 1.31$$

*UC*1 est facile à calculer mais quand la différence entre les temps d'opération n'est pas très grande, cette borne supérieure se trouve loin de la valeur du temps de cycle optimal. Dans ce cas, beaucoup plus de temps de simulation est nécessaire pour trouver le temps de cycle optimal.

UC2 : *UC*2 est donnée par si $LC1 > t_{max}$:

$$UC2 := \lceil t_{sum} / m \rceil + t_{\max} - 1 = LC1 + t_{\max} - 1$$

1. 32

UC2 est basée sur une solution infaisable construite selon LC1. Chaque opération divisée sur deux stations *k* et *k*+1 dans cette solution infaisable peut être assignée à la station *k* selon *UC2*, parce que le temps passé par une telle opération dans la station *k*+1 n'est pas plus grand que t_{max}-1.

Les bornes inférieure et supérieure sont utilisées dans différentes heuristiques proposées dans la littérature.

1.3.3.2 Heuristiques pour le SALBP de type II

Nous présentons ici brièvement les heuristiques pour générer une solution pour le problème simple de l'équilibrage des lignes d'assemblage de type II. La solution trouvée peut être le résultat final pour les entreprises (ex : l'heuristique pour le problème de type I, *COMSOAL*, est utilisée par un logiciel *Flexible Line Balancing* pour déterminer des solutions utilisables) ou insérée dans une méta-heuristique comme solution initiale ([WEN 98]).

La plupart des heuristiques pour le SALBP de type II utilisent une méthode de recherche qui résout un problème de SALBP de type I ([PON 99], [ARC 66]) afin de trouver le temps de cycle le plus petit correspondant au nombre de stations défini. Donc ce problème devient un problème SALBP-F. Dans la suite, nous présentons deux stratégies de recherche ([LIU 03], [LIU 05] et [ARM 96]) :

- **la méthode basée sur la borne inférieure :** Cette méthode débute en définissant le temps de cycle comme étant égal à une des bornes inférieures présentées plus haut, souvent *c*=*LC*1, parce que *LC*1 est plus facile à déterminer. Ensuite, une heuristique du SALBP de type I est utilisée pour trouver une solution respectant le temps de cycle *c*. Si le nombre de stations de la solution n'est pas supérieure au nombre fixé, le processus est terminé et *c* est la valeur du temps de cycle pour la ligne. Sinon *c* est augmenté par Δ et le processus se répète jusqu'à un critère d'arrêt ou l'obtention d'une solution. Généralement Δ = 1. Quand il y a un grand intervalle entre le temps de cycle minimum et la borne inférieure, Δ peut être défini comme un nombre de Fibonacci [WEL 86] de façon à réduire au maximum le temps de simulation ([ARM

96] et [ARM 99]).

- **La méthode de recherche bi-directionnelle :** Dans chaque itération, le temps de cycle est $c=\lfloor (LC+UC)/2 \rfloor$. Une heuristique du SALBP-1 est choisie et les opérations sont assignées aux stations des deux extrémités de la ligne. Si on ne trouve pas une solution avec le nombre de stations plus petit que le nombre donné *m*, alors la borne supérieure *UC* est redéfinie par $\max\{ST_k \mid k=1,\ldots,m\}$. Sinon *LC* est augmenté de 1. Le processus termine quand *LC=UC*. Lui, Ong et Huang ont développé ce type d'heuristiques, dans leurs travaux [LIU 03]. Ils ont utilisé une heuristique de type I, *Hoffmann backtracking procédé* [PON 99] pour générer les solutions faisables. A cette heuristique, ils ont ajouté une procédure d'amélioration qui échange les positions d'assignation de deux opérations exécutées par deux stations ou sélectionne aléatoirement une opération d'une station et la transfère sur une autre station. Dans ce processus, les contraintes de précédence doivent être respectées. Cette méthode est appelée *transfert et échange* (en anglais, transfer and trade) ([MCM 98], [PON 99]).

Les heuristiques présentées pour le SALPB de type II sont généralement basées sur les heuristiques utilisées pour le SALPB de type I avec la définition d'un temps de cycle initial et la répétition de ces heuristiques jusqu'à ce que le nombre de stations obtenu pour la ligne corresponde bien au nombre de stations fixé pour le SALPB de type II (tant que ce n'est pas le cas on augmente le temps de cycle). Par conséquent, la performance de ces heuristiques dépend des heuristiques de type I utilisées. Si l'heuristique ne peut pas assurer de trouver la meilleure solution pour le SALPB de type I, alors elle ne peut pas trouver la meilleure solution pour le SALPB de type II. D'autre part, le temps de calcul de ce type d'heuristiques dépend fortement de la valeur choisie pour le temps de cycle initial (le temps de cycle initial est défini souvent comme la borne inférieure ou supérieure). Si la différence entre le temps de cycle optimal et le temps de cycle initial est grande, le temps de calcul nécessaire est grand également.

1.3.4 Optimisation du SALBP de type II par des Méta-heuristiques

Les méta-heuristiques forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation complexes pour lesquels nous ne connaissons pas de méthode classique plus efficace. Dans le domaine de l'équilibrage de la ligne, plusieurs méta-heuristiques ont montré leurs avantages en termes d'efficacité et de la qualité de la solution.

La recherche tabou est utilisée par Chiang [WEN 98] pour un problème de type I. Il utilise deux stratégies d'amélioration *: le premier ajustement* et *le meilleur ajustement* qui choisi la première ou la meilleure solution respectivement qui est meilleure que la solution courante dans le voisinage. Une stratégie d'agrégation des opérations est alors proposée qui permet de transférer des opérations dans la station la moins chargée par rapport aux autres stations. Cette stratégie augmente donc la chance de diminuer le nombre de stations. Dans [LAP 06], les auteurs minimisent aussi le nombre de stations par la recherche tabou. Ils proposent une relaxation de la contrainte de temps de cycle si la relaxation permet de réduire une station.

Dans [MCM 06] et [JOA 07], les auteurs utilisent la méthode dite de colonie de fourmis pour résoudre des problèmes plus complexes. Dans [JOA 07], des contraintes liées à la position et aux stations (*voir 1.2.2*) sont ajoutées dans le système. Et dans [MCM 06], plusieurs objectifs sont optimisés en même temps. Dans cet article, une fonction objectif est construite qui considère à la fois le nombre d'opérateurs, l'utilisation du système, la probabilité des opérations non exécutées dans le temps de cycle et le coût de conception. Les algorithmes commencent tous par une solution initiale qui est améliorée par la méthode de transfert et échange (voir *1.3.2.2*).

Surech et Sahu ([SUR 94]) proposent une approche basée sur le **recuit simulé** qui prend en compte le problème des durées aléatoires des opérations. McMullen et Frazier ([MCM 98]) utilisent le recuit simulé pour un problème d'équilibrage multi-objectif. Erel, Szbuncuoglu et Aksu ont étudié une ligne de forme U à l'aide du recuit simulé. L'avantage de cette méthode est la facilité de mise en oeuvre. Mais les paramètres de la méthode sont difficiles à déterminer. Dans [SUR 94], [MCM 98], [ERE 02], les auteurs définissent les paramètres du recuit simulé par expérimentation. D'autre part, la qualité des solutions obtenues par le recuit

simulé est déterminée aussi par la technique employée pour modifier la solution courante afin de générer une nouvelle solution. Quatre méthodes ont été utilisées pour créer l'espace de voisinage. Ces méthodes peuvent être utilisées dans la recherche tabou et pour les colonies de fourmis.

Une autre méta-heuristique qui est basée sur la génération des populations, appelé ***algorithme génétique,*** a aussi été utilisée pour résoudre le problème d'équilibrage de la ligne d'assemblage ([SUR 96], [BRA 99]). Le principal avantage des algorithmes génétiques réside dans le fait qu'ils ne travaillent pas avec une seule solution mais avec toute une famille (une population) de solutions en même temps. Cet avantage rend les algorithmes génétiques efficaces pour des problèmes complexes comme par exemple, les problèmes multi-objectif ([YEO 96], [GEL 99], [PON 00]). En effet, pour les problèmes multi-objectif le but est d'obtenir un ensemble de solutions dites Pareto optimales. Il existe plusieurs types d'algorithmes génétiques proposés pour des problèmes multi-objectif, mais ils ne sont pas faciles à réaliser en raison d'un paramétrage difficile (la sélection des parents, le taux de mutation, etc.).

1.4 Conclusion

Dans ce chapitre, nous avons réalisé une étude de l'état de l'art sur l'équilibrage de lignes d'assemblage. Dans une première partie, la définition dee lignes d'assemblage est donnée. Nous détaillons ses caractéristiques et ses applications dans l'industrie. Ensuite, une présentation globale des problèmes d'équilibrage de lignes d'assemblage (ALBP) est donnée. Plusieurs classifications des problèmes ALBP sont proposées selon le type de la ligne ou les objectifs à optimiser. Dans nos travaux de recherche, nous nous intéressons au problème de l'équilibrage de lignes d'assemblage dans le cas où le nombre de stations est connu et fixé (SALBP-2). En fait, le problème SALBP-2 correspond à la mise en production d'un nouveau modèle de produit sur une ligne existante. C'est le cas des produits de biens de grande consommation qui sont renouvelés très souvent, (ex. des portables, des voitures, des ordinateurs). Il est donc important de développer des méthodes efficaces pour le problème d'équilibrage de lignes d'assemblage de type II.

Plusieurs méthodes d'optimisation ont été utilisées pour résoudre les problème d'équilibrage de lignes d'assemblage. Les méthodes exactes sont bien sur les plus précises, mais elles ne

peuvent pas résoudre les problèmes qui ont un grand nombre d'opérations et stations, parce que le temps nécessaire est trop long. Par conséquent, des heuristiques ont aussi été proposées pour résoudre ce type de problème efficacement. Mais les heuristiques ne peuvent garantir que la solution trouvée est la meilleure possible. La plupart des heuristiques pour le SALBP de type II utilisent une méthode de recherche qui résout un problème de SALBP de type I afin de trouver le temps de cycle le plus petit correspondant au nombre de stations défini. Dans ce cas, la performance des heuristiques pour les problèmes de type II dépend plutôt de celle des heuristiques de type I qui sontt utilisées dans le processus. La performance est donc parfois insuffisante. Des bornes supérieures et inférieures sont parfois utilisées par les méthodes exactes et pour les heuristiques. En effet une borne qui est près du temps de cycle optimal permet de réduire le temps de calcul. Lorsque les problèmes deviennent trop complexes ou dans le cas multi-objectif, les méta-heuristiques deviennent des outils attractifs pour résoudre le problème mais bien souvent un grand nombre de paramètres doivent être définis pour que la méta-heuristique soit efficace.

C'est pourquoi, dans nos travaux nous nous sommes intéressés tout d'abord à la définition d'une borne inférieure qui soit fiable. Puis nous avons défini une heuristique adaptée au SALBP-2.

Ces éléments ont ensuite été intégrés dans la définition d'une méta-heuristique particulière, ne nécessitant pas la définition de plusieurs paramètres, et utilisée dans le cas déterministe ou stochastique avec un ou plusieurs objectifs à optimiser.

Ce mémoire est organisé de la manière suivante. Dans le chapitre deux, nous proposons des méthodes heuristiques et méta-heuristiques pour résoudre les problèmes d'équilibrage de lignes d'assemblage de type II déterministes. Ces problèmes correspondent au cas de temps opératoires déterministes avec pour objectif la minimisation du temps de cycle de la ligne. Dans le chapitre suivant, nous avons étudié le problème d'équilibrage de lignes d'assemblage manuelles de type II caractérisé par des durées opératoires aléatoires. Pour ce type de lignes, nous avons utilisé une méta-heuristique et nous avons considéré une mesure de performance supplémentaire, qui est la fiabilité de la ligne. Nous avons alors considéré cette fiabilité selon deux approches : pour la première approche, nous considérons la fiabilité comme une contrainte supplémentaire à notre problème et pour la deuxième approche, nous considérons la fiabilité comme un critère supplémentaire. Dans ce cas, la fiabilité est définie comme la probabilité d'incomplétude des opérations, et nous cherchons alors à minimiser le temps de cycle de la ligne et minimiser cette probabilité d'incomplétude, ces deux critères étant

opposés. Enfin, dans le dernier chapitre, nous proposons une méthode basée sur la génération de colonnes pour déterminer une borne inférieure du temps de cycle du problème de l'équilibrage de la ligne d'assemblage de type II.

Chapitre 2.

Optimisation du temps de cycle d'une ligne d'assemblage déterministe

Dans ce chapitre, nous nous intéressons au problème de minimisation du temps de cycle d'une ligne d'assemblage de type II avec des temps opératoires déterministes. Ainsi, dans un premier temps, nous proposons une heuristique définie selon les spécificités du problème d'équilibrage de lignes d'assemblage de type II. Cette heuristique donne de bons résultats pour des problèmes relativement simples. Pour le cas de problèmes plus complexes, nous développons la méthodologie EM ainsi que la méthodologie ED et nous comparons ces méthodes avec le recuit simulé.

2.1 Introduction

Dans ce chapitre, nous considérons le problème d'équilibrage de lignes d'assemblage de type II avec des temps opératoires déterministes, c'est à dire le problème de la minimisation du temps de cycle pour un nombre de stations donné. Quand une ligne d'assemblage est entièrement automatisée, les temps opératoires sont plus ou moins constants. Dans ce cas là, si les pannes ne sont pas considérées, la représentation du système de production par un modèle déterministe est justifiée. Dans ce cas et pour rappel, nous étudions les problèmes de type II qui se posent généralement quand une entreprise veut produire un certain nombre de produits en utilisant les ressources existantes sans acheter de nouvelles machines ou sans employer d'ouvriers supplémentaires. Pour le modèle déterministe, le temps de cycle à minimiser est le critère, et il est donné par le plus grand temps de station de la ligne.

Bien que le problème d'équilibrage des lignes d'assemblage soit un problème d'optimisation combinatoire très connu, la plupart des heuristiques existantes sont proposées pour les problèmes de type I ([PON 99], [HAC 89]). Les heuristiques proposées pour les problèmes de type II sont généralement dérivées de ceux définis pour les problèmes de type I. En effet, les heuristiques pour les problèmes de type II commencent par la définition d'une borne inférieure du temps de cycle et applique ensuite une heuristique issue des problèmes de type I pour affecter les opérations aux stations et produire une solution initiale (voir Section 1.3.3.2). Si le nombre de stations de la solution obtenue est plus grand que le nombre donné, l'heuristique recommence avec un temps de cycle plus grand. Autrement le processus de calcul s'arrête. D'après ([ARM 96], [ARM 99]), deux types d'heuristiques basées sur cette idée sont proposés pour résoudre les problèmes de type II. Si les opérations sont affectées à partir de la première station de la ligne et ce jusqu'à la fin de la ligne, l'heuristique est dite mono-directionnelle. Et si les opérations sont affectées par les deux cotés de la ligne, alors l'heuristique est dite bi-directionnelle. Les performances de ces heuristiques dépendent de celles de l'heuristique utilisée pour le problème de type I. Autrement dit, l'heuristique ne peut pas obtenir de bons résultats pour les problèmes de type II si l'heuristique des problèmes de type I ne donne pas de bons résultats. Pour éviter ces inconvénients, nous proposons une heuristique adaptée aux problèmes déterministes de type II pour construire une solution faisable du problème de type II [GU 07c]. Notons que notre heuristique ne peut pas résoudre des problèmes complexes comme par exemple les problèmes stochastiques ou de grande taille.

Les méta-heuristiques sont alors une bonne alternative.

Les méta-heuristiques sont des méthodes intéressantes (voir Section 1.3.4) pour résoudre les problèmes d'équilibrage de lignes d'assemblage. En effet, elles peuvent trouver de bonnes solutions même pour des problèmes complexes. Dans [LIV 06], un algorithme génétique est proposé pour maximiser la productivité d'une ligne d'assemblage robotisée. Le problème consiste à affecter les opérations aux stations et à choisir pour chaque station le type de robot le mieux adapté parmi un nombre donné de types différents de robots. L'algorithme génétique proposé travaille avec une représentation partielle d'une solution du problème qui est une permutation des opérations respectant les contraintes de précédence. Deux procédures sont ensuite utilisées pour affecter les opérations aux stations suivant la permutation donnée et pour choisir pour chaque station le robot le mieux adapté. Les résultats de l'algorithme génétique sont ensuite améliorés par une optimisation locale. Les auteurs montrent que les résultats de l'algorithme génétique proposé sont meilleurs que les résultats d'une procédure par séparation et évaluation tronquée. Suresh, quant à lui, se concentre sur des problèmes stochastiques de la minimisation du coût de production et les résout à l'aide de la méthode du recuit simulé et d'un algorithme génétique ([SUR 94] et [SUR 96]). L'algorithme génétique proposé dans [SUR 96] utilise deux populations de solutions dans chaque itération. Dans la première population, toutes les solutions sont admissibles et dans la seconde, des solutions infaisables sont acceptées. Ces deux populations sont utilisées ensuite pour générer de nouvelles populations. Cette méthode a l'avantage d'augmenter la variété des solutions. Sarin et al. considèrent aussi des problèmes stochastiques pour minimiser les coûts liés à la main-d'œuvre et aux opérations non complètement achevées dans un temps de cycle donné [SUB 99]. Une méthode de programmation dynamique a été utilisée pour construire une solution initiale qui est ensuite améliorée par une procédure de séparation et d'évaluation utilisant une solution approximée à la place de la borne inférieure pour éliminer les nœuds.

Ces différents travaux montrent l'utilité des méta-heuristiques pour résoudre des problèmes complexes. Cependant, la performance des méta-heuristiques dépend du choix de ses différents paramètres. En général, les chercheurs configurent les paramètres par expérimentation. Dans ce cas là, la performance des méta-heuristiques ne peut pas être prouvée. Nous avons choisi d'utiliser deux méta-heuristiques : l'algorithme basé sur le mécanisme d'électromagnétisme (electromagnetism-like mechanism (EM)) ainsi que l'estimation de distribution (ED) du fait de leur nombre réduit de paramètres à définir. Nous

comparons nos résultats avec le recuit simulé (SA)([GU 07b], [GU 07d]).

Le reste du chapitre est organisé comme suit. D'abord, nous présentons l'heuristique proposée et l'utilisation de cette heuristique pour résoudre des problèmes déterministes d'équilibrage de lignes d'assemblage. Ensuite, nous présentons les deux méta-heuristiques : EM et ED (SA est donné en Annexe SA). Puis, nous utilisons ces méta-heuristiques pour résoudre le problème considéré. Une comparaison des résultats de simulation est présentée à la fin de ce chapitre.

2.2 Une heuristique pour le SALBP type II

Nous proposons ici une nouvelle heuristique pour minimiser le temps de cycle des lignes d'assemblage dans le cas où le nombre de stations est défini et les temps opératoires sont déterministes. L'heuristique proposée est constituée de deux phases. Dans la première phase, une solution initiale est produite à partir d'un poids défini selon la position des opérations dans le graphe de précédence et d'un certain seuil défini pour chaque station. Puis, la solution est améliorée par une procédure de transfert et d'échange [RAC 91].

2.2.1 Définition d'une solution initiale

Dans cette phase, nous proposons une nouvelle heuristique pour produire une première solution du SALBP-2. Cette heuristique est basée sur des paramètres associés aux opérations et aux stations que nous définissons ici.

Le poids de position d'une opération *i*, dénoté PW_i, est donné par la somme des temps opératoires de toutes les opérations qui suivent cette opération dans le graphe de précédence. Soit $F^*(i)$ l'ensemble des opérations qui suivent l'opération *i* dans le graphe de précédence. Le poids de position de l'opération *i* peut être défini par l'équation suivante :

$$PW_i = \sum_{h \in F^*(i)} t_h \quad \text{pour } i=1,..., n \qquad 2.1$$

où *n* est le nombre d'opérations.

A chaque station *k* est associé un seuil TS_k donné par l'équation suivante :

$$TS_k = (m-k+1)* \delta \quad k=1,..., m \qquad 2.2$$

où $\delta=\frac{\sum_{i=1}^{n} t_i}{n}$, et n est le nombre d'opérations.

Les opérations sont affectées aux stations comme suit. Si le poids PW_i de position de l'opération i est supérieur au seuil TS_1 de la station 1, nous l'affectons à la station 1. Si PW_i est inférieure au seuil TS_{m-1} de la station m-1, nous l'affectons à la station m. Dans les autres cas, l'opération i est affectée à la station k telle que PWi est compris entre TS_{k-1} et TS_k. La règle de construction de la solution initiale peut être représentée par l'algorithme suivant :

Algorithme 2.1 : Définition d'une solution initiale

Etape 1 : Déterminer les poids de position de chaque opération en utilisant l'équation (2. 1).

Etape 2 : Déterminer les seuils des stations à l'aide de l'équation (2. 2).

Etape 3 : Affecter l'opération i à la station 1 (respectivement à la station m), si la condition suivante est satisfaite :

$$PW_i > TS_1 \text{ (respectivement } PW_i \leq TSm_{-1}) \qquad 2.\ 3$$

Sinon, l'opération i sera affectée à la station k telle que :

$$TS_k \leq PW_i < TS_{k-1} \qquad 2.\ 4$$

Etape 4 : Répéter l'étape 3 jusqu'à ce que toutes les opérations soient affectées.

Un exemple de la construction de solutions initiales est donné dans ce qui suit.

Le problème est présenté dans la Fig. 1. 2. Le nombre de stations est égal à 6. Les poids de position des opérations qui sont déterminés selon l'équation (2. 1) sont présentés dans le Tableau 2. 1.

	Opt 1	Opt 2	Opt 3	Opt 4	Opt 5	Opt 6	Opt 7	Opt 8	Opt 9	Opt 10
PW_i	32	17	27	22	18	13	13	11	2	0

Tableau 2. 1 Poids de position des opérations

Les seuils des stations sont déterminés selon l'équation (2. 2) et présentés dans le Tableau 2. 2.

	Station 1	Station 2	Station 3	Station 4	Station 5	Station 6
ST_i	28.8	24	19.2	14.4	9.6	4.8

Tableau 2. 2 Seuils des stations

Nous construisons la solution initiale selon l'algorithme 2.1. Cette solution initiale est présentée dans le Tableau 2. 3.

	Station 1	Station 2	Station 3	Station 4	Station 5	Station 6
affectation	Opt_1	Opt_3	Opt_4	Opt_5, Opt_2	Opt_6, Opt_7, Opt_8	Opt_9, Opt_{10}

Tableau 2. 3 Solution initiale

Théorème 2.1 : Toute solution donnée par l'algorithme 2.1 respecte les contraintes de précédence.

***Preuve** : Si l'opération i précède l'opération h, la condition $PW_i \geq PW_h + t_h$ est vraie. Alors la station où l'opération h est affectée devra être positionnée après la station qui effectue l'opération i ou encore les deux opérations sont affectées à la même station en fonction des seuils des stations.*

QED.

Remarque : la solution initiale obtenue par cette heuristique peut ne pas être performante dans le cas de lignes d'assemblage particulières comme le cas d'opérations définies sans contrainte de précédence.

2.2.2 Procédure de transfert et d'échange

Nous n'avons pas cherché à équilibrer explicitement les charges des différentes stations lors de la construction de la solution initiale. Nous utilisons ainsi une procédure de transfert et d'échange pour améliorer cette solution initiale. La procédure de transfert et d'échange exécute deux actions (le transfert et l'échange) sur l'affectations des opérations pour équilibrer les charges des stations de la ligne. L'action de *transfert* choisit une opération *i* d'une station *k*, et l'affecte à une autre station. L'action d'échange permute les positions de deux opérations

qui ne sont pas affectées à la même station. Les contraintes de précédence doivent être toujours respectées, ce qui interdit certaines actions.

Nous améliorons la solution initiale en deux phases. En phase 1, nous examinons les stations deux à deux. Pour chaque paire de stations k_1 et k_2, nous cherchons et effectuons les actions de transfert et d'échange permettant de réduire le temps de cycle. En phase 2, nous effectuons un nombre *NUM* donné d'actions comme suit. A chaque itération de la phase 2, deux stations s_1 et s_2 sont choisies au hasard et nous cherchons et effectuons une action de transfert ou d'échange qui peut dégrader le temps de cycle mais dont la dégradation ne dépasse pas un seuil maximal donné ω défini par des expériances. La procédure d'amélioration que nous proposons est donnée par :

Algorithme 2.2 : Transfert et échange

Pour k_1 de *1* à *m*-1

  Pour k_2 de k_1 à *m*

    Exécuter toutes les actions de transfert et d'échange entre les stations k_1 et k_2 tant que le temps de cycle *c* n'est pas augmenté ;

Définir *c_best=c* ;

Pour *k* de *1* à *NUM*

  Choisir deux stations s_1 et s_2 aléatoirement ;

  Exécuter toutes les actions de transfert et d'échange qui n'augmentent pas le temps de cycle *c* au-delà du seuil ω défini ;

  Si *c* < *c_best* ;

    c_best = *c* ;

    Mémoriser la solution courante comme la meilleure solution obtenue ;

Fin

Remarque : L'heuristique proposée traite le problème déterministe d'équilibrage de lignes d'assemblage de type II. Il génère d'abord une solution initiale selon les poids de position des opérations et les seuils des stations. Une procédure d'amélioration de type transfert et échange est ensuite employée pour lisser les charges des stations de sorte que le temps de cycle de la ligne d'assemblage soit réduit au minimum. L'avantage de cette heuristique se situe sur la première phase. Elle améliore le processus de génération de la solution initiale. L'heuristique

ainsi définie permet de gagner du temps de calcul. Cette heuristique est très efficace pour résoudre des problèmes d'équilibrage de lignes d'assemblage simples. Cependant, elle n'est pas adapté à des problèmes plus complexes comme par exemple l'équilibrage d'une ligne de production stochastique.

2.3 Méta-heuristiques pour l'équilibrage des lignes déterministes

Les méta-heuristiques sont aussi souvent utilisées pour résoudre les problèmes SALBP-2. Elles ont prouvé leur efficacité par rapport aux méthodes exactes et en même temps la qualité de la solution trouvée est souvent meilleure que celle obtenue par une heuristique.

Nous utilisons deux méthodes basées sur des méta-heuristiques pour résoudre les problèmes SALBP-2. Les méthodes choisies sont l'algorithme basé sur le mécanisme électromagnétique (EM) ([PEI 04]) et l'estimation de distribution (ED). Nous comparerons nos meta-heuristiques avec le recuit simulé (SA). Nous présentons dans ce qui suit chacune des méthodes utilisées dans le cadre du SALBP-2.

2.3.1 EM pour le problème SALBP-2

Nous développons ici une méta-heuristique EM pour résoudre les problème de SALBP-2. Nous présentons d'abord les principes généraux de EM, puis, nous détaillons comment appliquer EM à notre problème.

2.3.1.1 Présentation de la méthode EM

EM est une méta-heuristique par évolution de populations de solutions pour des problèmes d'optimisation globale, comme par exemple la minimisation de fonctions non-linéaires *f(x)* ([BIR 03] et [PEI 04]). Chaque solution du problème correspond à un point dans un espace Euclidien. EM utilise un mécanisme d'attraction - répulsion issu de la théorie de l'électromagnétisme pour déplacer une population de points vers l'optimum.

EM commence par générer aléatoirement des points dans la région admissible. Ces points sont considérés comme des particules électriques chargées. La charge des points échantillonnés est déterminée par la valeur de la fonction objectif pour ces points. L'équation (2. 5) montre un exemple du calcul de la charge d'un point *i* :

$$q^t = \exp\left(-\beta \frac{f(x^t) - f(x^{best})}{\sum_d^{\pi} (f(x^d) - f(x^{best}))} \right) \qquad 2.5$$

où $f(x^t)$ est la valeur de la fonction objectif du point t,

$f(x^{best})$ est la valeur du critère de la meilleure solution trouvée,

β est la dimension de l'espace de solutions ou de l'espace Euclidien considéré,

π est le nombre de points dans chaque population.

Puis, selon la théorie de l'électromagnétisme, une force est déterminée pour chaque point selon l'équation suivante :

$$Force_t = \sum_{w \neq t}^{\pi} \begin{cases} (x_w - x_t) \dfrac{q_t q_w}{\left\| x_w - x_t \right\|^2} \text{ if } f(x_w) < f(x_t) \\ (x_t - x_w) \dfrac{q_t q_w}{\left\| x_w - x_t \right\|^2} \text{ if } f(x_w) \geq f(x_t) \end{cases} \quad t = 1,2,\ldots,\pi \qquad 2.6$$

Comme nous pouvons le voir dans l'équation (2. 6), entre deux points, le point avec la meilleure valeur de la fonction objectif attire l'autre. Au contraire, le point avec la valeur la plus mauvaise de la fonction objectif repousse l'autre. Après l'évaluation du vecteur de la force totale $Force_t$, le point t est déplacé dans la direction de la force totale selon un pas choisi aléatoirement comme :

$$x_t = x_t + \lambda \frac{Force_t}{\left\| Force_t \right\|} (RNG), \; t = 1,2,\ldots,\pi \qquad 2.7$$

où λ est un pas aléatoire qui suit une distribution uniforme entre 0 et 1, *RNG* est un vecteur qui dénote le mouvement faisable vers la borne inférieure ou supérieure pour la dimension correspondante.

L'algorithme de la méthode EM (sans adaptation à notre problème) est donné par :

Algorithme 2.3 : EM

Etape 1 : Choisir aléatoirement la population des solutions initiales et les évaluer à l'aide de la fonction objectif. Les solutions sont considérées comme des particules électriquement chargées.

Etape 2 : Calculer la charge des solutions dans la population courante suivant l'équation (2. 5).

Etape 3 : Evaluer la force totale de chaque solution selon l'équation (2. 6).

Etape 4 : Déplacer les solutions courantes selon l'équation (2. 7). Evaluer les nouvelles solutions.

Etape 5 : Tant que la condition d'arrêt n'est pas atteinte, retourner à l'étape 2.

A partir de cet algorithme, nous voyons que EM est facile à configurer. En dehors du vecteur *RNG* et du pas, il n'y a pas d'autre paramètre à régler dans ce processus. Le vecteur *RNG* et le pas λ sont utilisés pour assurer que les points échantillonnés soient faisables. Le mécanisme de répulsion et d'attraction assure de trouver la bonne solution.
Mais cette méthode demande de coder les solutions en espace Euclidien. Cette limitation augmente la région de recherche et diminue donc l'efficacité de l'algorithme.

2.3.1.2 Résoudre le SALBP-2 par EM

Représentation du problème

Le codage des solutions que nous proposons combine l'idée de *clés aléatoires* proposée dans [BEA 94] et une heuristique dédiée au SALBP-1 proposée par Kilbridge and Wester dans [PON 99]. Notre méthode EM transforme un espace de solutions en espace de clés aléatoires sur $[0 ; 1]^n$. Plus spécifiquement, nous associons à chaque opération i une clé aléatoire r_i générée suivant une loi uniforme dans l'intervalle [0 ; 1]. Par conséquent, chaque solution du SALBP-2 est représentée par un vecteur de clés aléatoires R de dimension n où n est le nombre d'opérations.

Dans la transformation d'un vecteur de clés aléatoires en solution du SALBP-2, les clés aléatoires sont considérées comme les poids des opérations dans l'heuristique du SALBP-1. Les temps opératoires sont pris en compte dans cette transformation. La transformation commence par une borne inférieure du temps de cycle déterminée par l'équation (2. 8) (voir également Section 1.3.3.1) :

$$\underline{c} = \mathrm{Max}\left\{t_{\max}, \left\lceil t_{sum} / m \right\rceil\right\} \qquad 2.\ 8$$

Les opérations dont les prédécesseurs sont déjà affectés ou qui n'ont aucun prédécesseur sont choisies et affectées à une station selon l'ordre croissant de leur *clé aléatoire* jusqu'à ce que le temps de cycle $\underline{c}$ soit obtenu. Ensuite une autre station est ouverte. Ce processus est répété jusqu'à ce que toutes les opérations soient affectées. Enfin, si le nombre de stations est plus grand que celui défini, le temps de cycle est augmenté et le processus est répété jusqu'à ce qu'une solution avec le nombre de station donné *m* soit obtenu. La solution obtenue est considérée comme la solution du SALBP-2 correspondant au vecteur de *clés aléatoires R*.

Pour chaque vecteur de *clés aléatoires R*, l'algorithme suivant est employé pour le convertir en solution de SALBP-2. *RT* dénote le temps restant pour la station courante.

Algorithme 2.4 : Représentation dans EM

Etape 1 : Choisir un temps de cycle faisable grâce à l'équation (2. 8).

Etape 2: SALBP-2 respectant le temps de cycle déterminé dans l'étape 1 :

2.1 : Ouvrir une nouvelle station avec $RT = \underline{c}$, définir $\Delta = \underline{c}$;
2.2 : Déterminer l'ensemble *S* d'opérations sans prédécesseur dans le graphe de précédence;
2.3 : Ordonner $S = \{[1], [2], ...\}$, tel que $r_i \leq r_{i+1}$;
2.4 : Déterminer l'opération *i* dans *S* avec la clé aléatoire la plus petite, ainsi que $t_i \leq RT$;
2.5 : Déterminer $\eta = MIN\{t_k: k < i\} - RT$,
Faire $\Delta = MIN\{\Delta, \eta\}$;
2.6 : Affecter l'opération *i* à la station courante, la supprimer du graphe de précédence et recalculer $RT = RT - t_i$;
2.7 : S'il n'y a plus d'opération dans le graphe de précédence, arrêter le processus. Sinon retourner à l'étape 2.2 ;
2.8 : Si l'opération *i* n'existe pas, retourner à l'étape 2.1.

Etape 3 : Si le nombre de stations est plus grand que *m*, augmenter $\underline{c} = \underline{c} + \Delta$ et retourner à l'étape 2. Sinon prendre la solution obtenue pour l'équilibrage de la ligne d'assemblage comme la solution correspondante à la *clé aléatoire R*.

Théorème 2.2 : *Pour toute solution admissible X du SALBP-2 avec un temps de cycle c, alors $c' \leq c$ où c' est le temps de cycle de la solution générée par l'algorithme 2.4 avec un*

vecteur R de clés aléatoires tel que $r_i < r_j$ pour deux opérations i et j telles que la station de l'opération i précède celle de l'opération j dans la solution X.

Ce théorème montre que le codage que nous avons choisi préserve les solutions optimales et exclut des solutions qui ne peuvent pas être optimales. Il est également possible de démontrer que la transformation est polynomiale. Notons que différents vecteurs de clés aléatoires peuvent correspondent à la même solution du SALBP-2.

Génération des solutions initiales

EM est utilisée pour améliorer les solutions du SALBP-2. Par contre, EM ne peut pas générer la population de solutions initiales. EM a donc besoin d'une heuristique pour générer les solutions initiales. Dans nos travaux, elles sont générées aléatoirement. L'idée est de générer aléatoirement les vecteurs des clés aléatoires distribuées suivant la loi uniforme. Les solutions sont donc générées à l'aide des vecteurs des clés aléatoires. Nous n'utilisons pas dans cette partie l'heuristique que nous avons développée dans la section 2.2 car cette heuristique ne définit qu'une seule solution et non une population de solutions.

Algorithme EM pour résoudre le SALBP-2

Notre méta-heuristique EM pour résoudre le SALBP-2 commence par échantillonner aléatoirement une population de solutions représentées par un ensemble de vecteurs de clés aléatoires. Chaque solution représente une ligne d'assemblage possible. Les solutions sont évaluées par les temps de cycle déterminés par les temps de stations maximums des lignes correspondantes aux solutions obtenues. Dès que les temps de cycle sont déterminés, les charges et les forces totales des solutions sont calculées par les équations (2. 5) et (2. 6). Puis, la nouvelle population de solutions peut être générée en déplaçant les solutions courantes selon leur force totale et l'équation (2. 7).

L'algorithme EM appliqué au problème SALBP-2 est donné par:

Algorithme 2.5 : EM pour le SALBP-2

Etape 1 : Initialisation du système étudié :
nombre d'opérations,
temps d'opérations,
nombre de stations,
contraintes de précédence.

Etape 2 : Générer les solutions initiales de EM :

2.1 : Générer aléatoirement un ensemble de vecteurs de clés aléatoires *R*. Les clés aléatoires sont distribuées selon une loi uniforme avec des valeurs comprises entre 0 et 1 ;

2.2 : Transformer l'ensemble des vecteurs de *clés aléatoires* en un ensemble de solutions du SALBP-2 selon l'algorithme 2.4 ;

2.3 : Evaluer les solutions obtenues selon leur temps de cycle déterminé par :

$$\underline{c}=\underset{1\le k\le m}{\text{Max}}\sum_{i=1}^{n} t_i\, x_{ik} \qquad 2.\ 9$$

2.4 : Mémoriser la meilleure solution dans la population initiale.

Etape 3 : Génération de nouvelles solutions :

3.1 : Calculer la charge des solutions dans la population courante selon l'équation (2. 5).

3.2 : Calculer la force totale des solutions par l'équation (2. 6).

3.3 : Déplacer les points selon l'équation (2. 7) pour générer une nouvelle population de solutions représentées par des vecteurs de *clés aléatoires* ;

3.4 : Transformer les vecteurs de clés aléatoires en solutions du SALBP-2 et évaluer ces solutions selon leur temps de cycle ;

3.5 : Mettre à jour la meilleure solution.

Etape 3 : Retourner à l'étape 3 si la condition d'arrêt n'est pas atteinte.

2.3.2 ED pour le SALBP de type II

L'estimation de distribution (ED) est une méta-heuristique inspirée des algorithmes génétiques.

2.3.2.1 Présentation de ED

L'estimation de distribution est utilisée pour résoudre des problèmes d'optimisation, via la manipulation d'un échantillonnage de la fonction objectif qui décrit la qualité des solutions faisables ([MUH 96], [OCE 02] et [TOP 02]). Comme toutes les méta-heuristiques utilisant une population de points, ED est itératif. À l'inverse des algorithmes d'évolution "classiques", le cœur de la méthode ED consiste à estimer les relations entre les différentes variables d'un problème d'optimisation, grâce à l'estimation d'une distribution probabiliste de la solution optimale. ED n'emploie donc pas d'opérateurs de croisement ou de mutation, l'échantillon

étant directement construit à partir des paramètres de distribution, estimés à l'itération précédente.

L'algorithme de ED commence par une population de solutions initiales. Une distribution de probabilité est alors choisie et utilisée dans l'algorithme. A chaque itération *i*, une nouvelle population de solutions est générée au hasard selon la distribution de probabilité choisie. On sélectionne ensuite un sous-ensemble de solutions considérées comme les "meilleures" solutions de la population. On établit ensuite une nouvelle estimation de la distribution probabiliste de la solution optimale à partir de la distribution du sous-ensemble des meilleures solutions de la population actuelle. Ce processus doit être répété jusqu'à la satisfaction d'une condition d'arrêt. Plus précisément, l'algorithme de ED procède comme suit :

Algorithme 2.6 : ED

Tirer au hasard l individus, pour former une population initiale D_0.

$i = 0$;

Tant qu'un critère d'arrêt n'est pas vérifié :

$i = i + 1$;

Sélectionner h "meilleurs" individus (avec $h < l$) dans la population précédente (D_{i-1}),

pour former un ensemble de solution : D_{i-1}^{se}.

Estimer une distribution de probabilité $P_i(x)$, décrivant la répartition de l'ensemble des solutions sélectionnées.

Tirer au hasard l individus dans $P_i(x)$.

Fin de la boucle.

Un exemple est donné dans la Fig. 2. 1. Dans cet exemple, une fonction objectif continue $f(x)$ est optimisée. Cette fonction a un seul optimum $\boldsymbol{O}$. Au fur et à mesure du déroulement de l'algorithme, l'échantillonnage qui suit une loi normale N se concentre autour de l'optimum.

A partir de la population totale D, à chaque itération i on effectue une sélection aléatoire d'une population étalon D^{se}, selon une distribution de probabilité ***PDu***. Les paramètres de la distribution ***PDe*** qui caractérisent les individus choisis (individus étalon appartenant à

l'ensemble D^{es}) sont ensuite estimés. Les paramètres de ***PDu*** sont remplacés par ceux de ***PDe***. Une nouvelle population est générée selon les nouvelles valeurs de la distribution ***Pdu***.

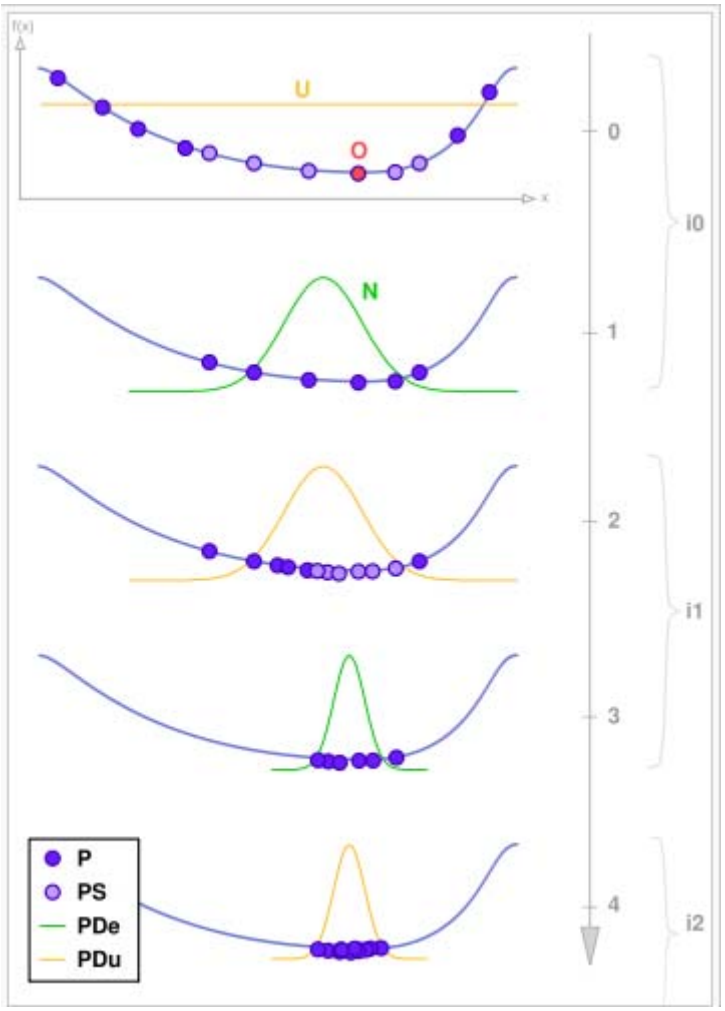


Fig. 2. 1 Processus de ED

ED contrôle la distribution de probabilité de recherche *p(x)* qui est construite de sorte qu'elle apprenne des informations sur la région intéressante. Par rapport aux autres algorithmes d'évolution, ED emploie un processus basé sur les statistiques au lieu d'imiter la nature. De plus, ED n'utilise pas beaucoup de paramètres. Il est donc très facile de définir ED pour optimiser des problèmes. Une définition appropriée de la famille de distributions probabilistes à utiliser pour caractériser la distribution de la solution optimale permet de garantir la convergence vers la solution optimale.

2.3.2.2 Résoudre le SALBP-2 par ED

Représentation du problème

Dans notre méta-heuristique ED, une solution du SALBP-2 est représentée par un vecteur de nombres entiers, dénoté par $Y \in Z^n$. La dimension de ce vecteur est égale au nombre d'opérations. Chaque élément y_i représente la station où l'opération i doit être affectée en priorité. Pour évaluer une solution *slt*, le vecteur correspondant Y_{slt} doit être transformé en une solution admissible. La méthode de transformation des vecteurs de représentation est donnée par l'algorithme suivant :

Algorithme 2.7 : Représentation dans notre méta-heuristique ED

Etape 1 : Choisir un temps de cycle faisable grâce à l'équation (2. 8).

Etape 2: construire une solution du SALBP-2 :

2.1 : Ouvrir une nouvelle station avec $RT = \underline{c}$, définir $\Delta = \underline{c}$;

2.2 : Déterminer l'ensemble S d'opérations sans prédécesseur dans le graphe de précédence;

2.3 : Ordonner $S = \{[1], [2], ...\}$, tel que $y_i \leq y_{i+1}$;

2.4 : Si i est une opération dans S avec le plus petit y_i, telle que $t_i \leq RT$;

2.4.1 : Déterminer $\phi = MIN\{t_h: h < i\} - RT$,
Faire $\Delta = MIN\{\Delta, \phi\}$;

2.4.2 : Affecter l'opération i à la station courante, la supprimer du graphe de précédence et recalculer $RT = RT - t_i$;

2.4.3 : S'il n'y a plus d'opération dans le graphe de précédence, arrêter le processus. Sinon retourner à l'étape 2.2 ;

2.5 : Sinon, retourner à l'étape 2.1.

Etape 3 : Si le nombre de stations utilisées est plus grand que m, augmenter $\underline{c} = \underline{c} + \Delta$ et retourner à l'étape 2. Sinon garder la solution obtenue comme solution finale du problème.

Théorème 2.3 : Pour toute solution admissible X du SALBP-2 avec temps de cycle c, alors $c' \leq c$ où c' est le temps de cycle de la solution générée par l'algorithme 2.7 avec codage Y correspondant à l'affectation dans la solution X.

Le théorème montre que le codage que nous avons choisi préserve les solutions optimales et exclut des solutions qui ne peuvent pas être optimales. Il est également possible de démontrer que la transformation est polynomiale.

Génération de la solution initiale

Les solutions initiales de ED sont aussi générées aléatoirement tout en respectant les contraintes de précédence.

Sélection des solutions étalon pour la méthode ED :

Dans chaque itération de l'algorithme ED, quelques solutions étalon doivent être choisies pour générer une nouvelle population. Dans nos travaux, un pourcentage des meilleures solutions de la population courante sont choisies comme solutions étalon. La meilleure solution obtenue est toujours insérée dans l'ensemble des solutions étalon. Si pendant un nombre donné d'itérations, la meilleure solution obtenue n'est pas améliorée, une solution aléatoire sera générée par le même procédé que la génération des solutions initiales. Cette solution aléatoire est incluse dans l'ensemble des solutions étalon.

Pour calculer l'estimation de la distribution de probabilité, nous employons un algorithme de distributions marginales univariantes (en anglais *univariate marginal distribution* (UMDA)) [MUH 98]. Cet algorithme préserve au mieux les caractères des solutions étalon. ED peut donc converger plus vite. Selon UMDA, la distribution est factorisée en un produit de *eta* distributions de probabilité :

$$P\ (Y) = \prod_{i=1}^{eta} P(y_i) \qquad \text{2. 10}$$

$P(y_i)$ est la probabilité marginale que l'opération i soit affectée à la station k où k est comprise entre 1 et le nombre total de stations m (voir l'équation 2. 11). Elle est définie par la distribution empirique de l'ensemble des solutions étalon :

$$P_l\ (y_i = k) = \frac{\sum \delta(y_i = k \mid D^{se}_{l-1})}{n} \qquad \text{2. 11}$$

où $\delta(y_i = k \mid D^{se}_{l-1}) = \begin{cases} 1 \text{ si } y_i = k \\ 0 \text{ sinon} \end{cases}$.

l est le nombre d'itérations.

eta est nombre de solutions étalon dans D^{se}_{l-1}

D^{se}_{l-1} est l'ensemble de solutions étalon choisies dans l'itération l-1.

La population l peut être générée selon la distribution de probabilité P_l.

Algorithme ED pour SALBP-2

L'algorithme ED commence par générer aléatoirement les solution initiales tout en respectant les contraintes de précédence. Ensuite, les solutions sont évaluées par l'équation (2. **9**). Des solutions avec des temps de cycle plus petits sont choisies pour construire un ensemble de solutions étalon. Ensuite la distribution de probabilité des variables est estimée par l'algorithme UMDA. Enfin, la nouvelle population est générée d'après la distribution obtenue. Ce procédé est répété tant que le nombre donné d'itérations n'est pas atteint.

Algorithme 2.8* : *ED pour SALBP-2

Etape 1 : Générer *K* solutions comme étant la population initiale pour la méthode proposée. Répéter les étapes 2-5 jusqu'à ce que le nombre d'itérations donné soit atteint.

Etape 2 : Evaluer les solutions dans la population courante en utilisant la fonction objectif (voir l'équation 2. **9**).

Etape 3 : Sélectionner un ensemble D_{l-1}^{se} de solutions de la population courante D_{l-1} comme des solutions étalon :

Les solutions dans D_{l-1}^{se} ont des temps de cycle plus petits que les autres ;
La meilleure solution obtenue est toujours insérée dans l'ensemble des solutions étalon D_{l-1}^{se} ;
Si la meilleure solution n'est pas améliorée pendant un nombre donné d'itérations, une solution est générée et insérée dans D_{l-1}^{se} ;

Etape 4 : Estimer la distribution de probabilité $P(Y) = P\left(Y \middle| D_{l-1}^{se}\right)$ à partir des solutions étalon.

Etape 5: Générer les individus de la nouvelle population D_l selon la distribution de probabilité obtenue.

Etape 6 : Tant que le nombre d'itération n'est pas atteint, retourne à l'étape 2.

2.3.3 SA pour le SALBP de type II

Pour comparer nos 2 précédentes méthodes, nous développons également le recuit simulé (SA) pour notre problème (voir Annexe SA)

2.3.3.1 Notre algorithme SA pour SALBP-2 :

La solution initiale

La solution initiale pour la méthode SA est générée aléatoirement.

Choix des paramètres

Dans le processus de SA, le choix de la température initiale T est important pour assurer une bonne performance de la méthode. Nous la définissons d'après les résultats de simulation. Ensuite, la température est réduite d'après l'équation 2. 12 suivante :

$$T=T*a \qquad 2.\ 12$$

où $a < 1$ est le taux de diminution de la température. Il est déterminé également par simulation.

Génération de solutions voisines

Quatre méthodes sont testées pour générer une solution voisine de la solution courante. Les méthodes sont:

M 1 : Choisir aléatoirement une opération dans une autre station et l'affecter à la station la moins chargée si les contraintes de précédence sont respectées.

M 2 : Choisir aléatoirement une opération dans la station la plus chargée et l'affecter à une autre station si les contraintes de précédence sont respectées.

M 3 : Choisir aléatoirement une opération dans la station la plus chargée et l'affecter à la station la moins chargée si les contraintes de précédence sont respectées.

M 4 : Choisir aléatoirement une station w_k et choisir aléatoirement une opération dans une autre station et affecter cette opération à la station w_k si les contraintes de précédence sont respectées.

Chacune de ces étapes est répétée si les contraintes de précédence ne sont pas respectées.

Un processus d'échange est aussi utilisé dans les quatre méthodes. Il choisit aléatoirement deux opérations effectuées par deux stations différentes et permute leur position si les contraintes de précédence sont respectées. De même que pour les quatre méthodes utilisées, cette étape est répétée si les contraintes de précédences ne sont pas respectées.

2.4 Résultats numériques

Nous présentons dans cette section les résultats numériques obtenus par les méthodes présentées dans ce chapitre. Les méthodes sont mises en œuvre à l'aide du langage C et avec un ordinateur dont le processeur est de 2Ghz et la carte mémoire de 512M.

2.4.1 Résultats numériques obtenus par l'heuristique proposée

Les problèmes que nous utilisons sont des problèmes de benchmark présentés sur le site : http://www.wiwi.uni-jena.de/Entscheidung/alb/salb2dat.htm. Dans les problèmes de benchmark présentés sur ce site Internet, le nombre d'opérations varie entre 8 et 297. Le nombre de stations est compris entre 3 et 52.

Les solutions trouvées par l'heuristique proposée sont comparées avec les résultats obtenus par l'heuristique bi-directionnelle [LIU 03]. Cette heuristique est à notre connaissance l'heuristique la plus récente de la littérature pour résoudre les SALBP-2 et elle est prouvée comme étant meilleure que les heuristiques plus anciennes (voir la section 1.3.3.2). Cette heuristique génère d'abord une solution initiale en utilisant une procédure de répartition bi-directionnelle basée sur l'heuristique de Hoffmann [HOF 63]. Ensuite la solution initiale est améliorée par la procédure de transfert et échange (voir l'algorithme 2.2). Les résultats obtenus pour les problèmes du benchmark sont donnés dans le Tableau 2. 4.

Problème	N_opération	N_station	Résultat*	Heuristique bi-directionnelle	Heuristique proposée
buxey	29	7	47	48	48
buxey		8	41	**41**	**41**
buxey		9	37	38	**37**
buxey		10	34	**34**	**34**
buxey		11	32	**32**	**32**
buxey		12	28	29	**28**
buxey		13	27	**27**	**27**
buxey		14	25	**25**	**25**
Lutz1	32	8	1860	**1860**	**1860**
Lutz1		9	1638	1664	1664
Lutz1		11	1400	**1400**	**1400**
Lutz1		12	1400	**1400**	**1400**
gunther	35	7	72	**72**	73
gunther		8	63	**63**	**63**
gunther		9	54	**54**	**54**
gunther		10	50	**50**	**50**
gunther		11	48	**48**	**48**
gunther		12	44	**44**	45
gunther		13	42	**42**	**42**
gunther		14	40	42	42
gunther		15	40	42	**40**
Wee-Mag	75	3	500	**500**	**500**
Wee-Mag		4	375	**375**	**375**
Wee-Mag		5	300	**300**	**300**
Wee-Mag		6	250	**250**	**250**
Wee-Mag		7	215	**215**	**215**
Wee-Mag		8	188	**188**	**188**
Wee-Mag		9	167	**167**	**167**
Wee-Mag		10	150	**150**	**151**
Wee-Mag		11	137	**137**	**137**
Wee-Mag		12	125	126	126
Wee-Mag		13	116	**116**	117
Wee-Mag		14	108	109	109
Wee-Mag		15	100	101	101
Wee-Mag		16	94	95	95
Wee-Mag		17	89	90	90
Wee-Mag		18	[84,87]	88	87
Wee-Mag		19	[84,85]	86	85
Wee-Mag		20	77	**77**	78
Wee-Mag		21	72	73	73
Wee-Mag		22	69	71	70
Wee-Mag		23	[66,67]	69	68

Problème	N_opération	N_station	Résultat*	Heuristique bi-directionnelle	Heuristique proposée
Wee-Mag	75	24	66	68	68
Wee-Mag		25	66	**66**	**66**
Wee-Mag		26	65	66	**65**
Wee-Mag		27	[63,65]	65	65
Wee-Mag		28	[63,64]	64	64
Wee-Mag		29	63	**63**	**63**
Wee-Mag		30	56	**56**	**56**
Mukherje	94	3	1403	**1403**	**1403**
Mukherje		4	1052	**1052**	**1052**
Mukherje		5	844	**844**	**844**
Mukherje		6	704	**704**	**704**
Mukherje		7	621	633	622
Mukherje		8	532	533	533
Mukherje		9	471	484	474
Mukherje		10	424	**424**	**424**
Mukherje		11	391	420	394
Mukherje		12	358	371	360
Mukherje		13	325	330	327
Mukherje		14	311	321	312
Mukherje		15	288	321	291
Mukherje		16	268	274	272
Mukherje		17	251	256	254
Mukherje		18	239	248	241
Mukherje		19	226	249	229
Mukherje		20	220	226	226
Mukherje		21	208	226	209
Mukherje		22	200	226	202
Mukherje		23	189	226	190
Mukherje		24	179	208	182
Mukherje		25	172	179	175
Mukherje		26	171	208	**171**
Barthol2	148	27	157	165	161
Barthol2		28	152	164	157
Barthol2		29	146	155	151
Barthol2		30	142	149	145
Barthol2		31	137	144	142
Barthol2		32	133	140	135
Barthol2		33	129	135	133
Barthol2		34	125	132	129
Barthol2		35	121	129	126
Barthol2		36	118	123	121
Barthol2		37	115	123	121
Barthol2		38	112	119	116
Barthol2		39	109	122	114
Barthol2		40	106	113	110

Problème	N_opération	N_station	Résultat*	Heuristique bi-directionnelle	Heuristique proposée
Barthol2	148	41	104	113	109
Barthol2		42	101	109	107
Barthol2		43	99	107	105
Barthol2		44	97	104	102
Barthol2		45	95	104	98
Barthol2		46	93	101	97
Barthol2		47	91	99	95
Barthol2		48	89	98	94
Barthol2		49	87	96	91
Barthol2		50	85	95	90
Barthol2		51	84	93	87

Tableau 2. 4 Résultats obtenus par l'heuristique proposée

N-opération : nombre d'opérations
N-station : nombre de stations
Résultat* : résultats optimaux ou bornes inférieure et supérieure présentés sur le site
Chiffres en caractère gras : résultats optimaux trouvés par l'heuristique correspondante

Sur les 99 problèmes traités dans le Tableau 2. 4, l'heuristique proposée permet de trouver les solutions optimales pour 36 problèmes. Alors que l'heuristique bi-directionnelle permet de trouver les solutions optimales pour 34 problèmes. L'approche que nous proposons apporte donc des résultats intéressants par rapport à l'heuristique bi-directionnelle. De plus, il n' y a que trois temps de cycle obtenus par l'heuristique bi-directionnelle qui sont plus petits que ceux obtenus par notre approche. Les temps de cycle obtenus par ces deux approches pour le reste des problèmes sont les mêmes. Nous voyons que les deux approches peuvent trouver de bonnes solutions pour des problèmes de tailles modestes, comme par exemple, les problèmes de buxey, Lutz1 etc. Plus le problème est complexe, c'est à dire plus le nombre de stations et le nombre d'opérations sont grands, plus la différence des deux solutions obtenues par ces deux méthodes est grande. L'heuristique que nous proposons ne donne pas des solutions trop éloignées des solutions optimales même pour les problèmes avec 148 opérations et 51 stations.

Les résultats obtenus dans le Tableau 2. 4 sont donnés pour des problèmes avec un nombre identique de solutions évaluées pour les deux heuristiques. Le temps de calcul est donné dans le Tableau 2. 5 suivant :

problème	N-opération	Mtcpu bi-dir	Mtcpu notre
buxey	29	0	0

Lutz1	32	0	0
gunther	35	3.7	3
Wee-Mag	75	56.8	56.3
Mukherje	94	103	103
Barthol2	148	193.7	190.4

Tableau 2. 5 Comparaison des temps de calcul

Mtcpu bi-dir : temps de calcul moyen de l'heuristique bi-directionnelle
Mtcpu notre : temps de calcul moyen de l'heuristique que nous proposons

Dans l'algorithme des deux heuristiques (bi-directionnelle et l'heuristique que nous proposons), la partie la plus consommatrice en temps CPU correspond à l'amélioration des solutions. Or, cette étape est identique pour les deux heuristiques. Par conséquent, la différence de temps de calcul pour les deux heuristiques n'est pas grande.

2.4.2 Résultats numériques obtenus par nos méta-heuristiques

Dans cette section, nous présentons les résultats numériques pour montrer les performances des méta-heuristiques que nous proposons pour résoudre les problèmes d'équilibrage de lignes d'assemblage dans le cas déterministe. Les exemples numériques proviennent du benchmark cité plus haut.

2.4.2.1 Résultats numériques pour les problèmes déterministes

Dans ce qui suit, ED et EM sont testées comparées avec la méthode SA (Tableau 2. 6).

Problème	ST	ED			EM			SA			Heuristique
		meilleure	moyen	Tcpu	meilleure	moyen	Tcpu	meilleure	moyen	Tcpu	
warnecke	6	260	261.7	0	260	260.1	0	259	260.3	0	260
tonge70	10	358	362.1	0	356	358.6	19	358	365.3	23	354
hahn	10	1775	1782.7	0	1775	1775	10	1792	1943.5	15	1792
Arc111	15	10468	10665	126	10232	10298.2	161	11934	14128	148	11139
Arc83	15	5331	5343.8	34	5217	5256.6	66	7332	8861.8	73	5148
heskia	5	206	206.8	0	205	206.2	0	205	206.6	0	206
lutz1	10	1592	1597.6	0	1534	1568.4	0	1592	1709.4	0	1526
lutz2	10	50	50	8	50	50	10	50	50.6	16	50
lutz3	10	171	171.5	19	170	171	22	170	175.2	57	171
barthol2	30	145	145.6	219	145	151	207	154	154.4	231	145

Tableau 2. 6 Problèmes déterministes résolus par méta-heuristiques

ST : nombre de stations

Dix classes de problèmes sont choisies pour tester les performances des méta-heuristiques. Le même nombre de solutions sont évaluées par les trois méta-heuristiques. ED converge plus vite que les deux autres méthodes. Cependant, les meilleures solutions de EM sont toujours plus proches de l'optimum que ceux des deux autres méthodes. La valeur moyenne du temps de cycle obtenu par EM est meilleure que celles fournies par ED et par SA sauf pour le problème barthol2.

Les résultats obtenus par l'heuristique proposée sont aussi donnés dans le Tableau 2. 6, nous voyons que pour les problèmes simples, c'est à dire que le nombre d'opérations et le nombre de contraintes de précédences ne sont pas très grandes, comme par exemple, warnecke et lutz1, l'heuristique proposée peut obtenir de bonnes solutions. Elle n'est pas efficace pour des problèmes complexes. En effet, les méta-heuristiques permettent de sortir facilement d'un optimum local et ce qui est plus difficile pour notre heuristique.

Remarque : Une borne inférieure serrée permet de diminuer le temps de calcul de EM et ED surtout pour les problèmes avec des temps opératoires très grands comme dans le problème de lutz1. Nous nous proposons donc de calculer dans le dernier chapitre une borne inférieure.

2.5 Conclusion

Dans ce chapitre nous nous intéressons à la minimisation du temps de cycle pour des problèmes d'équilibrage de lignes d'assemblage de type II, déterministe. Le problème déterministe est une simplification de la ligne de production réelle, sans considérer les facteurs de variation dans la durée des opérations. Pour le problème déterministe, le temps de cycle à optimiser est donc déterminé par le temps de station le plus grand.

Nous définissons tout d'abord une heuristique pour minimiser le temps de cycle de la ligne . L'avantage de cette heuristique est qu'elle peut générer la solution du problème SALBP-2 directement d'après les caractéristiques du problème. Cette heuristique est donc plus efficace que les heuristiques issues du SALBP-1 et appliquées au SALBP-2 avec différents essais sur les temps de cycle possibles. Mais cette heuristique ne permet pas de résoudre des problèmes complexes. Nous développons donc des méta-heuristiques pour le problème SALBP-2 comme l'algorithme basé sur le mécanisme d'électromagnétisme (EM) et l'estimation de distribution (ED). EM et ED sont des méta-heuristiques basées sur la génération d'une

population de solutions. Leur avantage commun est qu'elles n'ont pas beaucoup de paramètres à configurer. Elles sont donc faciles à implémenter et leurs performances sont stables. En comparant les deux méthodes, ED utilise un ensemble de vecteurs de nombres entiers au lieu de vecteurs de nombres réels pour EM. Par conséquent, l'espace de recherche de ED est plus petit que celui de EM. ED peut converger plus vite. Par contre, EM permet d'obtenir de meilleures solutions. Le recuit simulé est une méthode traditionnelle pour résoudre les problèmes d'optimisation combinatoire. Son avantage est qu'il utilise une représentation directe et simple. D'autre part, il permet de sortir de l'optimum local en acceptant de mauvaises solutions avec une probabilité qui diminue au fur et à mesure de l'avancement de la simulation. Dans ces travaux, nous ne développons pas de méthode propre à SA mais nous utilisons juste SA pour comparer nos résultats avec ceux obtenus par EM et ED. Nous remarquons alors que EM fournit de meilleurs résultats pour des problèmes déterministes.

Comme perspectives à nos travaux, il faudrait tout d'abord arriver à simplifier le mécanisme EM qui pour l'instant, même s'il est efficace, peut être à notre avis amélioré sur le processus de codage du problème SALBP-2 sous une forme utilisable par EM. Il serait également intéressant de tester d'autres types de pas. Pour la méta-heuristique ED, il faudrait pouvoir tester d'autres lois de distribution en fonction du problème considéré et des contraintes de ce problème. De plus, nous n'avons testé nos méta-heuristiques que sur un problème simple d'optimisation du temps de cycle de lignes d'assemblage. Il serait très intéressant de considérer d'autres facteurs comme la fiabilité de la ligne, les coûts de production, la qualité des produits, etc.

En effet, la fiabilité de la ligne est importante pour minimiser le coût de production des entreprises. Donc les travaux qui maximisent la fiabilité de la ligne et minimisent le temps de cycle simultanément sont très intéressants pour les entreprises. Dans la suite, nous proposons une approche basée sur EM pour résoudre ce type de problèmes dit multi-critère.

Chapitre 3.

Equilibrage de lignes d’assemblage stochastiques

Dans ce chapitre, nous considérons les problèmes d’équilibrage de lignes d’assemblage stochastiques dont les temps opératoires sont aléatoires. Dans un premier temps, nous cherchons à minimiser le temps de cycle tout en respectant une fiabilité donnée pour la ligne. Nous nous intéressons ensuite à l'optimisation simultanée de deux critères : le temps de cycle et la probabilité d'inachèvement des opérations dans le temps de cycle. Le but des problèmes d’optimisation multi-critère est de donner un ensemble de solutions optimales au lieu de déterminer une solution comme c’est le cas des problèmes avec un seul objectif. Nous proposons une méthode basée sur l’algorithme fondé sur le mécanisme d’électromagnétisme (EM).

3.1 Introduction

Dans ce chapitre, nous cherchons à résoudre les problèmes d'équilibrage de lignes d'assemblage de type II avec des temps opératoires stochastiques. Quand les opérations sont exécutées par des opérateurs humains, les temps opératoires sont rarement constants. En effet, beaucoup de facteurs peuvent influencer le travail des opérateurs, comme par exemple, la compétence, la température dans l'atelier, etc. Afin d'approcher le plus possible la réalité et fournir une solution utilisable par les entreprises, des modèles stochastiques ont été développés. Dans un modèle stochastique, le temps de cycle est lié à la probabilité que les opérations soient complètement réalisées sur une station dans le temps de cycle alloué. Un temps de cycle trop petit entraîne une probabilité élevée de dépassement ce qui n'est pas acceptable du fait de la réduction de la productivité de la ligne et de l'augmentation du coût de production. Un long temps de cycle réduit la probabilité de dépasser ce temps de cycle et augmente la fiabilité de la ligne. Cependant il réduit de manière significative la productivité et les gains. Il est donc intéressant de concevoir une ligne d'assemblage en tenant compte de l'aspect stochastique afin de définir un temps de cycle suffisamment petit mais avec une fiabilité acceptable([GU 06], [GU 07a]).

Lors de la conception ou ré-ingénierie d'une ligne d'assemblage, nous sommes souvent contraints d'optimiser plusieurs objectifs contradictoires à la fois, comme par exemple, maximiser la productivité, maximiser la fiabilité de la ligne, augmenter le taux d'utilisation des machines, minimiser les coûts de production, etc. (voir chapitre 1.2.3). Cependant la plupart des travaux de recherche existants n'optimisent qu'un seul critère ([BAY 86], [ARM 96] et [ARM 03]) et considèrent les autres objectifs comme des contraintes pour simplifier leur étude. Les problèmes qui nécessitent l'optimisation simultanée de plusieurs objectifs contradictoires sont des problèmes d'optimisation multi-objectif [OLG 04]. Ces problèmes d'optimisation multi-objectif consistent à déterminer un ensemble de solutions optimales pour lesquelles l'amélioration d'un objectif engendre une détérioration des autres.

Le problème d'équilibrage de lignes d'assemblage multi-objectif (ALBP multi-objectif) est un problème difficile à résoudre. A notre connaissance, peu de travaux existent dans la littérature. Nous pouvons citer les travaux de McMullen et Frazier qui utilisent une méthode basée sur le recuit simulé pour résoudre ce type de problèmes [MCM 98]. Ils travaillent sur des lignes

d’assemblage parallèles en modèle mixte (voir Section 1.2.1). Deux objectifs, le coût total de production et le temps de cycle, sont considérés dans leurs travaux. Le recuit simulé est utilisé dans un premier temps pour optimiser un des deux objectifs. Leurs résultats expérimentaux prouvent que le recuit simulé permet de déterminer de bonnes solutions au niveau du temps de cycle et peut aussi déterminer des solutions moyennes au niveau du coût de production par rapport à d’autres heuristiques. Ensuite, une fonction objectif qui combine les deux objectifs avec des poids identiques est définie pour évaluer les solutions obtenues. Cependant, dans cet article, le recuit simulé ne donne pas un ensemble de solutions optimales, donc pas de choix multiples pour les décideurs sur le compromis entre objectifs. Pour déterminer des solutions optimales, McMullen propose ensuite une autre méthode basée sur la colonie de fourmis pour résoudre des problèmes multi-objectif [MCM 06]. Dans cet article, quatre objectifs sont considérés : le nombre d’ouvriers, l’utilisation des machines, la probabilité que les opérations soient réalisées sur une station dans un temps de cycle donné et le coût de conception du système. Afin d’évaluer les solutions obtenues, il a proposé une fonction objectif efficace qui intègre les quatre objectifs en même temps. Mais la manière de construire la fonction objectif ne peut pas être généralisée, parce que cette fonction objectif est construite selon les caractéristiques de chaque objectif. Dans la littérature, la méthode la plus populaire pour les problèmes d’optimisation multi-objectif est l'algorithme génétique [CAR 99]. En 2000, Ponnambalam et *al.* ont utilisé des algorithmes génétiques pour résoudre des problèmes multi-objectif d’équilibrage de lignes d’assemblage correspondants à l’optimisation du nombre de stations, le lissage de la charge des stations et l’efficacité de la ligne [PON 00]. Ils ont mis en place plusieurs modifications de sélection des parents pour générer la population suivante par croisement et mutation. La méthode est prouvée comme étant efficace par simulation. En revanche, le choix des paramètres de la méthode (le taux de mutation, nombre des parents, etc.) est difficile. Les apports de cet article sont difficilement généralisables pour d’autres problèmes.

Dans les travaux de recherche existants, les méthodes fondées sur l'évolution de populations ont montré leur efficacité pour déterminer un ensemble de solutions optimales. Elles peuvent améliorer continuellement l’ensemble des solutions optimales par la génération de solutions non dominées dans chaque itération [DRA 02]. L’algorithme basé sur le mécanisme de l’électromagnétisme (EM) a également cet avantage. De plus, il est simple à utiliser parce qu’il ne nécessite pas beaucoup de paramètres à définir. Dans ce chapitre, nous proposons une

méthode basée sur EM pour résoudre des problèmes multi-critère d'équilibrage de lignes d'assemblage.

Dans un premier temps, nous présentons d'abord les problèmes ALBP stochastiques. Ensuite, nous proposons une approche basée sur EM pour résoudre des ALBP multi-critère. Les deux critères que nous cherchons à optimiser ainsi que le concept du problème d'optimisation multi-critère sont également explicités. Nous présentons ensuite une nouvelle méta-heuristique EM pour les ALBP multi-critère stochastiques. Des applications numériques sont proposées pour montrer l'efficacité de notre méthode. Enfin, les conclusions et perspectives sont présentées.

3.2 Equilibrage de lignes stochastiques : cas mono-critère

Dans cette partie, nous nous intéressons au problème stochastique de l'équilibrage de lignes d'assemblage (ALBP stochastique). Les temps opératoires sont maintenant aléatoires [ROB 89]. En effet, l'efficacité des ouvriers varie dans le temps. De plus, les pannes des machines peuvent influencer grandement la productivité de la ligne. Dans le cas stochastique, la ligne ne peut pas garantir que tous les produits soient fabriqués dans un temps de cycle donné. Soit la ligne est stoppée de façon à achever leur production, soit des produits incomplets sont mis en stock. Ces deux possibilités augmentent largement le coût de production. Le temps de cycle de la ligne doit donc être minimisé sous la contrainte de sa fiabilité, i.e. la probabilité que certaines stations n'aient pas terminé leurs opérations au bout du temps de cycle alloué. Nous développons une méthode basée sur le principe EM (voir section 2.3.1.1) pour résoudre ce problème stochastique.

Dans ce qui suit, nous présentons le problème ALPB stochastique, puis le processus basé sur EM pour résoudre ce problème.

3.2.1 ALBP stochastique

Dans cette partie, nous étudions le ALBP stochastique suivant : minimisation du temps de cycle de la ligne tout en respectant la fiabilité donnée. Les temps opératoires T_i sont aléatoires et suivent chacun une loi normale. Soit t_i la moyenne du temps opératoire de l'opération i et

soit σ^2_i sa variance. L'objectif est de minimiser le temps de cycle *c* de la ligne tout en assurant une fiabilité minimale α dont la définition précise est donnée ultérieurement.
Les temps des stations, déterminés par le temps que chaque produit passe sur une station, sont des variables aléatoires. Le temps de la station *k* est désigné par ST_k . Il est calculé par l'équation suivante :

$$ST_k = \sum_{i \in W_k} T_k \qquad 3.1$$

où W_k est l'ensemble des opérations affectées à la station *k*.

Sous l'hypothèse des temps opératoires mutuellement indépendants et suivant une loi normale, chaque temps de station ST_k est une variable aléatoire distribuée selon une loi normale de moyenne $\sum_{i \in Wk} t_i$ et de variance $\sum_{i \in Wk} \sigma i^2$.

Nous définissons d'abord la fiabilité d'une station comme la probabilité P_k que la station termine l'ensemble de ses opérations dans le temps de cycle *c* donné. Comme les temps de stations suivent une loi normale :

$$P_k = \Pr ob(ST_k \leq c) = \Phi((c - \sum_{i \in Wk} t_i) + \sqrt{\sum_{i \in Wk} \sigma_i^2}) \qquad 3.2$$

où $\Phi(z)$ est la fonction de distribution d'une loi normale standard.

La fiabilité d'une ligne peut alors être définie par la probabilité qu'un produit peut progresser sur chaque station dans un temps de cycle *c*. En supposant que les temps opératoires sont indépendants, la fiabilité d'une ligne peut donc être déterminée par :

$$FL = \prod_{k=1}^{m} P_k \qquad 3.3$$

La fiabilité d'une ligne d'assemblage mesure la capacité d'une ligne à synchroniser ses opérations selon un temps de cycle donné. Une fiabilité élevée est intéressante car elle est plus facile à gérer et à synchroniser avec les fournisseurs et les clients. Une fiabilité élevée permet au système de fournir les produits aux clients avec un délai de livraison fiable. Mais la fiabilité ne doit pas compromettre le taux de productivité. Dans nos travaux, nous ne considérons pas la résolution sur les produits avec au moins une opération qui n'est pas finie

sur la ligne. En réalité, soit ces produits sont jetés ou fabriqués sur une station hors la ligne. Soit le temps de cycle est prolongé pour finir toutes les opérations sur ces produits.

3.2.2 EM pour les ALBP stochastiques

Notre méta-heuristique EM pour résoudre le ALBP stochastique commence par un ensemble de solutions initiales choisies aléatoirement. Ces solutions sont évaluées par la méthode présentée dans l'algorithme 3.1. Dès que les temps de cycle des solutions sont déterminés, les charges et les forces totales des solutions sont calculées par les équations 2. 5 et 2. 6. Ensuite une nouvelle population est générée par déplacement des solutions courantes selon la direction de la force totale. Ce processus est répété jusqu'à ce que le nombre donné d'itérations soit atteint. Nous détaillons dans ce qui suit la méthodologie basée sur EM.

Représentation du problème

Nous employons la représentation par *clés aléatoires* (voir Algorithme 2.6, Chapitre 2), pour introduire les solutions du ALBP stochastique dans EM.

Evaluation des solutions

Il s'agit de déterminer le temps de cycle d'une ligne d'assemblage tel que la fiabilité du temps de cycle soit supérieure ou égale à une valeur donnée α.

Le théorème suivant assure qu'il n'existe qu'un seul temps de cycle *c* le plus petit possible tel que la fiabilité *FL* de la ligne est au moins α.

Théorème 3.1 : Pour une affectation donnée des opérations aux stations, la fiabilité *R* de la ligne d'assemblage est une fonction croissante du temps de cycle *c*.

Preuve : la conclusion est évidente selon les équations (3. 2) et (3. 3) qui présentent la définition de *FL*.

QED.

L'algorithme suivant permet de déterminer le temps de cycle par dichotomie :

Algorithme 3.1 : Evaluation des solutions du ALBP stochastique

Etape 1 : Définir deux réels *a* et *b*, tels que $a = 0$ et $b = max(\sum_{i \in Wk} t_i)$.

Etape 2 : Calculer la fiabilité *FL* à l'aide des équations (3. 2) et (3. 3) avec comme temps de cycle $c = b$.

Etape 3 : Si la fiabilité *FL* est plus petite que la valeur donnée α, c'est à dire que le temps de cycle n'est pas suffisamment grand, alors $a \leftarrow b$, $b \leftarrow 2b$ et on retourne à l'étape 2.

Etape 4 : Calculer la fiabilité *FL* avec le temps de cycle $c = (a + b) / 2$.

Etape 5: Si la fiabilité obtenue *FL* est plus petite que α, définir $a \leftarrow (a + b)/2$, sinon, définir $b \leftarrow (a + b)/2$.

Etape 6: Répéter les étapes 4 et 5 jusqu'à ce que l'intervalle [a, b] soit suffisamment petit.

3.3 Equilibrage de lignes sous plusieurs critères

Dans le cas où deux ou plusieurs critères contradictoires sont optimisés, le problème est un problème d'optimisation multi-critère. Dans ce cas, généralement, plus un critère est bon, plus les autres critères sont mauvais. L'optimisation multi-critère cherche à déterminer l'ensemble des solutions qui ne sont pas dominées par d'autres solutions sur l'ensemble des critères. L'ensemble des solutions ainsi obtenu est appelé le front Pareto et les solutions sont dites des optimums Pareto.

Nous nous limitons à l'étude des deux critères suivants : la minimisation du temps de cycle et la maximisation de la fiabilité de la ligne ou de manière équivalente la minimisation de la probabilité qu'au moins une machine ne peut pas terminer ses opérations dans le temps de cycle. Nous présentons ces deux critères dans ce qui suit.

3.3.1 Problèmes multi-critère

3.3.1.1 Critères à optimiser

Nous considérons deux critères contradictoires : le temps de cycle et la probabilité qu'un produit ne puisse pas progresser sur une station dans un temps de cycle donné (la probabilité d'inachèvement des opérations). Le temps de cycle est un laps de temps alloué à chaque station de la ligne d'assemblage pour terminer toutes ses opérations. Il est un des objectifs les plus considérés pour les SALBP (voir Chapitre 1). En effet, un grand temps de cycle entraîne

une faible productivité et donc une mauvaise satisfaction des clients. La probabilité d'inachèvement des opérations est liée aux coûts de production. Dans le cas où les opérations ne peuvent pas être complétées dans une station dans le temps de cycle alloué, trois solutions sont possibles : la première solution consiste à abandonner la fabrication du produit correspondant et reprendre ultérieurement sa fabrication, la deuxième solution consiste à affecter les opérations qui ne sont pas terminées sur des postes de travail à l'extérieur de la ligne, la troisième solution consiste à rallonger le temps de cycle pour terminer le produit. Ces trois solutions entraînent un coût de production plus important. Par conséquent, une grande probabilité d'inachèvement des opérations peut donc augmenter les coûts de production, comme par exemple le coût lié aux ressources, le coût de stockage des produits etc.

Les deux critères à optimiser sont contradictoires parce que pour une affectation donnée des opérations sur les stations, plus le temps de cycle est grand, plus la probabilité d'inachèvement des opérations est petite et vice-versa.

Dans les modèles stochastiques, le temps de cycle et la probabilité d'inachèvement des opérations peuvent être calculés en fonction des lois des temps opératoires. Dans la littérature, plusieurs lois aléatoires sont proposées. La loi exponentielle est souvent utilisée pour représenter les pannes des machines sur une ligne automatique [SMU 85]. La loi normale, qui est la plus utilisée dans la littérature, permet de modéliser des lignes manuelles d'assemblage ([MCM 06], [JUN 97] et [PON 06]).

- ***Temps de cycle***

Le temps de cycle des lignes d'assemblage est une durée de temps qui représente le temps entre deux produits successifs fabriqués sur la ligne. Pour les différents types de problèmes d'équilibrage de lignes d'assemblage, le temps de cycle est déterminé de façon différente. Pour les problèmes SALBP déterministes, le temps de cycle est déterminé par le temps de station le plus grand (voir Chapitre 2). Dans le cas où les temps opératoires sont aléatoires, les opérations pour assembler un produit ne peuvent pas être obligatoirement réalisées pendant le temps de cycle. Le temps de cycle de la ligne est donc lié à une probabilité que les opérations soient complétées sur une station. Pour un problème stochastique qui consiste à minimiser le temps de cycle, le temps de cycle est déterminé par la valeur la plus petite qui peut assurer que chaque station termine ses opérations avec une probabilité donnée (voir Section 3.2).

Plusieurs temps de cycle sont estimés pour une affectation des opérations aux stations. La méthode que nous employons pour générer les temps de cycle est présentée dans ce qui suit.

Pour chaque affectation des opérations, un nombre donné l de temps de cycles sont estimés. Le plus grand temps de cycle estimé est déterminé par :

$$c_g = \sum_{i \in Wk} t_i + 4\sigma_k$$

où σ_k^2 est la variance du temps de station pour la station *k*. Le temps de station moyen pour la station *k* est plus grand que ceux des autres stations.

Le plus petit temps de cycle estimé est déterminé par :

$$c_p = \mathrm{Max}(\frac{1}{m}\sum_{i=1}^{n} t_i \, , t_{\max})$$

où *n* est le nombre d'opérations, *m* est le nombre de stations et t_{max} est le plus grand temps opératoire.

Les autres temps de cycle estimés sauf les deux valeurs extrêmes présentées ci-dessus se situent entre l'intervalle c_g et c_p. La probabilité d'inachèvement des opérations correspondant à chaque temps de cycle est ensuite déterminée.

- ***Probabilité d'inachèvement des opérations***

Pour étudier des modèles stochastiques, la probabilité d'inachèvement des opérations par rapport à leur affectation aux stations est un objectif intéressant, parce qu'une petite probabilité d'inachèvement des opérations entraîne un faible surcoût de production. Dans nos travaux, les temps opératoires suivent chacun une loi normale. La probabilité d'inachèvement des opérations aux stations selon un temps de cycle donné *c*, dénotée par *IP*, peut donc être déterminée par l'équation suivante :

$$IP = 1 - \prod_{k=1}^{m} P_k \qquad 3.4$$

où p_k est la probabilité que les opérations affectées à la station *k* peuvent être complétées ou réalisées pendant le temps de cycle. La probabilité p_k est estimée par l'équation (3. 2).

3.3.1.2 Fonction objectif

Pour les problèmes d'optimisation multi-critère, l'évaluation des solutions est un point difficile.

Une méthode pour évaluer les solutions de ce type de problèmes est de combiner linéairement les critères dans une seule fonction objectif. La fonction objectif d'un exemple d'un problème multi-critère qui optimise *mc* critères, peut être présentée par l'équation suivante :

$$f(x) = w_1 f_1(x) + \ldots + w_i f_i(x) + \ldots + w_{mc} f_{mc}(x) \qquad 3.5$$

Le poids des critère de la fonction *f(x)* peut être constant [MCM 98], aléatoire [WAN] ou obtenu par une fonction donnée [MUR 96]. Dans cette fonction, le critère avec la valeur la plus importante va être prépondérant, c'est à dire qu'il sera mieux optimisé que les autres. Pour contourner ce problème, Yoo et *al*. proposent une approche d'adaptation des poids (en anglais adaptive weight approach -AW).

L'approche AW utilise des informations intéressantes de la population courante pour réguler les poids et obtenir une pression vers un point idéal. La fonction objectif basée sur AW peut être calculée par l'équation suivante :

$$F(x) = \frac{f_1(x) - f_1(x_{1best})}{f_1(x_{1worst}) - f_1(x_{1best})} + \frac{f_2(x) - f_2(x_{2best})}{f_2(x_{2worst}) - f_2(x_{2best})} + \cdots + \frac{f_{mc}(x) - f_{mc}(x_{nbest})}{f_{mc}(x_{nzorst}) - f_{mc}(x_{nbest})} \qquad 3.6$$

où *mc* est le nombre de critères à optimiser,

$f_i(x)$ est la valeur du *i*ème critère correspondante à la solution *x*,

$f_i(x_{ibest})$ est la valeur optimale du *i*ème critère,

$f_i(x_{iworst})$ est la valeur la plus mauvaise du *i*ème critère.

Dans nos travaux, nous utilisons l'approche AW pour construire la fonction objectif. Selon l'approche AW, nous employons la fonction objectif suivante pour évaluer les solutions obtenues par EM :

$$F(x) = \frac{c(x) - c(\underline{x})}{c(x_{worst}) - c(\underline{x})} + IP(x) - IP(x_{best}) \qquad 3.7$$

où $c(x)$ est un temps de cycle effectué pour la solution *x*,

$\underline{x}$ est la borne inférieure du temps de cycle,

$c(x_{worst})$ est le plus grand temps de cycle effectué,

$IP(x_{best})$ est la probabilité d'inachèvement des opérations la plus petite obtenue.

3.3.1.3 Solutions Pareto optimales

Dans ce qui suit, nous présentons la définition des solutions Pareto optimales [SAB 00].
Soit un problème d'optimisation multi-objectif avec *v* variables de décision *x* (paramètres) et *obj* objectifs *y*. Les objectifs sont :

$$\text{Min } Y=f(x)=(f_1(x_1,\ldots,x_v),\ldots, f_{obj}(x_1,\ldots,x_v)) \qquad 3.8$$

où $x=(x_1,\ldots,x_v)\in \tilde{X}$ et $y=(y_1,\ldots,y_{obj}) \in \tilde{Y}$,

$\tilde{X}$ est l'espace de solutions,

$\tilde{Y}$ est l'espace des ensembles des objectives.

Définition 3.1 : Une solutions $a\in \tilde{X}$ est dite dominant la solution $b\in \tilde{X}$ (qui peut être représentée mathématiquement par $a\succ b$), si et seulement si :

$$\forall i \in \{1,\cdots,obj\}: \ f_i(a) \le f_i(b) \wedge \exists\, j \in \{1,\cdots,obj\}: f_j(a) < f_j(b)$$

Autrement dit *a* couvre *b* ($a \underline{\succ} b$) si et seulement si $a\succ b$ ou $f(a)=f(b)$.

Nous pouvons alors donner la définition des solutions Pareto optimales basées sur la définition 3.1 :

(a) la solution *a* est dite non-dominée en ce qui concerne un ensemble $X'\in \tilde{X}$ si et seulement s'il n'y a pas de solution *a'* qui domine la solution *a* :

$$a'\in X' : a'\succ a$$

(b) la solution *a* est Pareto optimale si et seulement si *a* est non-dominée dans l'espace de solutions $\tilde{X}$.

3.3.2 Résolution du problème multi-critère par EM

EM permet de déplacer une population de points vers l'optimum. Comme toute méthode basée sur l'évolution de populations, les solutions Pareto optimales peuvent être actualisées par les populations de chaque itération.

3.3.2.1 Algorithme

Dans la suite, nous donnons la méta-heuristique EM pour les ALBP multi-critère :

Algorithme 3.2 : EM pour ALBP multi-objectif

Etape 1 : Initialiser les données.
nombre d'opérations,
contraintes de précédence,
temps d'opération moyens,
nombre de stations.

Etape 2 : Générer l'ensemble des vecteurs de clés aléatoires,
transformer les vecteurs de clés aléatoires en un ensemble de solutions,
évaluer ces solutions initiales.

Etape 3 : Déterminer les solutions non-dominées dans la population courante.

Etape 4 : Mettre à jour l'ensemble des solutions Pareto optimales.

Etape 5 : Sélectionner les individus élitistes selon la stratégie élitiste.

Etape 6 : Calculer la force totale de chaque solution et déplacer les solutions courantes pour générer la nouvelle population.

Etape 7 : Retourner à l'étape 3, si le nombre donné d'itérations n'est pas atteint.

3.3.2.2 Représentation des solutions

La représentation des solutions combine la définition de clés aléatoires [BEA 94] et un heuristique défini par Kilbridge et Wester pour les SALBP-1 [KIL 61a]. La représentation ainsi que la transformation des clés aléatoires en solution SALBP-2 sont les même que celles du chapitre 2 pour l'équilibrage déterministe. Plus précisément, nous utilisons les temps opératoires moyens dans la transformation et les variances des temps opératoires ne sont pas prises en compte dans le codage des solutions.

3.3.2.3 Définition du mécanisme d'attraction répulsion pour le cas multi-critère

En général, la performance d'une solution donnée par EM est complètement déterminée par la fonction objectif. Si la valeur de la fonction objectif de la solution *i* est meilleure que celle de la solution *j*, alors la particule électrique chargée correspondant à la solution *i* attire le point chargé *j*, sinon le point chargé *i* repousse le point chargé *j*. Le mécanisme déplace les points échantillonnés vers l'optimum selon la fonction objectif. Par conséquent, cette méthode peut donner la meilleure solution pour un problème d'optimisation avec un seul objectif.

Pour un problème d'optimisation multi-critère, l'objectif est de construire un ensemble de solutions Pareto optimales. A partir des différences des critères que nous considérons, nous modifions le mécanisme d'attraction-répulsion pour améliorer la performance de EM et l'adapter à des problèmes multi-critère. Ainsi, il est raisonnable de noter que si la solution *i* domine la solution *j*, la solution *i* a alors la priorité. Sinon au cas où les deux solutions ne sont pas dominées, la solution avec la plus petite valeur de la fonction objectif donnée par l'équation (3.8) a la priorité. Entre deux solutions, la solution prioritaire attire l'autre et en même temps la solution la moins prioritaire repousse l'autre. Le nouveau mécanisme que nous proposons est basé sur cette idée. La force totale d'une solution x_i est donc donnée par :

$$F(x_i) = \sum_{i=1, i\neq j}^{n} \begin{cases} (x_j - x_i)\dfrac{q_i q_j}{\left\| x_j - x_i \right\|^2} \text{ si } x_j \succ x_i \\ (x_i - x_j)\dfrac{q_i q_j}{\left\| x_j - x_i \right\|^2} \text{ si } x_i \succ x_j \\ (x_j - x_i)\dfrac{q_i q_j}{\left\| x_j - x_i \right\|^2} \text{ si il n'y a pas de relation de domination entre } x_j \text{ et } x_i \text{ et } f(x_j) \leq f(x_i) \\ (x_i - x_j)\dfrac{q_i q_j}{\left\| x_j - x_i \right\|^2} \text{ si il n'y a pas de relation de domination entre } x_j \text{ et } x_i \text{ et } f(x_i) \leq f(x_j) \end{cases}$$

où q_i est la charge correspondante à la solution *i* et *f(x)* est la fonction objectif définie dans la section 3.3.1.2.

3.3.2.4 Stratégie élitiste

Nous utilisons aussi une stratégie élitiste pour assurer que certaines caractéristiques des meilleures solutions soient préservées dans le processus d'évolution des populations [MUR 96]. Tout d'abord, une solution qui peut optimiser au moins un des objectifs peut être considérée comme un individu élitiste. Par conséquent, nous avons au plus *obj* individus élitistes pour un problème avec *obj* critères. Cette stratégie préserve la meilleure solution correspondante à chaque critère de la prochaine population. Elle peut assurer que les caractéristiques qui améliorent les solutions obtenues ne soient pas perdues. Au cours de l'exécution de notre procédure EM, nous mémorisons un ensemble de solutions Pareto optimales et nous mettons à jour cet ensemble de solutions dans chaque itération. Certaines solutions Pareto optimales sont choisies aléatoirement et préservées comme individus élitistes.

Dans nos travaux, nous retenons 5 solutions comme des individus élitistes pour chaque population : 2 solutions correspondent aux objectifs à optimiser, les trois autres sont choisies aléatoirement dans l'ensemble de solutions Pareto optimales. Ces individus élitistes sont choisis pour préserver le bon caractère dans la prochaine population.

3.4 Résultats numériques

3.4.1 Résultats numériques pour les problèmes stochastiques mono critère

Dans cette section, nous présentons les résultats numériques obtenus par l'approche basée sur EM pour les problèmes ALBP stochastiques mono-critère. Les résultats de EM sont comparés avec ceux de SA (voir Annexe 1).

Nopération	Nstation	Fiabilité	CV	EM		SA	
				MTC	Tcalcule	MTC	Tcalcule
11	3	0,9	0,1	17,15	0	18,21	0
		0,925		17,26	0	18,38	0
		0,975		17,66	0	19,1	0
		0,9	0,2	18,61	0	19,6	0
		0,925		18,83	0	20,08	0
		0,975		19,83	0	21,12	0
		0,9	0,5	23,3	0	24,31	0

		0,925		24	0	25,09	0
		0,975		25,82	0	27,22	0
29	12	0,9	0,1	33,84	0	34,06	0
		0,925		34,15	0	34,48	0
		0,975		35,23	0	35,87	0
		0,9	0,2	38,19	0	37,99	0
		0,925		38,57	0	39,01	0
		0,975		40,46	0	41,37	0
		0,9	0,5	52,08	0	55,67	0
		0,925		53,06	0	54,95	0
		0,975		57,17	0	57,95	0
35	5	0,9	0,1	107,39	7	108,03	12
		0,925		107,98	7	108,84	12
		0,975		109,68	7	110,77	12
		0,9	0,2	116,83	7	117,95	12
		0,925		118,4	7	118,56	12
		0,975		122,37	7	122,43	12
		0,9	0,5	146,21	7	147,2	12
		0,925		149,03	9	150,12	12
		0,975		157,53	8	159,04	12
89	8	0,9	0,1	66,02	64	66,63	101
		0,925		66,2	62	66,86	100
		0,975		66,95	64	67,96	102
		0,9	0,2	70,3	64	71,2	102
		0,925		70,78	64	71,59	102
		0,975		72,2	68	73,18	102
		0,9	0,5	83,76	64	84,79	102
		0,925		84,93	64	85,87	102
		0,975		88,29	68	89,25	102

Tableau 3. 1 Comparaison des résultats donnés par EM et SA pour des ALBP stochastiques

CV : coefficients communs de variance des temps opératoires
Nopération : nombre d'opérations
Nstation : nombre de stations
MTC : moyen de temps de cycle obtenu
Tcalcule : temps de calcul

Quatre types d'instances (c'est à dire le nombre d'opérations et les contraintes associées) et 30 problèmes sont définis pour évaluer la performance de EM pour ce type de problèmes. Les temps opératoires sont distribués selon une loi normale. Les coefficients de variation des temps opératoires sont notés par CV. Par conséquent, les écart-types des temps opératoires sont $\sigma_i = CV*t_i$**,** les temps opératoires sont générés aléatoirement par $T_i=t_i+ \sigma_i*\omega$ où

$\varpi = \sqrt{-2\ln\beta_1} * \cos(2\pi\beta_2)$. β_1 et β_2 sont deux variables indépendantes et distribuées selon une loi uniforme entre 0 et 1.

Les résultats numériques sont donnés dans le Tableau 3. 1. Nous voyons que les deux méthodes permettent d'obtenir de bonnes solutions dans un temps de calcul acceptable. Par exemple, pour un problème avec 89 opérations, les solutions données dans le Tableau 3. 1 sont obtenues avec un temps CPU égal à 102 unité CPU. Nous pouvons remarquer que EM converge plus rapidement que SA. De plus, EM permet de trouver de meilleures solutions que SA. Plus la fiabilité donnée d'une ligne est grande, plus la différence des temps de cycle obtenus pour chaque méthode est grande. Ainsi, plus les coefficients de variation des temps d'opération sont grands, plus la différence de performance est grande.

3.4.2 Résultats numériques pour les problèmes multi-critère

Dans cette section, nous donnons des résultats numériques de notre méta-heuristique EM pour les problèmes ALBP multi-critère. Nous rappelons que l'objectif est de déterminer l'ensemble des solutions Pareto optimales.

D'abord, nous montrons un ensemble des solutions Pareto optimales pour un système de 35 opérations et 5 stations. Les contraintes de précédence entre les opérations sont identiques aux contraintes du problème de benchmark gunther de SALBP-2 [GUN 83]. Les temps opératoires moyens sont égaux aux temps opératoires des problèmes de Gunther. Les variances sont égales à 1. Les solutions Pareto optimales obtenues sont présentées dans la Fig. 3. 1.

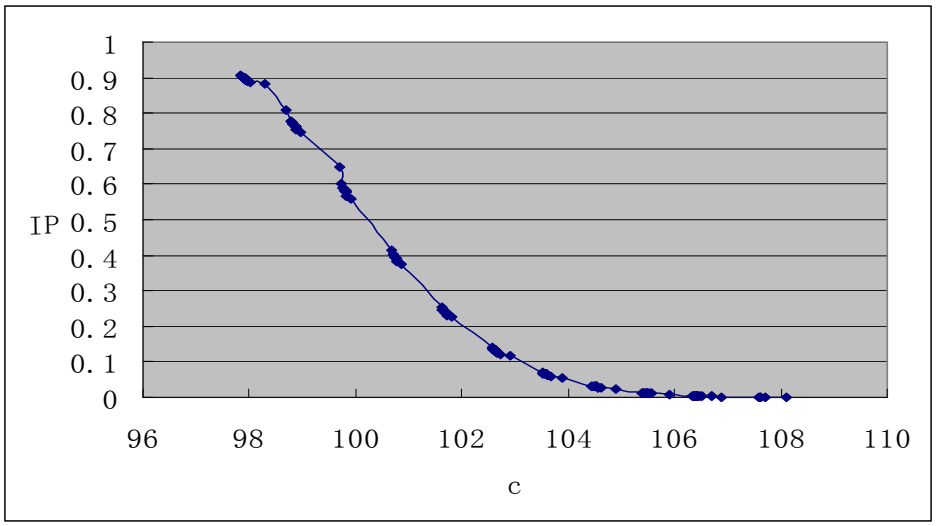


Fig. 3. 1 Ensemble des solutions Pareto optimales

Dans l'ensemble des solutions Pareto optimales obtenues, 135 solutions Pareto optimales sont données. Le plus petit temps de cycle est égale à 97.85 avec une probabilité d'inachèvement des opérations égale à 0.9. La plus petite probabilité d'inachèvement des opérations est nulle avec un temps de cycle correspondant égal à 108.1.

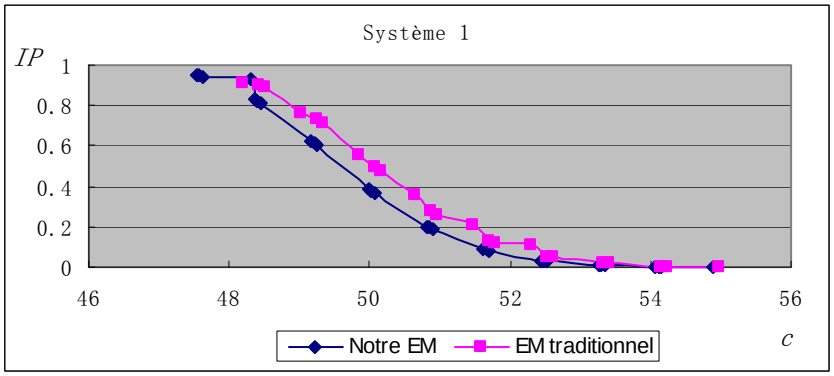
Pour examiner la performance du nouveau mécanisme d'attraction répulsion, nous comparons aussi les résultats numériques de la méthode EM proposée avec ceux d'un processus EM « traditionnel », c'est à dire sans modification du calcul de la force totale. EM « traditionnel » utilise aussi une stratégie élitiste pour améliorer l'ensemble des solutions Pareto optimales. Dans le mécanisme d'attraction répulsion de la méthode EM « traditionnel », pour deux solutions, la solution correspondante à la valeur la plus petite de la fonction objectif attire l'autre et en même temps la solution correspondante à la valeur la plus grande repousse l'autre.

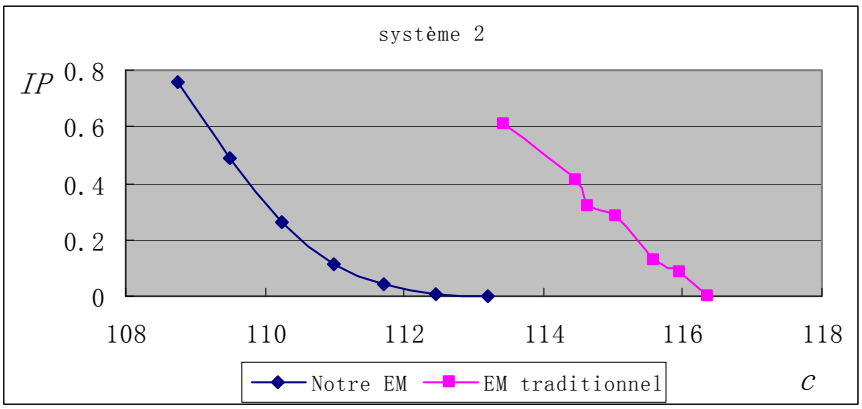
Instances	Nopérations	Nstations	Référence
Système 1	29	7	Buxey
Système 2	28	10	Heskia
Système 3	30	6	Sawyer
Système 4	70	11	Tonge
Système 5	45	8	Kilbrid
Système 6	58	10	Warnecke

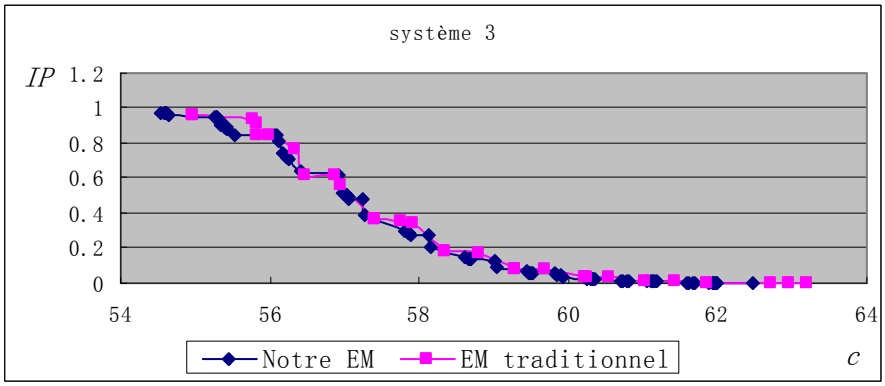
Tableau 3. 2 Systèmes étudiés

Nopération : nombre d'opérations
Nstation : nombre de stations

Les systèmes étudiés sont présentés dans le Tableau 3. 2, les contraintes de précédence des systèmes étudiés sont identiques à celles des problèmes benchmark de SALBP-2 présentés dans la colonne Référence. Les temps opératoires moyens des systèmes étudiés sont égaux aux temps opératoires de ces problèmes. Les variances d'opération sont égales à 1. Dans les figures suivantes, nous présentons les solutions Pareto optimales obtenues par les deux méthodes. Les solutions obtenues par notre EM sont présentées par des losanges et les solutions de EM « traditionnel » sont présentées par des carrés.







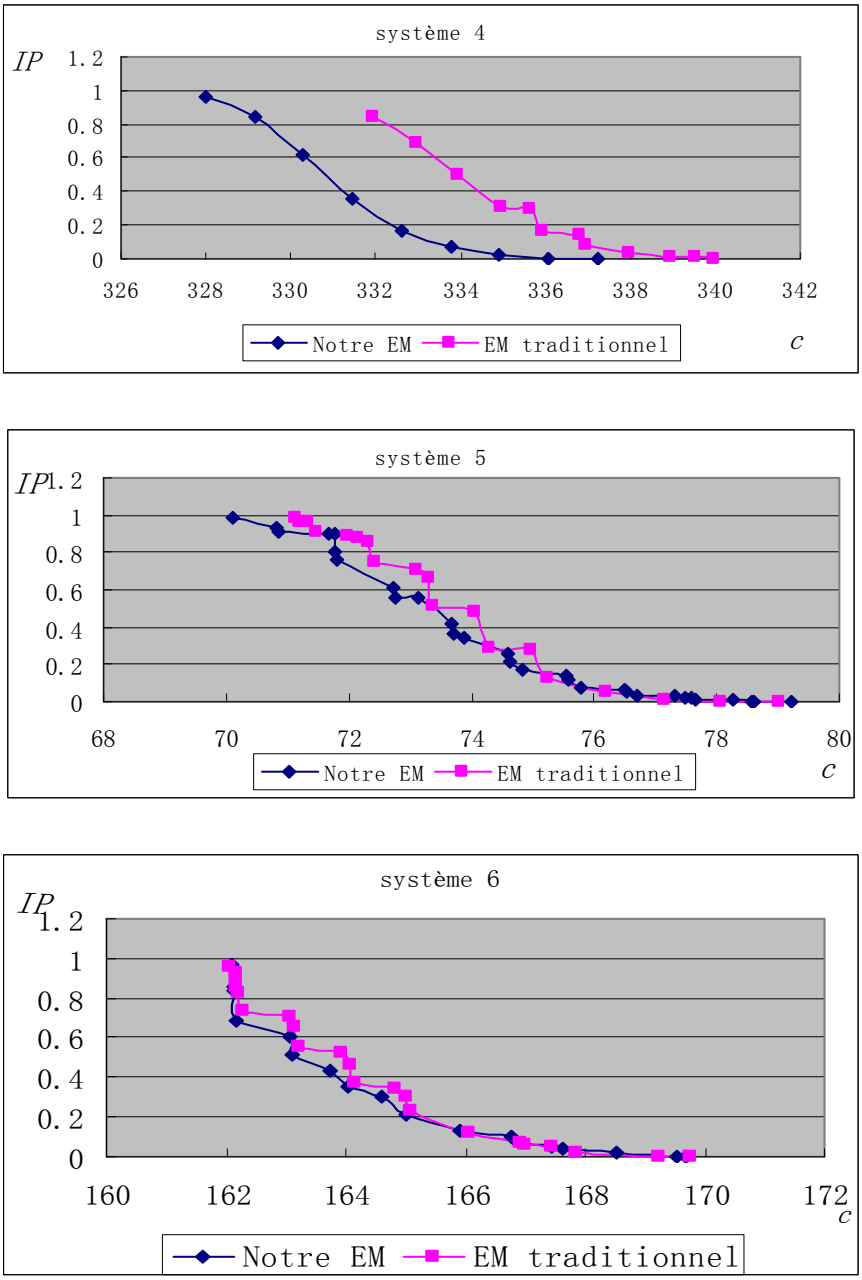


Fig. 3. 2 Solutions Pareto optimales des deux méthodes basées sur EM

Nous comparons les résultats obtenus par les deux méthodes dans le Tableau 3. 3. Nous voyons que plupart des solutions obtenues par EM « traditionnel » sont dominées par celles de notre EM. Plus le problème est complexe, c'est à dire que plus il y a de contraintes de précédence comme par exemple le système 2 ou lorsque le nombre d'opération est grand comme par exemple le système 4, plus la différence de performance des deux méthodes est

grande. En général, la méthode EM proposée permet de déterminer plus de solutions Pareto optimales qui dominent la plupart des solutions du EM « traditionnel ».

Instances	Notre EM			EM traditionnel		
	Npareto	Ndominé	Temps CPU	Npareto	Ndominé	Temps CPU
Système 1	29	0	106	23	22	103
Système 2	7	0	117	7	7	112
Système 3	86	4	101	30	16	101
Système 4	9	0	787	12	12	764
Système 5	22	5	178	18	10	174
Système 6	30	8	184	19	11	183

Tableau 3. 3 Comparaison des résultats obtenus

Npareto : nombre de solutions Pareto optimales obtenues
Ndominé : nombre de solutions obtenues par une méthode qui sont dominés par celles de l'autre méthode.
Temps CPU : temps de calcul

3.5 Conclusion

Dans ce chapitre, nous proposons d'utiliser la méthode EM pour optimiser le temps de cycle du problème d'équilibrage de lignes d'assemblage stochastiques en respectant la fiabilité de la ligne d'assemblage considérée. Les temps d'opération suivent une loi normale. Nous proposons ensuite une nouvelle méthode basée sur l'approche EM pour résoudre des problèmes multi-critères d'équilibrage de lignes d'assemblage. Nous travaillons sur deux aspects :

1. Construire une fonction objectif afin de guider la recherche vers l'ensemble des solutions Pareto optimales.
2. Maintenir la diversité de la population afin de garder les bonnes caractéristiques pour assurer la convergence des solutions Pareto optimales.

Pour le premier aspect, la fonction objectif est basée sur une approche AW. Cette approche utilise des informations de la population courante pour corriger les poids et obtenir la direction de recherche vers les points ideaux. Nous préservons ainsi les meilleurs individus

dans la population pour produire la nouvelle génération et améliorer la qualité des solutions avec de bonnes caractéristiques. Le deuxième objectif est résolu par la construction d'un nouveau mécanisme d'attraction répulsion. Ce nouveau mécanisme d'attraction - répulsion repose sur l'idée de solutions non-dominées qui attirent les autres solutions. Ce mécanisme est approprié pour déplacer les solutions choisies vers l'ensemble des solutions Pareto optimales. Les résultats numériques montrent la bonne performance de la méthode proposée.

Cependant nous n'avons testé notre approche qu'avec deux critères contradictoires. Il serait très intéressant de considérer d'autres critères comme par exemple le lissage de la charge des machines, l'efficacité de la ligne et comparer notre méthode avec d'autres méthodes de génération de populations.

De plus, pour améliorer encore plus la vitesse de convergence de l'algorithme que nous proposons, il serait intéressant de calculer une bonne solution initiale. Par conséquent, dans le chapitre suivant nous proposons une méthode basée sur la relaxation lagrangienne et la génération de colonnes pour améliorer l'estimation de la borne inférieure du temps de cycle.

Chapitre 4.

Calcul de borne inférieure par la méthode de génération de colonnes

Dans ce chapitre, nous présentons une méthode basée sur la génération de colonnes pour calculer une borne inférieure du temps de cycle afin d'améliorer les performances de la méthode de résolution du problème d'équilibrage de lignes d'assemblage de type II. D'abord, nous modélisons le problème étudié sous forme d'un programme en nombres entiers et nous présentons brièvement la méthode de génération de colonnes. Nous proposons ensuite une formulation orientée colonnes du problème ainsi que l'algorithme que nous avons développé pour le calcul de la borne inférieure du temps de cycle.

4.1 Introduction

Bien que la majorité des méthodes exactes permettent de trouver la meilleure solution pour un problème de SALBP, elles sont peu utilisées dans la littérature car le temps de calcul est souvent très grand. Le nombre d'opérations est souvent inférieur à cent dans les problèmes d'équilibrage de la ligne d'assemblage traités par méthode exacte ([AGH 95], [ROB 96] et [DES 02]). D'autre part, des contraintes supplémentaires (voir Chapitre 1.2.2) sont difficiles à ajouter dans la formulation du problème. Les problèmes, élargis avec des contraintes supplémentaires, sont difficiles voir impossibles à résoudre par méthode exacte. Pour ces raisons plutôt que de déterminer la solution optimale par une méthode exacte nous nous limitons à calculer une borne inférieure serrée du temps de cycle. Cette borne inférieure pourra ensuite être utilisée dans des heuristiques ou des méta-heuristiques.
Ainsi, nous proposons une méthode utilisant la génération de colonnes pour reformuler le programme linéaire et ainsi déterminer une borne inférieure du temps de cycle pour le SALBP-2. L'idée centrale de la génération de colonnes est que les programmes linéaires de grande taille ont trop de variables. Alors qu'à l'optimum, la plupart des variables sont hors base et, très souvent, la plupart d'entre elles sont nulles. C'est-à-dire que seulement un (petit) sous-ensemble de variables doit être pris en compte pour résoudre le problème. La génération de colonnes détermine les variables qui peuvent améliorer la qualité de la solution obtenue à travers l'optimisation d'un sous-problème appelé problème de génération de colonnes (en anglais *pricing problem).* Nous proposons une méthode basée sur la programmation dynamique pour résoudre les problème de génération de colonnes.

Le reste du chapitre est organisé de la manière suivante. D'abord, nous présentons les formulations du problème SALBP-2 existantes dans la littérature et nous expliquons le principe de la méthode de génération de colonnes. Ensuite, nous présentons notre méthode de génération de colonnes pour calculer une borne inférieure du temps de cycle pour un problème SALBP-2. Enfin, nous appliquons notre méthode à plusieurs lignes d'assemblage et concluons sur les avantages et limites de notre démarche.

4.2 Génération de colonnes

L'objectif de cette section est d'introduire le principe de la génération de colonnes. Ainsi,

nous formulons tout d'abord notre problème en programmation linéaire en nombre entiers, avant de détailler les étapes de la génération de colonnes.

4.2.1 Formulation de la programmation linéaire en nombres entiers (modélisation mathématique du problème)

Plusieurs formulations du problème d'équilibrage de lignes d'assemblage sont données dans la littérature ([BAY 86], [AME 06], [GOK 98] et [PAT 75]). Généralement, un problème SALBP-2 est représenté par un programme linéaire en nombres entiers. Un état de l'art sur les formulations du problème SALBP-2 est présenté dans la section 1.3.1.

4.2.2 Principe de la méthode de génération de colonnes

La génération de colonnes est une variante spécialisée de la méthode de Simplexe pour résoudre efficacement des programmes linéaires de grande taille. Elle repose sur la décomposition de Dantzig-Wolfe [DAN 60] qui consiste à fractionner le problème original en un ensemble de problèmes de génération de colonnes et un problème maître. Le problème maître coordonne les problèmes de génération de colonnes et assure que les contraintes soient respectées. Une solution du problème original peut être obtenue par la résolution du problème maître correspondant. Le problème maître a un grand nombre de variables, beaucoup plus que le problème original. Les variables sont définies implicitement par un ensemble des bases des solutions faisables et selon les directions données par les problèmes de génération de colonnes. Au lieu d'utiliser toutes les variables du problème maître, la méthode de génération de colonnes exécute toujours des opérations avec un petit sous-ensemble de variables. Le problème maître restreint est optimisé avec un ensemble courant de solutions actives. La génération de colonnes ajoute dans le problème maître restreint les variables qui sont susceptibles d'entrer dans la base, c'est à dire qui peuvent optimiser le coût de la solution obtenue. Ce processus continue jusqu'à ce qu'il n'y ait aucune variable à ajouter. Comme dans le processus de la méthode Simplexe classique, cela signifie qu'une solution optimale est trouvée. En général, la solution optimale sera trouvée sans utiliser explicitement toutes les variables du problème maître.

Pour identifier les variables qui doivent entrer dans la base courante du problème maître restreint nous travaillons de la manière suivante. D'abord nous définissons une fonction

objectif de chaque problème de génération de colonnes. Elle est construite sur les valeurs duales du problème maître restreint. Pour un problème de maximisation, en optimisant le problème de génération de colonnes on détermine les variables qui ont les coûts les plus réduits. Si la valeur de la fonction objectif pour la solution optimale est positive, alors la variable avec le coût maximal réduit est ajoutée au problème maître. Sinon, aucune variable du problème maître ne peut plus améliorer la solution. Dans ce cas, la solution courante est la solution optimale du problème original.

4.2.2.1 Problème maître restreint

Problème maître

La décomposition de Dantzig-Wolfe est utilisée pour résoudre le problème maître, qui n'est qu'une re-formulation du problème original. Le problème maître est obtenu à partir du problème original par remplacement des contraintes linéaires représentant la région faisable pour une contrainte donnée par la combinaison convexe des points et des rayons extrémaux. Par exemple un problème original est présenté par :

$\text{Max}\{cx : Ax \rhd\lhd b, x \in R^n_{\lozenge}\}$

où $\rhd\lhd$ présente trois relaxations possibles : >, <,=.

La matrice *A* des coefficients a une structure de bloc primal diagonal. *P* est la région faisable du problème original (un polyèdre).

$$P = \left\{ \begin{array}{l} x_0 \in R^{n_1}_{\lozenge_0} \\ x_1 \in R^{n_1}_{\lozenge_1} \\ \quad\vdots \\ x_K \in R^{n_K}_{\lozenge_K} \end{array} : \begin{pmatrix} A_0 & A_1 & \cdots & A_K \\ & B_1 & & \\ & & \ddots & \\ & & & B_K \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_K \end{pmatrix} \begin{matrix} \rhd\lhd_0 \\ \rhd\lhd_1 \\ \vdots \\ \rhd\lhd_K \end{matrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_K \end{pmatrix} \right.$$

le problème maître de ce problème original peut être présenté par :

$$
\begin{array}{lrcl}
\text{Max } c_0 x_0 + \sum_{i_1} \left(\bar{c}_1\right)_{i_1} \left(\lambda_1\right)_{i_1} + \cdots + \sum_{i_K} \left(\bar{c}_K\right)_{i_K} \left(\lambda_K\right)_{i_K} & & & \\
s.c. \quad A_0 x_0 + \sum_{i_1} \left(\bar{A}_1\right)_{i_1} \left(\lambda_1\right)_{i_1} + \cdots + \sum_{i_K} \left(\bar{A}_K\right)_{i_K} \left(\lambda_K\right)_{i_K} & \triangleright\triangleleft_0 & b_0 & \\
\sum_{i_1} \left(\delta_1\right)_{i_1} \left(\lambda_1\right)_{i_1} & = & 1 & \\
\ddots & \vdots & \vdots & \\
\sum_{i_K} \left(\delta_K\right)_{i_K} \left(\lambda_K\right)_{i_K} & = & 1 & \\
x_0 \in R^{n_0}_{\Diamond_0} : \lambda_1 \in R_+^{|E_1|}, \ldots, \lambda_K \in R_+^{|E_K|} & & &
\end{array}
\qquad 4.\ 1
$$

où $\left(\bar{c}_k\right)_{ik} := c_k x_k^{ik}$ et $\left(\bar{A}_k\right)_k := A_k x_k^{ik}$ avec x_k^{ik} est un point extrême ou un rayon extrême du polyèdre P $R_+^{|E_k|}$ est un sous-ensemble de R_+ (voir Annexe 2).

La forme la plus simple est présentée par :

$$
\left\{
\begin{array}{l}
\text{Max} \quad c_0 x_0 + \bar{c}\lambda \\
s.c. \quad A_0 x_0 + \bar{A}\lambda \triangleright\triangleleft_0 b_0 \\
\Delta\lambda = 1 \\
x_0 \in R^{n_0}_{\Diamond_0}; \quad \lambda \in R_+^{|E_!|+\ldots+|E_K|}
\end{array}
\right\}
\qquad 4.\ 2
$$

où $\Delta := diag(\delta_1, ..., \delta_K)$.

La présentation du principe de la re-formulation du problème original est donnée dans l'Annexe 2

Le problème maître restreint

Le problème maître restreint est un artifice technique spécifique à la méthode de génération des colonnes. Il est défini pour les sous-ensembles E'_k des ensembles E_k des variables du problème maître . Les définitions suivantes sont données pour construire le problème maître restreint.

Soit $E'_k \subseteq E_k$, il faut alors définir les variables $\lambda', \bar{c}', \bar{A}'$ et Δ' correspondants aux $\lambda, \bar{c}, \bar{A}$ et Δ du problème maître de Dantzig-Wolf (voir l'équation 4. 2) limité dans E'_k. Ensuite, le problème maître restreint peut être présenté par :

$$\left\{\begin{array}{c} \text{Max} \quad c_0 x_0 + \bar{c}'\lambda' \\ s.c. \quad A_0 x_0 + \bar{A}'\lambda' \triangleright\triangleleft_0 b_0 \\ \Delta'\lambda' = 1 \\ x_0 \in R^{n_0}_{\Diamond_0}; \quad \lambda' \in R^{|E'_1|+\ldots+|E'_K|}_{+} \end{array}\right\} \qquad 4.3$$

Un problème maître restreint est simplement un problème maître avec un grand nombre de variables enlevées. Les variables enlevées peuvent être considérées nulles, et on les empêche d'entrer dans la base. Le problème maître est limité. Toutefois, le problème maître restreint évolue d'une itération à l'autre car l'ensemble des variables E'_k est modifié par les itérations du processus de la génération de colonnes. Dans chaque itération, de nouvelles variables sont ajoutées dans le problème maître restreint. La génération de colonnes relaxe les contraintes sur les nouvelles variables en interdisant de les faire entrer dans la base et en fixant leurs valeurs à 0. Les variables qui quittent la base peuvent être supprimées du problème maître restreint et leur valeur est fixée à 0. Le problème maître restreint est renouvelé à chaque itération.

Le problème maître restreint a un nombre de variables plus petit que le problème maître. Pour les problèmes de grande taille, son problème maître restreint peut être résolu par ordinateur alors que son problème maître ne peut pas. En effet, les processus d'optimisation de la programmation linéaire s'exécutent beaucoup plus vite sur de petits problèmes. D'autre part, il est plus facile de générer les coefficients des contraintes et de l'objectif des nouvelles variables que générer les coefficients de toutes les variables dans l'initialisation.

4.2.2.2 Problème de génération de colonnes

Les *K* problèmes de génération de colonnes sont définis dans le processus de la génération de colonnes pour déterminer les variables intéressantes à intégrer dans le problème maître restreint. Les solutions du problème de génération de colonnes *k* correspondent au vecteur des variables λ_k du problème maître. Parce que la région faisable du problème de génération de colonnes *k* est définie comme P_k, l'ensemble des solutions faisables dans la base et des rayons extrêmes du problème de génération de colonnes *k* sont l'ensemble des points extrêmes Π_k et des rayons extrêmes R_k de P_k. Par la résolution du problème de génération de colonnes avec une fonction objectif donnée, nous pouvons identifier un élément $x_k = x_k^{i_k}, i_k \in E_k$ qui

représente un point extrême $i_k \in \Pi_k$ ou un rayon extrême $i_k \in R_k$. Pour évaluer les variables du problème maître, la valeur de la fonction objectif du problème de génération de colonnes doit refléter le coût réduit des variables correspondantes $(\lambda_k)_{ik}, i_k \in E_k$ du problème maître dans une base particulière.

Le coût réduit $(d_k)_{ik}$ d'une variable $(\lambda_k)_{ik}$ peut être obtenu par référence au problème maître (voir l'équation 4. 3). D'abord, (π, μ) est défini comme les valeurs duales du problème maître. Elles peuvent être déterminées par le problème maître restreint [TEB 01]. Alors le coût réduit $(d_k)_{ik}$ est déterminé par l'équation suivante :

$$\begin{aligned}(d_k)_{i_k} &= (\overline{c_k})_{i_k} - [\pi(\overline{A_k})_{i_k} + \mu(\delta_k)_{i_k}] \\ &= (c_k - \pi A_k)x_k^{i_1} - u\end{aligned} \qquad 4.\ 4$$

La fonction objectif du problème de génération de colonnes est défini comme :

$$Z_k = (c_k - \pi A_k)x_k - \mu \qquad 4.\ 5$$

Elle dépend des valeurs duales (π, μ) du problème maître dans une base particulière. Le problème de génération de colonnes *k* peut être formulé par :

$$Z_k^* = \left\{ \begin{array}{c} \text{Max} \quad (c_k - \pi A_k)x_k - \mu \\ s.\text{c.}\ B_k x_k \rhd\lhd_k b_k \\ x_k \in R_{\Diamond_k}^{n_k} \end{array} \right\} \qquad 4.\ 6$$

Quand les variables duales (π, μ) sont données, nous pouvons déterminer les variables $(\lambda_k)_{ik}$ du problème maître avec le coût réduit le plus grand par rapport à la solution optimale Z^*_k du problème de génération de colonnes. Si la valeur optimale du problème de génération de colonnes est $Z^*_k > 0$, les variables correspondantes vont entrer dans le problème maître

restreint. Sinon le processus de génération de colonnes est arrêté parce qu'il n'y a plus de variables qui doivent entrer dans la base et qui peuvent améliorer la solution obtenue dans l'itération précédente.

Un algorithme classique de la génération de colonnes est présenté dans l'Annexe 2.

La génération de colonnes analyse un problème compliqué en résolvant itérativement des problèmes beaucoup plus simples (les problème maître restreint et les problèmes de génération de colonnes). Dans ce qui suit, nous allons utiliser la génération de colonnes pour déterminer une borne inférieure de la valeur optimale du problème SALBP-2 formulé en programmation linéaire en nombres entiers.

4.3 Méthode de génération de colonnes appliquée à notre problème

Par la suite, nous présentons la formulation du problème SALBP-2. Ensuite, nous montrons l'application de la méthode proposée pour ce type de problèmes

4.3.1 Formulation du problème SALBP-2

Le problème SALBP-2 est formalisé comme un problème de programmation linéaire en nombre entiers [ARM 99]. Les variables de décision x_{ik} sont binaires. Si l'opération i est affectée à la station k, la variable x_{ik} est égale à 1, sinon elle est égale à 0. L'objectif est de minimiser le temps de cycle c de la ligne, présenté par l'équation (4.7).

$$\text{Min } c \qquad (4.7)$$

Les contraintes sont données par :

- Chaque opération doit être assignée à une seule station :

$$\sum_{k \in SI_j} x_{ik} = 1 \quad \text{pour } i = 1,\ldots,n \qquad (4.8)$$

où n est le nombre d'opérations,

SI_i est l'ensemble des stations entre E_i et L_i,
L_i est la station au plus tard pour l'opération i,
E_i est la station au plus tôt pour l'opération i.

- Le temps de cycle de la ligne, noté *c*, est déterminé par le temps de station le plus grand :

$$c \geq \sum_{i \in S_k} t_i * x_{ik} \quad \text{pour} \quad k=1,\ldots, m \qquad 4.9$$

où m est le nombre de stations,
t_i est la durée de l'opération i.

- Les contraintes de précédences :

$$\sum_{k \in SI_h} k * x_{hk} \leq \sum_{k \in SI_i} k * x_{ik} \quad \text{pour } h \in P_i^* \text{ et } L_h \geq E_i \qquad 4.10$$

où P_i^* est l'ensemble des prédécesseurs (directs ou indirects) de l'opération i.

- Les contraintes qui stipulent le caractère binaire des variables de décision :

$$x_{ik} \in \{0,1\} \quad \text{pour } i = 1,\ldots,n \qquad 4.11$$

4.3.2 Méthode proposée

La borne inférieure du problème formulé par les équations (4.7) – (4.11) est déterminée dans cette section par une méthode de génération de colonnes. Par la suite, nous détaillons cette méthode.

4.3.2.1 Une approche par génération de colonnes

Au lieu d'appliquer directement la technique de génération de colonnes au problème précédent, nous donnons une formulation orientée colonnes permettant d'obtenir une bonne borne inférieure du temps de cycle *c*. La formulation orientée colonnes est la suivante :

Chaque colonne (j, k) relative à la station k est définie par les paramètres suivants :

$$q_{ij}^k = \begin{cases} 1, & \text{si l'opération } i \text{ est dans la colonne } (j, k) \\ 0, & \text{sinon} \end{cases}$$

$$c_{jk} = \sum_{i=1}^{n} t_i \, q_{ij}^{k}$$

Soit:

Ω_k: ensemble des colonnes de la station k composées des opérations $i \in S_k$ et $c_{jk} \leq UC$ où UC est une borne supérieure du temps de cycle minimal;

Ω: ensemble des colonnes.

Nous cherchons alors à résoudre le problème suivant :

$$\text{Min} \sum_{(j,k)\in\Omega} f(c_{jk}) Z_{jk} \qquad \text{4. 12}$$

sous les contraintes suivantes :

$$\sum_{(j,k)\in\Omega} k q_{hj}^{k} Z_{jk} - \sum_{(j,k)\in\Omega} k q_{ij}^{k} Z_{jk} \geq 0, \forall (h,i) \in A \qquad \text{4. 13}$$

$$\sum_{(j,k)\in\Omega} q_{ij}^{k} Z_{jk} = 1, \forall i = 1, \ldots, n \qquad \text{4. 14}$$

$$\sum_{(j,k)\in\Omega_k} Z_{jk} \leq 1, \forall k = 1, \ldots, m \qquad \text{4. 15}$$

$$Z_{jk} \in \{0,1\}, \forall (j,k) \in \Omega \qquad \text{4. 16}$$

où A est l'ensemble des arcs du graphe de précédence, $f(x) = x^N$. Cette fonction objectif permet de déterminer un ensemble de colonnes (j, k) avec un C_{jk} plus petit que les autres tout en respectant les contraintes. N est une valeur entière et constante que nous fixons lors de la simulation.

Dans cette formulation, la contrainte (4.13) est la contrainte de précédence, la contrainte (4. **14**14) garantit que chaque opération est affectée à une station, la contrainte (4.15) assure qu'au plus m colonnes sont utilisées et la contrainte (4.16) est la contrainte d'intégrité.

Au lieu de résoudre directement ce programme en nombres entiers, nous le simplifions à l'aide de la relaxation linéaire. Ce problème relaxé est considéré comme un problème maître donné par :

$$\text{Min} \sum_{(j,k)\in\Omega} f(c_{jk}) Z_{jk} \qquad \text{4. 17}$$

sous les contraintes suivantes :

$$\sum_{(j,k)\in\Omega} \left(kq_{hj}^{k} - kq_{ij}^{k} \right) Z_{jk} \geq 0, \forall (h,i) \in \text{A} \qquad \text{4. 18}$$

$$\sum_{(j,k)\in\Omega} q_{ij}^{k} Z_{jk} = 1, \forall i = 1,...,n \qquad \text{4. 19}$$

$$\sum_{(j,k)\in\Omega_k} Z_{jk} \leq 1, \forall k = 1,...,m \qquad \text{4. 20}$$

$$Z_{jk} \geq 0, \forall (j,k) \in \Omega \qquad \text{4. 21}$$

Le problème maître est résolu par la génération de colonnes dans nos travaux. Cependant, dans un premier temps, nous allons établir la relation entre le problème maître et le problème d'équilibrage des lignes d'assemblage. Nous prouvons qu'une borne inférieure du temps de cycle c du problème SALBP-2 peut être déterminée par optimisation du problème maître.

Théorème 4.1: Soit c^* le maximum des temps de cycle c_{jk} tel que $Z_{jk} > 0$. Alors il existe un entier positif N_0 tel que, pour tout $N \geq N_0$, c^* est le temps de cycle minimal tel que le système linéaire 4.18-4.21 a une solution avec seulement des colonnes (j, k) telles que $c_{jk} \leq c^*$.

Remarque 4.1: Si nous remplaçons 4.12 par $MIN \sum_{(j,k)\in\Omega} Z_{jk}$ et nous nous limitons aux colonnes (j, k) avec (j, k) telles que $c_{jk} \leq c$, nous obtenons alors la formulation orientée colonnes pour le problème SALBP-1 avec un temps de cycle c donné. c^* est alors équivalent au minimum du temps de cycle c tel que la borne inférieure de la génération de colonnes du SALBP-1 est supérieure ou égale à m.

Remarque 4.2 : Il est montré dans la littérature que la génération de colonnes donne de bornes inférieures très serrées pour SALBP-1. Nous en déduisons que *c** est une bonne borne du temps de cycle minimal du SALBP-2.

Remarque 4.3: Une autre formulation orientée colonnes du problème SALBP-2 est la suivante :

Min *c*
sous les contraintes suivantes :

$$\sum_{(j,k)\in\Omega} kq_{hj}^{k} Z_{jk} - \sum_{(j,k)\in\Omega} kq_{ij}^{k} Z_{jk} \geq 0, \forall (h,i) \in \mathrm{A}$$

$$\sum_{(j,k)\in\Omega} q_{ij}^{k} Z_{jk} = 1, \forall i = 1,...,n$$

$$c \geq \sum_{(j,k)\in\Omega k} c_{jk} Z_{jk}, \forall k = 1,\ldots,m$$

$$Z_{jk} \in \{0,1\}, \forall (j,k) \in \Omega$$

Malheureusement, la relaxation linéaire de cette formulation donne une très mauvaise borne inférieure égale à $\sum_{i=1}^{n} t_i \Big/ m$ qui est même moins bonne que *LC1* (voir Section 1.3.3.1).

Pour comprendre cela, il suffit de considérer une séquence quelconque d'opérations respectant les contraintes de précédence. Nous affectons les opérations dans l'ordre donné à différentes stations avec un temps de cycle $c^* = \sum_{i=1}^{n} t_i \Big/ m$. Lorsque le placement d'une opération dépasse *c**, nous la découpons en deux et poursuivons le placement sur la station suivante. Soit a_{ik} le pourcentage de l'opération *i* placé dans la station *k*. Il est clair que $\sum_{i=1}^{n} t_i \, a_{ik} = c^*$.

Pour chaque station *k*, considérons les colonnes suivantes. Soit S_k = {[1], [2], ...} l'ensemble des opérations partiellement ou entièrement placées dans la station *k* classées dans l'ordre $a_{[1]k} \leq a_{[2]k} \leq \ldots$.

Cas I: la station k ne contient pas d'opération fractionnée, alors nous définissons une seule colonne (1, k) avec les opérations dans S_k et $Z_{1k} = 1$.

Cas II: la station k contient une opération [1] fractionnée, alors nous définissons deux colonnes : colonne (1,k) avec toutes les opérations dans S_k et $Z_{1k} = a_{[1]k}$ et colonne (2,k) avec toutes les opérations dans $S_k - \{[1]\}$ et $Z_{1k} = 1- a_{[1]k}$.

Cas III : la station k contient deux opérations [1] et [2] fractionnées, alors nous définissons trois colonnes : colonne (1,k) avec toutes les opérations dans S_k et $Z_{1k} = a_{[1]k}$, colonne (2, k) avec toutes les opérations dans $S_k - \{[1]\}$ et $Z_{2k} = a_{[2]k} - a_{[1]k}$ et colonne (3, k) avec toutes les opérations dans $S_k - \{[1], [2]\}$ et $Z_{3k} = 1- a_{[2]k}$.

Selon la construction,

$$\sum_{(j,k)\in\Omega k} c_{jk} Z_{jk} = \sum_{i=1}^{n} t_i a_{ik} = \frac{1}{m}\sum_{i=1}^{n} t_i$$

pour tout k, ce qui conclut la preuve.

Preuve du Théorème 4.1 : Puisque le nombre de stations et le nombre d'opérations sont finis, Ω est un ensemble fini. Alors, 4.18-4.21 forment un polyèdre fini. Il existe donc a> 0 tel que pour tout sommet Z du polyèdre, $Z_{jk} > 0$ implique $Z_{jk} \geq a$.

Pour chaque sommet Z, considérons

$$u(Z) = \max_{(j,k)\in\Omega \text{ et } Z_{jk}>0} c_{jk}$$

Soit Z* le sommet tel que $u(Z)$ est minimal. Considérer un sommet quelconque $Z°$ tel que $u(Z°) > u(Z*)$. Puisque le polyèdre est fini, le nombre de sommets l'est également. Par conséquent, il existe $b > 0$ tel que $u(Z°) = u(Z*) + b$.

Comparons les valeurs de critère des deux sommets:

$$f(Z^*) \leq mu^N(Z^*)$$
$$f(Z^\circ) \geq au^N(Z^\circ) = a^*(u(Z^*)+b)^N$$

ce qui implique $f(Z°) \geq f(Z*)$ lorsque

$$N \geq \frac{\ln(m/a)}{\ln(1+b/u(Z^*))}$$

ce qui conclut la preuve.

Q.E.D.

Résolution du problème maître par la génération de colonnes

Le cœur de la génération de colonnes est la résolution du problème de génération de colonnes qui consiste à déterminer la colonne avec le coût réduit le plus faible. Les problèmes de génération de colonnes correspondants au problème maître peuvent être formulés par :

$$\underset{k,q_{ij}^k}{\text{Min}} \; f(c_{jk}) - \sum_{(h,i)\in A} \lambda_{hi}(kq_{hj}^k - kq_{ij}^k) - \sum_{i=1}^{n} \mu_i q_{ij}^k - \gamma_k \qquad 4.\,22$$

ce qui se décompose en *m* sous-problèmes de génération de colonnes, chacun relatif à une station. Le problème de génération de colonnes de la station *k* est donné par :

$$R_k = \underset{q_{ij}^k}{\text{Min}} \; f(c_{jk}) - \sum_{i=1}^{n} p_{ik} q_{ij}^k - \gamma_k \qquad 4.\,23$$

avec

$$p_{ik} = \mu_i + \sum_{(i,h)\in A} k\lambda_{ih} - \sum_{(h,i)\in A} k\lambda_{hi} \quad \text{et} \quad c_{jk} = \sum_{i=1}^{n} t_i \, q_{ij}^k$$

sous les contraintes suivantes :

$$q_{ij}^k = 0, \forall i \notin S_k \qquad 4.\,24$$

$$\sum_{i=1}^{n} t_i q_{ij}^k \leq UC \qquad 4.\,25$$

Pour faciliter la résolution du problème de génération de colonnes, nous reformulons le problème de génération de colonnes de la station *k* par l'équation suivante :

$$R_k = \underset{C=0,1,\ldots,UC}{\text{Min}} f(C) - g_n(C) - \gamma_k \qquad \text{4.26a}$$

où $g_l(C)$ avec l = 1, 2, ..., n est le coût optimal du problème de sac-à-dos suivant :

$$g_l(C) = \underset{q_{ij}^k}{\text{Max}} \sum_{i=1}^{l} p_{ik} q_{ij}^k \qquad \text{4. 26b}$$

sous les contraintes suivantes :

$$q_{ij}^k = 0, \forall i \notin S_k \qquad \text{4. 27}$$

$$\sum_{i=1}^{n} t_i q_{ij}^k \leq UC \qquad \text{4. 28}$$

Le problème de génération de colonnes peut être optimisé en testant un ensemble de valeurs C et en déterminant la valeur qui minimise R_k. Il suffit donc de résoudre le problème de sac à dos.

Une méthode de programmation dynamique est proposée pour résoudre ce problème de sac à doc. La programmation dynamique est une méthode de résolution, pour les problèmes qui satisfont au principe d'optimalité de Bellman [BEL 55] : une sous-trajectoire d'une trajectoire optimale est elle-même optimale pour la fonction objectif restreinte aux trajectoires ayant pour origine celle de cette sous-trajectoire. Ce principe permet une méthode de résolution ascendante, qui détermine une solution optimale d'un problème à partir des solutions de tous les sous-problèmes. Cette méthode est efficace pour résoudre les problèmes de sac à dos [PEL 04]. Dans ce qui suit, nous donnons la procédure de résolution du problème de génération de colonnes.

La matrice $g_l(C)$ peut être obtenue par la programmation dynamique en se restreignant aux opérations dans S_k comme suit :

$$g_{l+1}(C) = \begin{cases} g_l(C), & \text{si } t_l < C \text{ ou } l+1 \notin S_k \\ \max\{g_l(C), g_l(C - t_l) + p_{lk}\}, & \text{sinon} \end{cases} \qquad \text{4. 29}$$

avec

$$g_1(C) = \begin{cases} 0, & \text{si } t_1 < C \text{ ou } 1 \notin S_k \\ \max\{0, p_{1k}\}, & \text{sinon} \end{cases}$$

Exemple de résolution du SALBP-2 par la génération de colonnes

Dans ce qui suit, pour bien expliquer notre méthode un petit problème avec 5 opérations et 3 stations est résolu par la génération de colonnes. Les contraintes de précédence et les temps opératoires sont présentés dans le graphe suivant :

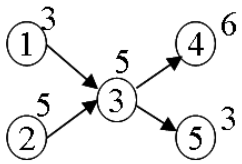


Fig. 4. 1 Système étudié dans l'exemple

Le problème original est donné par les équations suivantes :

Min c

$$\sum_{k=1}^{3} x_{1k} = 1$$

$$\sum_{k=1}^{3} x_{2k} = 1$$

$$\sum_{k=1}^{3} x_{3k} = 1$$

$$c \geq \sum_{i=1}^{5} x_{i1} t_i$$

$$c \geq \sum_{i=1}^{5} x_{i2} t_i$$

$$c \geq \sum_{i=1}^{5} x_{i3} t_i$$

$$\sum_{k=1}^{3} k x_{3k} \geq \sum_{k=1}^{3} k x_{1k}$$

$$\sum_{k=1}^{3} k x_{3k} \geq \sum_{k=1}^{3} k x_{2k}$$

$$\sum_{k=1}^{3} k x_{4k} \geq \sum_{k=1}^{3} k x_{3k}$$

$$\sum_{k=1}^{3} k x_{5k} \geq \sum_{k=1}^{3} k x_{3k}$$

$$x_{ik} \in \{0,1\}$$

avec k qui correspond aux stations, i aux opérations et c le temps de cycle.

Nous ne présentons pas ici le problème maître car il correspond à la combinaison de toutes les solutions possibles.
Le processus commence par l'insertion des colonnes déterminées par une solution initiale dans le problème maître restreint. Dans nos travaux, la solution initiale est générée par la première phase de l'heuristique que nous proposons (voir Chapitre 2).

La solution initiale est représentée par un ensemble de variables q^k_{ij} données dans le tableau suivant :

q^1_{10}	q^1_{20}	q^1_{30}	q^1_{40}	q^1_{50}
1	1	0	0	0
q^2_{10}	q^2_{20}	q^2_{30}	q^2_{40}	q^2_{50}
0	0	1	0	0
q^3_{10}	q^3_{20}	q^3_{30}	q^3_{40}	q^3_{50}
0	0	0	1	1

Où *i* représente l'opération, *k* correspond à la station et *j* l'indice correspondant à l'itération. Trois colonnes correspondantes au trois stations peuvent donc être déterminées pour la solution initiale. Pour cette solution initiale (itération 0), dans la station 1 on a donc les opérations 1 et 2, dans la station 2 on a l'opération 3 et dans la dernière station on a les opérations 4 et 5.
Nous ajoutons ensuite les trois colonnes dans le problème restreint (voir les équations 4.17 – 4.20).

$$
\begin{aligned}
&\text{Min } 8^N Z_{01} + 5^N Z_{02} + 9^N Z_{03} \\
&-Z_{01} + 2Z_{02} \geq 0 \\
&-Z_{01} + 2Z_{02} \geq 0 \\
&-2Z_{02} + 3Z_{03} \geq 0 \\
&-2Z_{02} + 3Z_{03} \geq 0 \\
&Z_{01} = 1 \\
&Z_{01} = 1 \\
&Z_{02} = 1 \\
&Z_{03} = 1 \\
&Z_{03} = 1 \\
&Z_{01} \leq 1 \\
&Z_{02} \leq 1 \\
&Z_{03} \leq 1
\end{aligned}
$$

Pour la fonction objectif (équation 4.17), le 8 correspond donc à la somme des temps opératoires des opérations 1 et 2 affectées à la première station, le 5 correspond à la durée de l'opération 3 affectée à la deuxième station et le 9 correspond à la somme des temps opératoires des opérations 4 et 5 affectées à la dernière station. Dans notre programme et pour cet exemple, nous avons choisi $N = 3$ (par expérimentation).

Remarque : pour nos applications numériques nous avons pris $N=4$.

Les Z_{jk} correspondent aux variables de décisions de notre problème (avec k la station et j l'itération considérée).

Le problème maître restreint courant est ensuite résolu par le Simplexe. Les vecteurs des variables duales des contraintes dans le problème sont obtenus. Pour notre exemple, nous obtenons :

$$\lambda_{hi} = \{0, 512, 0, 125\}$$

$$\mu_i = \{729, 0, 0, 0, 0\}$$

$$\gamma_k = \{0, 0, 0\}$$

avec λ_{hi} le vecteur des variables duales pour les contraintes de précédence, μ_i le vecteur des variables duales qui assure de respecter les contraintes liées au fait qu'une opération ne peut être affectée qu'à une seule station et γ_k le vecteur de variables duales qui assure qu'une seule colonne au plus soit définie par station.

Ce problème se décompose ensuite en 3 sous-problèmes de génération de colonnes, chacun relatif à une station. Le problème de génération de colonnes de la station k après reformulation (équations 4. 26a et 4.26b) est donné par :

$$R_1 = \underset{C=0,1,\ldots,13}{\text{Min}} f(C) - g_5(C) \text{ avec } g_l(C) = \underset{q_{ij}^k}{\text{Max}}\, 729q_{11}^1 - 512q_{21}^1 + 387q_{31}^1 + 125q_{41}^1$$

$$R_2 = \underset{C=0,1,\ldots,13}{\text{Min}} f(C) - g_5(C) \text{ avec } g_l(C) = \underset{q_{ij}^k}{\text{Max}}\, 729q_{11}^1 - 1024q_{21}^1 + 774q_{31}^1 + 250q_{41}^1$$

$$R_3 = \underset{C=0,1,\ldots,13}{\text{Min}} f(C) - g_5(C) \text{ avec } g_l(C) = \underset{q_{ij}^k}{\text{Max}}\, 729q_{11}^1 - 1536q_{21}^1 + 1161q_{31}^1 + 375q_{41}^1$$

où R_1 correspond au coût réduit associé à la station 1 pour l'itération 1, R_2 correspond au coût réduit associé à la station 2 pour l'itération 1 et R_3 correspond au coût réduit associé à la station 3 pour l'itération 1. 13 correspond à la borne supérieure du temps de cycle déterminée par l'équation (1. 32).

Les problèmes de génération de colonnes sont alors résolus par maximisation d'un ensemble de problèmes sac-à-doc.

Si au moins un R_k est négatif, les colonnes déterminées par les problèmes de génération de colonnes sont ajoutées dans le problème maître restreint. Le processus se termine quand tous les R_k sont positifs.

4.4 Application

Dans cette sous-section, nous appliquons la méthode proposée pour déterminer la borne inférieure du temps de cycle pour une ligne d'assemblage. Dans un premier temps, nous donnons l'algorithme de résolution. Ensuite nous l'appliquons sur des exemples numériques. Les résultats numériques sont comparés avec la borne inférieure déterminée par la formulation LC1 (voir Section 1.3.3.1).

4.3.3 Algorithme

Algorithme 4.1 : Procédure de résolution

Etape 1 : Lire les donnés du problème à résoudre : le nombre de stations, les temps opératoires, les contraintes de précédence.

Etape 2 : Définir N comme un nombre entier positif ($N=4$ pour nos applications).

Etape 3 : Générer une solution initiale du problème SALBP-2 selon le première phase de l'heuristique proposée (voir Chapitre 2, Algorithme 2.1) et déterminer les colonnes définies par cette solution initiale ;

Etape 4 : Ajouter les colonnes courantes dans le problème maître restreint (voir les équations 4.17 – 4.21).

Etape 5 : Résoudre le problème maître restreint courant par la méthode Simplexe.

Etape 6 : Déterminer les vecteurs des variables duales des contraintes (voir les équations 4.18 – 4.20).

Etape 7 : Construire un problème de génération de colonnes pour chaque station. Résoudre les problèmes de génération de colonnes par la programmation dynamique et déterminer les nouvelles colonnes.

Etape 8 : Si toutes les valeurs de la fonction objectif des problèmes de génération de colonnes sont positives, le processus est arrêté, et la borne inférieure du temps de cycle est déterminée par C_{jk} maximum avec Z_{jk} positif, sinon retourner à l'étape 4.

4.3.4 Résultats numériques

Dans cette section, les résultats numériques montrent la performance de la méthode pour déterminer des bornes inférieures du temps de cycle du SALBP-2.

La méthode est programmée en Langage C++ avec un compilateur de Microsoft Visual C++ 6.0 et l'outil d'optimisation CPLEX qui est conçu pour résoudre les problèmes d'optimisation mathématiques de grande taille [CPLEX]. Le programme est exécuté sur un ordinateur portable avec 2GHz AMD CPU. Les bornes inférieures obtenues par notre méthode sont comparées avec LC1 (voir Section 1.3.3.1).

Les problèmes testés sont présentés dans les Fig. 4. 2 - Fig. 4. 5.

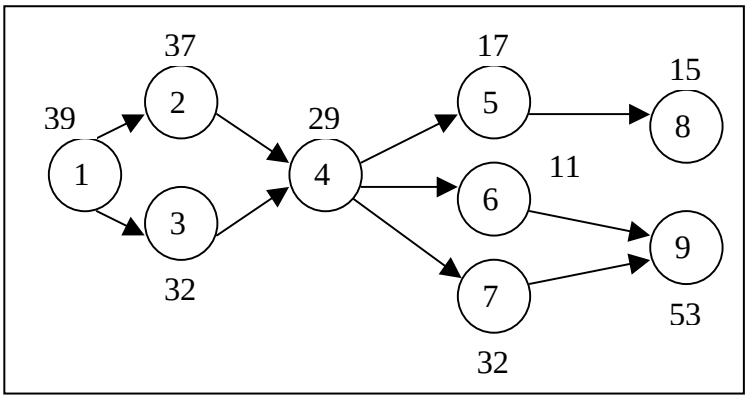


Fig. 4. 2 P_1

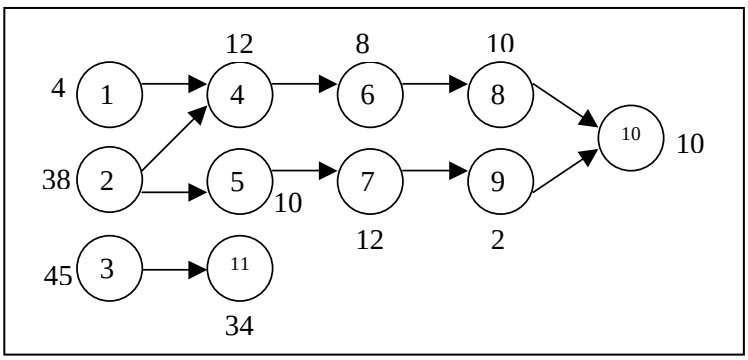


Fig. 4. 3 P_2

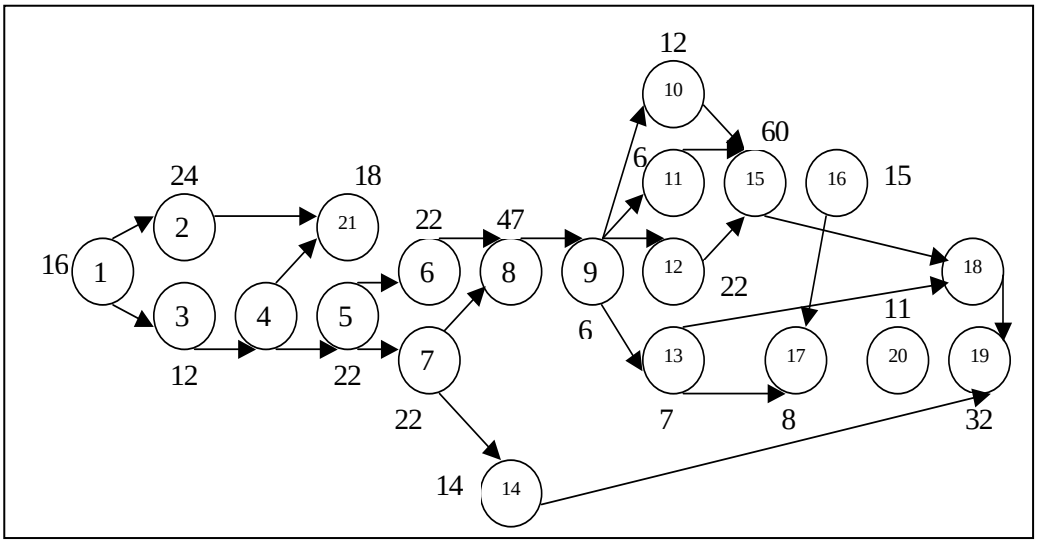


Fig. 4. 4 P_3

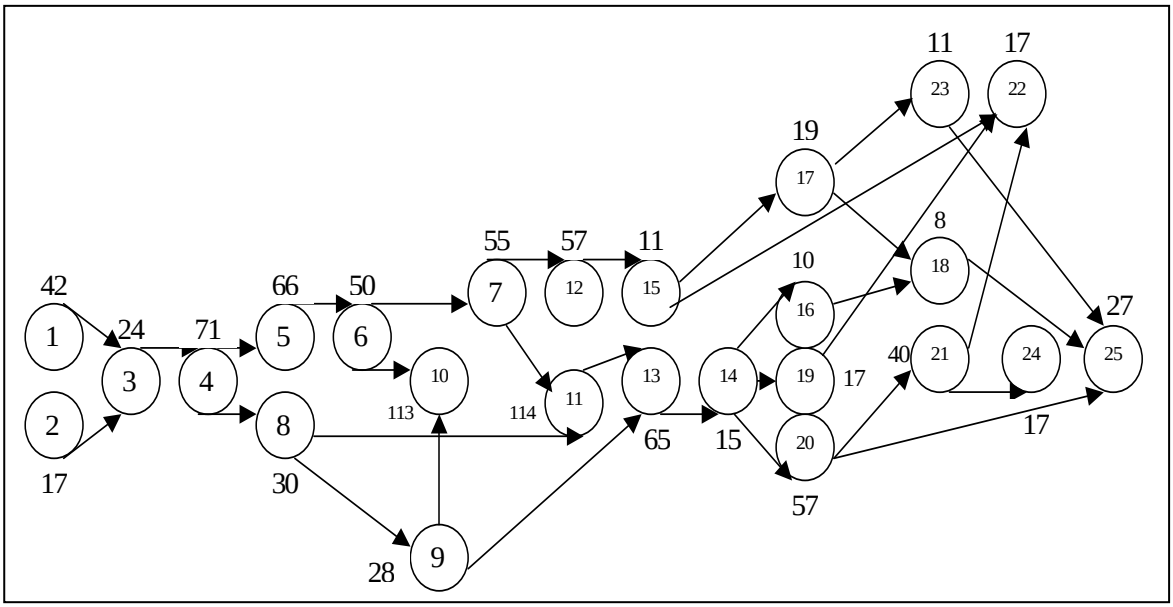


Fig. 4. 5 P_4

Problème	N_st	N_op	CT optimal	Temps de cycle optimal par EM	LC1	Méthode proposée	
						Borne	CPU(s)
P_1	3	9	96	96	89	93	1,70
	4		71	71	67	68	1,83
	5		64	64	53	61	1,89
P_2	3	11	62	62	62	62	3,51
	4		48	48	47	47	3,92
	5		45	45	45	45	4,06

P_3	6	21	-	80	76	76	12,90
	7		-	70	65	66	13,45
	8		-	66	60	64	14,20
P_4	6	25	-	171	164	168	25,07
	7		-	154	141	150	26,12
	8		-	137	123	133	26,75

Tableau 4. 1 Bornes inférieures du temps de cycle

Selon les résultats obtenus dans le Tableau 4. 1, nous voyons que la méthode proposée permet d'obtenir de bonnes bornes inférieures des temps de cycle pour tous les problèmes. Pour les problèmes dont le temps de cycle minimum est égal au temps opératoire maximum, LC1 (voir Section 1.3.3.1) peut être égale aux temps de cycle minimum obtenus par EM. Par exemple, c'est le cas du problème P_1 avec 3 stations. Cependant ce n'est pas assuré pour les problèmes avec des temps opératoires grands.

En combinant la méthode de génération de colonnes et la séparation et évaluation, des solutions admissibles des problèmes peuvent être aussi obtenues. A cause de problèmes de licence de CPLEX, nous n'avons pas pu exécuter cette méthode pour les problèmes de plus de 300 variables.

4.5 Conclusion

Nous proposons une méthode utilisant la génération de colonnes pour déterminer une borne inférieure du temps de cycle pour des problèmes d'équilibrage de lignes d'assemblage de type II. Ce problème est formulé en programme linéaire en nombre entiers. Une formulation orientée colonnes permettant d'obtenir une bonne borne inférieure du temps de cycle *c* est proposée à partir de la décomposition de Dantzig-Wolfe. La méthode de génération de colonnes est utilisée pour résoudre la relaxation linéaire du problème formulé et déterminer une borne inférieure du temps de cycle. Cette borne inférieure peut être utilisée ensuite comme solution initiale pour une méta-heuristique ou être combiné à une heuristique pour accélérer le calcul de la solution optimale. Les problèmes de génération de colonnes sont définis comme les problèmes de sac à dos résolus par la méthode de programmation dynamique. La performance de la méthode proposée est montrée dans la section 4.4.

Ainsi, nous pourrions combiner notre méthode au branch and bound pour déterminer une solution admissible. En effet, notre méthode ne peut pas résoudre des problèmes avec des temps de cycle plus grands que 600 unités de temps à cause de la définition de la fonction $f(x)$ – problème d'overflow - (voir l'équation 4.16). Dans les travaux futurs, nous devons donc essayer de proposer une autre formulation de la fonction objectif de façon à ne plus avoir cette limitation.

Conclusion Générale

Les travaux présentés dans ce mémoire sont consacrés aux problèmes de l'équilibrage de lignes d'assemblage. Ce type de problèmes consiste à affecter des opérations aux stations de la ligne de façon à équilibrer les charges de ces stations tout en respectant les contraintes de production. Dans ce mémoire, nous avons considéré les contraintes de précédence entre les opérations ainsi qu'un nombre fixé de stations. Ce modèle correspond aux problèmes d'équilibrage de lignes d'assemblage de type II. Tout d'abord nous avons traité le cas des lignes d'assemblage où les durées des opérations sont déterministes. Celles-ci correspondent aux lignes d'assemblage automatisées sans prise en compte des pannes des machines. Ensuite, nous avons étudié le cas des lignes d'assemblage manuelles caractérisées par des durées d'opérations aléatoires. Dans les deux cas, l'affectation des opérations sur les machines de la ligne est réalisée afin d'optimiser des performances telles que le temps de cycle et/ou la fiabilité de la ligne.

Après avoir fait un état de l'art des méthodes de résolution des problèmes d'équilibrage de lignes d'assemblage les plus importantes par rapport à nos travaux, nous avons présenté nos travaux sur la minimisation du temps de cycle de lignes d'assemblage de type II caractérisées par des durées opératoires déterministes. Nous avons défini une nouvelle heuristique qui s'adapte pleinement aux problèmes d'équilibrage de lignes d'assemblage de type II. Cette heuristique procède de la façon suivante : elle génère d'abord la solution initiale du problème d'équilibrage de lignes d'assemblage selon les caractéristiques propres à ce problème. Une procédure de transfert et d'échange est utilisée pour améliorer la solution initiale. Une application numérique nous a permis de mettre en évidence l'efficacité de notre méthode par rapport à l'heuristique bi-directionnelle définie dans la littérature considérée comme étant l'une des plus performantes.

Ensuite, nous considérons deux méta-heuristiques existantes dans la littérature et nous les adaptons aux problèmes d'équilibrage de lignes d'assemblage de type II déterministes: l'algorithme basé sur le mécanisme d'électromagnétisme (EM) et l'estimation de distribution

(ED). EM et ED sont des méta-heuristiques basées sur la génération de populations de solutions. Leur avantage commun est qu'elles n'ont pas beaucoup de paramètres à définir. Elles sont donc faciles à implémenter et leurs performances sont stables. En comparant les deux méthodes, ED utilise un ensemble de vecteurs de nombres entiers alors que la méthode EM utilise des vecteurs de nombres réels. Par conséquent, l'espace de recherche de ED est plus petit que celui de EM. ED peut converger plus rapidement. Par contre, EM permet d'obtenir de meilleures solutions.

Puis, nous avons étudié le problème d'équilibrage de lignes d'assemblage de type II caractérisé par des durées opératoires aléatoires. Pour ce type de lignes, nous avons considéré une mesure de performance supplémentaire, qui est la fiabilité de la ligne. Cette mesure de performance représente la probabilité que le temps de cycle ne soit pas dépassé. Nous avons considéré deux problèmes dépendants de la définition choisie pour intégrer la fiabilité de la ligne : cette fiabilité correspond dans un premier temps à une contrainte et dans un deuxième temps à un objectif à optimiser. Dans le premier cas, la fiabilité de la ligne est spécifiée par une contrainte. Nous cherchons à minimiser le temps de cycle tout en respectant une valeur minimale donnée pour la fiabilité de la ligne. Pour résoudre ce problème nous avons développé un algorithme basé sur la méthode EM. Dans le deuxième cas, nous considérons un problème multi-critère où deux critères ont été considérés simultanément : le temps de cycle et la fiabilité de la ligne. Dans ce cas on ne cherche plus une solution optimale, mais un ensemble de solutions appelées Pareto optimales Un nouveau mécanisme basé sur la méthode EM est ainsi défini pour améliorer les solutions Pareto optimales obtenues.

Il est connu que, généralement, le temps de calcul d'une méthode d'optimisation dépend fortement de la qualité de la solution initiale. Ainsi, afin d'améliorer la convergence des méthodes que nous avons proposées, nous étudions dans le dernier chapitre une méthode pour calculer la borne inférieure de la solution optimale pour un problème d'équilibrage de lignes d'assemblage déterministe. Ce problème est formulé en programme linéaire en nombre entiers. Une formulation orientée colonnes permettant d'obtenir une bonne borne inférieure du temps de cycle c est proposée à partir de la décomposition de Dantzig-Wolfe. La méthode de génération de colonnes est utilisée pour résoudre la relaxation linéaire du problème formulé et déterminer une borne inférieure du temps de cycle. Les problèmes de génération de colonnes sont définis comme les problèmes de sac à dos résolus par la méthode de programmation dynamique. Cette méthode nous a permis de déterminer une borne inférieure du temps de cycle dans un temps de simulation acceptable.

Comme perspectives à nos travaux, la méthode issue de la génération de colonnes peut être combinée à une heuristique pour déterminer une bonne solution et transformer la solution issue de la génération de colonnes en une solution admissible du problème.

L'heuristique que nous proposons pourrait être utilisée pour définir les solutions initiales pour ED. Ainsi, la première phase de l'heuristique proposée pourrait être améliorée en considerer une borne inférieure du temps de cycle.

De plus, pour les méta-heuristiques, il faudrait tout d'abord arriver à simplifier le mécanisme EM qui pour l'instant, même si il est efficace, peut être à notre avis amélioré notamment par rapport au processus de codage du problème SALBP-2 sous une forme utilisable par EM. Pour le méta-heuristique ED, il faudrait pouvoir tester d'autres lois de distribution en fonction du problème considéré et des contraintes de ce problème.

Enfin, il serait également très intéressant de considérer des contraintes ou des objectifs supplémentaires comme par exemple les coûts de production, la qualité des produits, etc., ainsi que d'adapter la méthode ED à des problèmes multi-objectif.

Bibliographie

[ABD 95] Abdallah Enmer
Contribution à l'étude de l'équilibrage des lignes d'assemblage
Thèse de l'institut national des sciences appliquées de Lyon 1995

[AGH 95] Aghezzaf E. H; et Artiba A.
A Lagrangian relaxation technique for the general assembly line balancing problem
Journal of Intelligent Manufacturing, 1995,vol, N° 2, pp 123-131

[AJE 98] Ajenblit, D.A.; Wainwright, R.L.
Applying genetic algorithms to the U-shaped assembly line balancingproblem
The May 1998 IEEE International Conference on Vol Issue, 4-9 Page(s):96–101

[AME 06] Amen Matthias.
Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds
Operation research, 2006, vol 163, issue 3, pp 747-770

[ARC 66] Arcus A. L.
COMSOAL a computer method of sequencing operations for assembly lines
Production research 1966 vol 4, N° 4, 259-277

[ARM 96] Armin Scholl, Stefan Voβ
Simple assembly line balancing-heuristic approaches
Journal of heuristics 1996 (2) 217-244

[ARM 99] Armin Scholl
Balancing and sequencing of assembly lines
Management science 1999. ISBN 3-7908-1180-7

[ARM 03] Armin Scholl, Becker, C.
State of the art exact and heristic solution procedures for simple assembly line balancing
Jenaer Schrifter Zur Wirtschaftswissenschaft 20/03, FSU Jena
Sur la site: http://econpapers.repec.org/paper/jenjenasw/2003-20.htm

[BAR 96] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, Pamela H. Vance.
Branch_and_Price: Column Generation for Solving Huge Integer Programs
Research work supported by: NSF SES-91-22674, NSF and AFORS DDM-9115768, NSF DDM-9058074, NSF DMI- 9410102and IBM MHV2379, 1996.
Sur la site: http://www.lehigh.edu/~jtl3/teaching/ie418/papers/or46b.pdf

[BAY 86] Baybars,I
Asurvey of exact algorithms for the simple assemblyline balancing problem
Management science, 1986, vol 32, N° 8, pp 909-932

[BAU 00] Bautista, J.,Suarez, R., Mateo, M., Companys, R.,
Local search heuristics for the assembly line balancing problem with incompatibilities between tasks
Probceedings of the 2000 IEEE International conference on robotics and automation, San Francisco, CV, 2404-2409

[BEA 94] Bean. J.
Genetics and random keys for sequencing and optimization.
ORSA Journal on Computing, 1994, vol 6(2), pp 154-160.

[BEL 06] Sana Belmokhtar, Alexandre Dolgui, Xavier Delorme, Ivan Ignatenko
Optimizing modular machining line design problem
Conference IFAC 2006

[BIR 03] Birbil. S. I. and Fang. S.-C.
An electromagnetism- like mechanism for global optimization.
Journal of Global Optimization, 2003, vol 25(3), pp263-282.

[BOY 07a] Boysen, N.; M. Fliedner and A. Scholl
A classification for assembly line balancing problems
European Journal of Operational Research 2007, 183, 674-693,

[BOY 07b] Boysen, N.; M. Fliedner and A. Scholl
Assembly line balancing: Which model to use when?
International Journal of Production Economics 2007
Sur la site: http://econpapers.repec.org/paper/jenjenasw/2006-23.htm

[DOL 06] Alexandre Dolgui
Balancing Assembly and Transfer Lines
European Journal of Operational Research, vol 168, 3, 1 February 2006, pp 663-665.

[BRA 99] Brahim Pierre De Lit, Fabrice Pellichero, Emanuel Falkenauer, Alain Delchambre
Applying the equal piles problem to balance assembly lines
Proceedings of the 1999 IEEE international symposium on assembly and task planning

[BUX 74] BUXEY, G.M.,
Assembly line balancing with multiple stations,
Management Science, 1974, vol 20,No 6, pp 1010-1021.

[CAR 99] Carlos A.
A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques,
Knowledge and Information Systems, 1999, Vol. 1, No. 3, pp. 269-308.

[CEL 99] G. Celano, S. Fichera, V. Grasso, U. la Commare, G. Perrone
An evolutionary approach to multi-objective scheduling of mixed model assembly lines
Computers and industrial engineering 1999, (37), 69-73

[CHR 05] Christian Becker, Armin Scholl
A survey on problems and methods in generalized assembly line balancing
European Journal of Operational Research 2006, 168: 694-715

[CPLEX] Sur la site: http://www.ilog.com/products/optimization/archive.cfm#2

[DAN 60] Dantzig, G. B., Wolfe, P..
Decomposition principle for linear programming
Operation research 1960, vol 8. pp 101-111

[DEC 89] Deckro, R. F.
Balancing cycle time and workstations
IIE transactions, 1989, 21, 106-111

[DES 02] Deshpande Vinayak, Rajaram Kumar, Guignard Monique
An integer programming based formulation of the U-line balancing problem
Sur la site: http://opim.wharton.upenn.edu/~guignard/papers/ippaper_ejor.pdf

[DRA 02] Dragan. S
Single-objective vs. Multiobjective Optimisation for Integrated Decision Support

Integrated Assessment and Decision Support, Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society, 2002 24-27 June, Lugano, Switzerland, Vol. 1, pp. 7-12.

[FIS 85] M.L. Fisher
An applications oriented guide to lagrangian relaxation
Interfaces, 1985, 15: 10-21

[FIN 04] Brigitte Finel
Structuration de lignes d'usinage : méthodes exactes et heuristiques
Thèse d'université de metz 2004

[GER 98] Gerry Aase, Steven E. Moss and Glenn Brame,
An Application of U-Shaped Lines with Parallel Work Stations
Decision Sciences Conference Proceedings Las Vegas, NV, 1998, pp. 1403-1405.

[GOK 98] Hadi Gokçen, Erdal Erel
Binary integer formulation for mixed-model assembly line balancing problem
Computers and industrial engineering 1998, vol 34, issue 2, pp 451-461

[GOK 06] Hadi Gokçen , Kursad Agpak , Recep Benzer
Balancing of parallel assembly lines
Production economics, 2006, vol. 103, issue 2, 600-609

[GRA 88] Graves C. S., Redfield Holmes Carol
Equipment selection and task assignment for multi-product assembly system design
Flexible manufacturing systems 1988, vol 1, pp 31-50

[GRA 95] Graves Robert
A review of line balancing for flow lines
Sur site:http://mhia.org/vango/Core/orders/product.aspx?catid=26&prodid=193

[GU 06] Gu L, S. Hennequin, A. Sava, and X. Xie
EM algorithm for Solving the Stochastic Assembly Line Balancing Problem
In : 12th IFAC symposium on information control problems in manufacturing INCOM'06, Saint-Etienne, France, May 2006.

[GU 07a] Gu L, S. Hennequin, A. Sava, and X. Xie
Electromagnetism-Like Mechanism Algorithm For Stochastic Assembly Line Balancing With Reliability Constant *
International Conference on Industrial Engineering and Systems Management IESM 2007, BeiJing – China, May 2007.

[GU 07b] Gu L, S. Hennequin, A. Sava, and X. Xie
Assembly Line Balancing Problems Solved by Estimation of Distribution
Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on vol , Issue , 22-25 September. 2007 pp:123 – 127

[GU 07c] Gu L, S. Hennequin, A. Sava, and X. Xie
A New Heuristic for Simple Assembly Line Balancing type II Problem
IX Triennial International SAUM Conference on Systems, Automatic Control and Measurements, Niš, Serbia, November ., 2007

[GU 07d] Liya GU, Alexandre SAVA,Sophie HENNEQUIN and Xiaolan XIE
Electro-Magnetism Based Optimization For Stochastic Assembly Line Balancing With Reliability Constant
Soumis pour publication à Journal of Operation and Logistic

[GUI 03] Monique Guignard
Lagrangean relaxation
Top 2003, vol 11, N° 2, pp 151-200.

[GUN 83] R.E. Gunther et al,
Currently Practiced Formulations for the Assembly Line Balance Problem
Journal of Operations Management, 1983, vol. 3, No. 4, pp. 209-221.

[HAC 89] Hackman S. T, Magazine M. J. ; Wee T. S.
Fast, effective algorithms for simple assembly line balancing problems
Operations research, 1988, ISSN 0030-vol. 37, n°6, pp. 916-924 (2 p.)

[HAH 72] R. Hahn,
Produktionsplanung bei Linienfertigung
Walter de Gruyter, Berlin, 1972.

[HEL 71] Held M., Karp R.M.
The Travelling Salesman Problem and Minimum Spanning Trees-Part Ⅱ
Mathematical programming, 1971, vol 1, pp 6～25.

[HOF 63] Hoffmann, TR
Assembly line balancing with a precedence matrix
Management Science, 1963, vol 9, pp. 551–562

[HOF 92] Hoffmann, TR
Eureka: A Hybrid System for Assembly Line Balancing
Management Science, 1992 Vol. 38, No. 1, pp. 39-47

[JIN 03] Jin Mingzhou S.David WU
A new heuristic method for mixed model assembly line balancing problem
Computers and Industrial Engineering 2003 Volume 44， Issue 1 159 – 169

[JOA 07] Joaquin Bautista, Jord Pereira
ant algorithms for a time and space constrained assembly line balancing problem
Operational Rearch 2007 (177) 2016-2023

[JOH 83] Johnson, R. V.,
A branch and bound algorithm for assembly line balancing problems with formulation irregularities
Managenment science, 1983, (29), 1309-1324

[JOH 88] Johnson, R.V.
Optimally Balancing Large Assembly Lines with Fable
Management Science, 1988 Vol. 34, No. 1, pp. 240-253
.

[JON 05] Jonathan F. Bard and Hadi W. Purnomo
A column generation-based approach to solve the preference scheduling problem for nurses with downgrading
Socio-Economic Planning Sciences, 2005, vol. 39, issue 3, pages 193-213

[JOS 00] Joseph Bukchin, Michal Tzur
Design of flexible assembly line to minimize equipment cost
Design and manufacturing 2000 Volume 32, Number 7 585-598

[JOS 02a] Joseph Bukchin, Ezey M. Dar-El, Jacob Rubinovitz
Mixed model assembly line design in a make-to-order environment
Computers and Industrial Engineering,2002 Volume 41 Issue 4 405-421

[JOS 02b] Joseph Bukchin, Jacob Rubinovitz
A weighted approach for assembly line design with station paralleling and equipment selection

IIE transactions 2002 vol 35 N° 010005

[JUN 97] Jung lyu
A single run optimization algorithm for stochastic assembly line balancing problem
Journal of manufacturing systems 1997 (16) 3

[KAO 79] Kao, E.P.C.
Computational experience with a stochastic assembly line balancing algorithm
Computer & Operations Research, 1979 (6), 79-86.

[KIL 61a] Kilbridge D. M., Wester Leon
A heuristic method of assembly line balancing
Journal of industrial engineering 1961, vol XII No 4 292-198

[KIL 61b] Lilbridge, D.M., Wester Leon
The balancing delay problem
Management science, 1961, 8, 69-84

[KIM 00] Kim Y.K.; Kim Y.; Kim Y.J.
Two-sided assembly line balancing: a genetic algorithm approach
Production Planning and Control, 1 January 2000,Vol 11, No 1, pp. 44-53(10)

[KIR 83] Kirkpatrick S, Gelatt C.D., Vecchi M.P.
Optimization by simulated annealing
Science, 1983, 200(5), PP 671-680

[LAP 06] Sophie D. Lapierre , Angel Ruiz , Patrick Soriano
Balancing assembly lines with tabu search
Operational Research (2006), 168, 826–837

[LIU 03] Liu S. B., Ong H. L., Huang H. C.
Two bi-directional heuristics for the assembly line type II problem
Advanced Manufacturing Technology 2003 vol 22 N° 9-10 656-661

[LIU 05] Liu S. B., Ong H. L., Huang H. C.
A bidirectional heuristic for stochastic assembly line balancing Type II problem
Advanced Manufacturing Technology 2005, vol 25 N° 1-2 71-77

[LIV 06] Livitin, G. and Rubinovitz, J. and Shnits, B.
A genetic algorithm for robotic assembly line balancing
European Journal of Operational Research, 168, 811-825.

[LOW 05] Low Chinyao
Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines
Computer and operation research, 2005, vol. 32, issue 8, pp 2013-2025

[MAH 98] Philippe Mahey
Relaxation lagrangienne
Cours d'Optimisation Convexe, 1998, ISIMA.
Sur la site: http://www.isima.fr/mahey/PART2.PS

[MAT 07] Matoušek Jiří and Gärtner Bernd
Integer programming and LP relaxation
Livre Understanding and Using Linear Programming pp 29-40
Sur la site: http://www.springerlink.com/content/u1wrq0548173q172/

[MCM 98] Mcmullen P. R., FRAZIER G. V.
Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations

International Journal of Production Research 1998, Vol 36, Issue 10 2717 – 2741

[MCM 03] Mcmullen P. R., P. Tarasewich
Using ant techniques to solve the assembly line balancing problem
IIE Transactions 2003, vol 35, pp 605 – 617

[MCM 06] Mcmullen P. R., P. Tarasewich
Multi-objective assembly line balancing via a modified ant colony optimization technique
Production Research, 1 January 2006, Vol 44, pp 27 – 42

[MUH 96] Mühlenbein, H. and Paaß,G.
From Recombination of Genes to the Estimation of Distributions I. Binary Parameters
In Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature-PPSN IV, 1996, pp 178-187.

[MUH 98] Mühlenbein, H.
The equation for response to selection and its use for prediction
Evolutionary Computation, 1998, vol 5(3), pp 303-346.

[MUR 95] Murata, Y, Ishibuchi, H and Tanaka, H.
Multi-objective genetic algorithm and its applications to flowshop scheduling
Computers Ind. Engng, 1996, Vol. 30, no. 4, pp 957-968.

[NOO 05] A. Noorul Haq, K. Rengarajan and J.Jayaprakash
A hybrid genetic algorithm approach to mixed-model assembly line balancing
The International Journal of Advanced Manufacturing Technology 2006 Volume 28, Numbers 3-4 / March 337-341

[OCE 02] Očenášek Jiří, Schwarz Josef.
Estimation Distribution Algorithm for mixed continuous-discrete optimization problems
In: Proceedings of the 2nd Euro-International Symposium on Computational Intelligence, Kosice, SK, IOS, 2002, pp. 227-232, ISBN 1-58603-256-9

[OLG 04] Olga Roudenko
Application des algorithmes evolutionnaires aux problèmes d'optimisation multi-objectif avec contraintes
Thèse en mathématiques appliquées de l'école polytechnique, soutenue le 5 mars 2004.
Sur la site : http://www.imprimerie.polytechnique.fr/Theses/Files/Roudenko.pdf

[PAT 75] James H. Patterson, Joseph J. Albracht
Assembly-Line Balancing: Zero-One Programming with Fibonacci Search
Operations Research 1975, Vol. 23, No. 1, pp. 166-172

[PEE 06] Peeters Marc, Degraeve Zeger
An linear programming based lower bound for the simple assembly line balancing problem
Operational Research 2006, vol 168, pp 716–731

[PEI 04] Peitsang, W. and Wenhung, Y.
An electromagnetism algorithm of neural network analysis - an application to textile retail operation
Journal of The Chinese Institute of Industrial Engineers, 2004, vol 21(1), pp 59-67.

[PLA 85] Plateau, Gérard; Elkihel, Moussa
A hybrid method for the 0-1 knapsack problem. methods
Operation Research. 1985, 49, 277-293.

[PON 99] Ponnambalam S. G., Aravindan P., Mogileeswar Naidu G.
A comparative evaluation of assembly line balancing heuristics
Advanced manufacturing technology 1999 (15) 577-586

[PON 00] Ponnambalam S. G., Aravindan P., Mogileeswar Naidu G.
A multi-objective genetic algorithm for solving assembly line balancing problem
Advanced manufacturing technology, 2000, (16, 341-352)

[RAC 91] Rachamadugu R, Talbot B
Improving the equality of workload assignments in assembly lines.
International journal production research, 1991, vol 29(3), pp 619-633

[RAF 02] Rafael Pastor et Carlos Andrés
Scheduling and line balancing in a multi-product assembly line using tabu search
Sur la site: http://citeseer.ist.psu.edu/638333.html

[RAM 70] Ramsing, K and D. Downing.
Assembly line balancing with variable element times
Industrial Engineering 1970 (January), 41-43.

[REK 02] Brahim Rekiek, Alexandre Dolgui, Alain Delchambre et Antoneta Bratcu
State of art of optimization methods for assembly line design
Annual Reviews in Control, vol 26, 2, 2002, pp 163-174

[ROB 89] Robert L. Carraway
A Dynamic Programming Approach to Stochastic Assembly Line Balancing
Management Science, 1989, vol. 35, No. 4, pp. 459-471

[ROB 96] Robert Klein, Armin Scholl
Maximizing the production rate in simple assembly line balancing — A branch and bound procedure
European Journal of Operational Research 7 June 1996 Vol 91, Issue 2, 367-385

[SAL 55] Salveson, M. E
The assembly line balancing problem
Journal of Industrial Engineering, 1955, 6, 3.

[SEL 01] Sellmann Meinolf, Fahle Torsten
CP-bqsed lagrangian relaxation for a multimedia application
In [17], 2001
http://citeseer.ist.psu.edu/603853.html

[SHA 02] Shaharuddin Salleh, Bahrom Sanugi, Hishamuddin Jamaluddin, Stephan Olariu, Albert Y. Zomaya
Enhanced Simulated Annealing Technique for the Single-Row Routing Problem
The Journal of Supercomputing, 2002, Vol 21, Iss 3, pp 285-302

[SPI 00] Spinellis, C., Papadopoulos C.
Large production line optimization using simulated annealing
Production research, 2000, vol 38, pp 509-541.

[SPR 99] Sprecher A.
A competitive branch-and-bound algorithm for the simple assembly line balancing problem
Production research, 1999, vol 37, N° 8, pp 1787-1816(30).

[STE 87] Stephen C. Graves, Carol A. Holmes
Equipment selection and task assignment for multiproduct assembly system design
Work paper of massachusetts institute of technology OR 164-87, June 1987 sur site:
http://econpapers.repec.org/paper/mitsloanp/2171.htm

[SNI 81] Sniedovich, M.
Analysis of a preference order assembly line problem
Management Science, 1981, (27), 1067-1080

[SUB 99] Subhash C. Sarin, Erdal Erel, Ezey M. Dar-El
A methodology for solving single-model, stochastic assembly line balancing problem
Management science 1999, (27) 525-535

[SUR 94] Sutesh G. Sahu S.
Stochastic assembly line balancing using simulated annealing
Management Science. 1994, 27,523-535

[SUR 96] Suresh, G., V. V. Vinod, S. Sahu
A genetic algorithm for assembly line balancing
Production planning and control, 1996, vol 7, N°1, 38-46

[TEB 01] James Richard Tebboth
A computational study of Dantzig-Wolfe decomposition
Thesis 2001, for the degree of doctor of philosophy in the university of Buckinghanm

[TIM 98] Timothy L. Urban
Optimal Balancing of U-Shaped Assembly Lines
Management Science, May, 1998, Vol. 44, No. 5, pp. 738-741

[TOP 02] Topon Kumar Paul, Hitoshi Iba
Linear and combinatorial optimizations by estimation of distribution algorithms
The 9th MPS symposium on Evolutionary Computation, IPSJ, Japan, 2002

[WEL 86] Wells Devid
The penguin dictionary of curious and interesting numbers
Penguin Books, pp. 61-67, 1986.

[WEN 98] Wen Chyuan Chiang
The application of a tabu search metaheuristic to the assembly line balancing problem
Operations research 1998 (77) 209-227

[YAO 96] Yeo Keun Kim,, Yong Ju Kim, Yeongho Kim
Genetic algorithms for assembly line balancing with various objectives
Computers ind. Engng, 1996, Vol 30, N° 3, 397-409

[YOO 05] Yoo, M. R., Gen, M.
Bicriteria real-time tasks scheduling using genetic algorithm
Complexity International, 2005, Vol.11, Nov.

Annexe 1

Présentation de SA

SA, développé en 1983 par Kirkpatrick, Gelatt et Vecchi [KIR 83], est un méta-algorithme probabiliste générique pour traiter des problèmes d'optimisation fortement non-linéaires ([SPI 00], [SHA 02] et [LOW 05]).

SA s'appuie sur l'algorithme de Metropolis, qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie E du système. Partant d'une solution donnée, en la modifiant, nous en obtenons une seconde solution. Soit celle-ci améliore le critère que nous cherchons à optimiser, nous disons alors que nous avons fait baisser l'énergie du système, soit celle-ci le dégrade. Si nous acceptons une solution améliorant le critère, nous tendons ainsi à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une « mauvaise » solution permet alors d'explorer une plus grande partie de l'espace de solution et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local.

L'algorithme du recuit simulé peut être représenté de la manière suivante:

Algorithme : SA

Fournir une *solution initiale* x (configuration initiale) ;
$x_{opt} = x$ (x_{opt} : la meilleure solution obtenue);
$f_{opt} = f(x_{opt})$ ($f()$: fonction objectif) ;
Fournir une *température initiale* T_0
Tant que (température T > température finale T_f) faire,
    tant que (répétition > max-répétition K constant) faire,
        choisir y dans le voisinage de (x) ;
        calculer $df = f(y) - f(x)$;
        Si $df < 0$ alors
            $x = y$;
            Si $f(x) < f(x_{opt})$ alors
                $x_{opt} = x$;

$f_{opt} = f(x_{opt})$;

Sinon

Générer aléatoirement δ dans [0, 1]

Si $\delta \leq \exp(-\frac{df}{T})$

$x = y$;

$T = g(T)$;

Retourner x_{opt}

SA est basée sur un processus physique qui assure de sortir de l'optimum local pour s'approcher de l'optimum global. L'algorithme de SA est également très flexible parce qu'il ne dépend d'aucune propriété restrictive des modèles. Une autre caractéristique importante de SA est qu'il utilise des représentations simples pour décrire des modèles étudiés.

Annexe 2

La formulation du problème maître

Le problème maître de la génération de colonnes est basé sur la représentation de Minkowski qui peut assurer qu'un polyèdre *P* peut être décrit par la combinaison de ses points extrémaux et de ses rayons extrémaux [TEB 01]. Nous présentons ce théorème dans la suite.

Représentation de Minkowski [TEB 01] *:*

Soit $P = \{ x \in R^n : Ax \bowtie b\}$ n'est pas vide et la dimension de *A* est égale à *n*, alors *P* = *P'*, où

$$P' := \left\{ x \in R^n : x = \sum_{i \in \Pi} \lambda_i x^i + \sum_{j \in R} \mu_j r^j ; \sum_{i \in II} \lambda_i = 1; \mu \geq 0 \right\}$$

$\{ x^i \}_{i \in \Pi}$ et $\{ r^j \}_{j \in R}$ sont les ensembles des points extrémaux et les rayons extrémaux de *P* respectivement, où $\triangleright\triangleleft$ présente trois relations possibles : >, <,=.

Le polyèdre *P* peut être aussi représenté par le polyèdre suivant selon le théorème de Minkowski [TEB 01] :

$$P^{Mk} := \left\{ \lambda \in R^{|E|} : \sum_{i \in E} \delta_i \lambda_i = 1; \lambda \geq 0 \right\}$$

avec la projection :

$$\pi^{Mk} : R^{|E|} \rightarrow R^n ; \lambda \rightarrow x = \sum_i \lambda_i x^i$$

où $E := \Pi \cup R$ et δ_i est constante et égale à 1 si x^i est un point extrême, et égale à 0 si x^i est un rayon extrême.

Pour illustrer le théorème de Minkowski, nous donnons un exemple. Il correspond à un polyèdre P_e de deux dimensions présenté dans la Fig. 1.

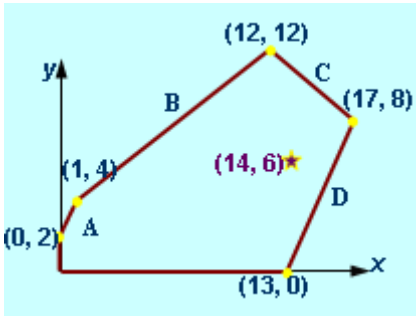


Fig. 1 Exemple de polyèdre

Ce polyèdre peut être décrit par deux variables non-négatives et quatre contraintes de la manière suivante :

$$P_e = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \geq 0 : \begin{matrix} -2x + y \leq 2 & (A) \\ -8x + 11y \leq 36 & (B) \\ 4x + 5y \leq 108 & (C) \\ 2x - y \leq 26 & (D) \end{matrix} \right\}$$

D'après le théorème de Minkowski, le polyèdre peut être représenté par une combinaison convexe de ses points extrêmes et de ses rayons extrêmes. Dans ce cas, comme P_e est borné, il n'y a pas de rayon extrême. Sa représentation de Minkowski consiste à utiliser une variable λ_i pour combiner les six points extrêmes :

$$P_e^{Mk} = \left\{ \lambda \in R^6 : \sum_{i=1}^{6} \lambda_i = 1; \lambda \geq 0 \right\}$$

Avec la projection :

$$\pi_e^{Mk} : R^6 \to R^2 \ ;$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \lambda_1 \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \lambda_2 \begin{pmatrix} 0 \\ 2 \end{pmatrix} + \lambda_3 \begin{pmatrix} 1 \\ 4 \end{pmatrix} + \lambda_4 \begin{pmatrix} 12 \\ 12 \end{pmatrix} + \lambda_5 \begin{pmatrix} 17 \\ 8 \end{pmatrix} + \lambda_6 \begin{pmatrix} 13 \\ 0 \end{pmatrix}.$$

Par exemple, le point (14, 6) peut être représenté par : $\lambda = (0, 0, 0, \frac{8}{28}, \frac{9}{28}, \frac{11}{28})$.

La représentation de Dantzig-Wolfe [DAN 60] est à la base de la transformation du programme linéaire original en un problème maître.

Soit un problème original est un problème de la programmation linéaire :

$$\text{Max}\{cx : Ax \bowtie b,\ x \in R^n_{\diamond}\} \qquad \text{Équation 1}$$

où $\triangleright\triangleleft$ présente trois relaxations possibles : >, <,=.

La matrice *A* des coefficients a une structure de bloc primal diagonal. *P* est la région faisable du problème original (un polyèdre).

$$P = \left\{ \begin{matrix} x_0 \in R^{n_1}_{\diamond_0} \\ x_1 \in R^{n_1}_{\diamond_1} \\ \vdots \\ x_K \in R^{n_K}_{\diamond_K} \end{matrix} : \begin{pmatrix} A_0 & A_1 & \cdots & A_K \\ & B_1 & & \\ & & \ddots & \\ & & & B_K \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_K \end{pmatrix} \begin{matrix} \triangleright\triangleleft_0 \\ \triangleright\triangleleft_1 \\ \vdots \\ \triangleright\triangleleft_K \end{matrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_K \end{pmatrix} \right. \qquad \text{Équation 2}$$

Nous décomposons le polyèdre *P* en plusieurs sous-polyèdres selon les blocs B_k :

$$P_k := \left\{ x_k \in R^{n_k}_{\diamond_k} : B_k x_k \triangleright\triangleleft_k b_k \right\} \text{ avec } k = 1,...,K.$$

Alors nous avons :

$$P = P_0 \cap (R^{n_0} \times P_1 \times \cdots \times P_K)$$

où $P_0 := \left\{ x \in R^n_{\diamond} : (A_0\ A_1\ \cdots A_K)\, x \triangleright\triangleleft_0 b_0 \right\}$.

On représente chaque sous-polyèdre P_k par la représentation de Minkowski :

$$P_k^{Mk} = \left\{ \lambda_k \in R_+^{|E_k|} : \sum_{i_k \in E_k} (\delta_k)_{i_k} (\lambda_k)_{i_k} = 1 \right\}$$

Avec la projection associée :

$$\pi_k^{Mk} : R^{|E_k|} \to R^{n_k};\ \ \lambda_k \mapsto x_k = \sum_{i_k \in E_k} (\lambda_k)_{i_k} x_k^{i_k} \ .$$

Selon la représentation de Minkowski, nous avons $\pi_k^{Mk}(P_k^{Mk}) = P_k$.

Après combinaison avec la contrainte P_0 du problème original, une projection combinatoire de Minkowski est obtenue :

$$\bar{\pi}^{Mk} : R^{n_0} \times R^{|E_1| + \cdots + |E_K|} \to R^n;\ (x_0, \lambda_1, \ldots, \lambda_K) \mapsto (x_0, x_1, \ldots, x_K) \qquad \text{Équation 3}$$

Cette projection combinatoire est donnée d'après les définitions suivantes :

$\overline{P} := R^{n_0} \times P_1 \times \cdots \times P_K$

et $\overline{P}^{Mk} := R^{n_0} \times P_1^{Mk} \times \cdots \times P_K^{Mk}$

Nous pouvons savoir que $\overline{\pi}^{Mk}(\overline{P}^{Mk}) = \overline{P}$ selon le théorème de Minkowski.

Soit $\hat{P}_0$ défini comme la contrainte correspondante à la contrainte P_0 du problème original après la transformation de projection $\overline{\pi}^{Mk}$:

$$\hat{P}_0 := \left\{ (x_0, \lambda) \in R^{n_0}_{\lozenge 0} \times R^{|E_1| + \cdots + |E_K|} : \overline{\pi}^{Mk}(x_0, \lambda) \in P_0 \right\}.$$

La région faisable du problème maître selon Dantzig-Wolfe P^{DW} par la projection $\overline{\pi}^{Mk}$ est définie par :

$$P^{DW} := \hat{P}_0 \cap \overline{P}^{Mk}$$

Équation 4

Représentation de Dantzig-Wolfe :
Soit P défini par l'Équation 1. P peut être reformulé sous condition que rang(B_k) = n_k pour chaque k = 1,..., K, par :

$$\overline{\pi}^{Mk}(P^{DW}) = P$$

où P^{DW} et $\overline{\pi}^{Mk}$ sont définis ci-dessus (voir les Équation 3 et Équation 4).

Pour obtenir la formulation du problème maître, nous devons juste transformer la fonction objectif en s'assurant qu'elle est invariante par rapport à la projection. Dans la fonction objectif, le coefficient de la variable $(\lambda_k)_{ik}$ est $c_k x_k^{ik}$, où x_k^{ik} est un point extrême ou un rayon extrême et c_k est son coefficient. Alors suivant les définitions $(\overline{c}_k)_{ik} := c_k x_k^{ik}$ et $(\overline{A}_k)_{ik} := A_k x_k^{ik}$, le problème maître Dantzig-Wolfe associé au problème original (voir l'Equation 1) peut être présenté par :

$$
\begin{aligned}
&\max c_0 x_0 + \sum_{i_1} \left(\bar{c}_1\right)_{i_1} \left(\lambda_1\right)_{i_1} + \cdots + \sum_{i_K} \left(\bar{c}_K\right)_{i_K} \left(\lambda_K\right)_{i_K} \\
&\text{s.t.} \quad A_0 x_0 + \sum_{i_1} \left(\bar{A}_1\right)_{i_1} \left(\lambda_1\right)_{i_1} + \cdots + \sum_{i_K} \left(\bar{A}_K\right)_{i_K} \left(\lambda_K\right)_{i_K} \triangleright\triangleleft_0 \; b_0 \\
&\qquad\qquad \sum_{i_1} \left(\delta_1\right)_{i_1} \left(\lambda_1\right)_{i_1} \qquad\qquad = \quad 1 \\
&\qquad\qquad\qquad\qquad \ddots \qquad\qquad \vdots \quad \vdots \\
&\qquad\qquad\qquad\qquad \sum_{i_K} \left(\delta_K\right)_{i_K} \left(\lambda_K\right)_{i_K} \quad = \quad 1 \\
&x_0 \in R^{n_0}_{\Diamond_0} : \lambda_1 \in R_+^{\left|E_1\right|}, \ldots, \lambda_K \in R_+^{\left|E_K\right|}
\end{aligned}
$$

Équation 5

La forme la plus simple est présentée par :

$$
\left\{
\begin{array}{l}
\max \quad c_0 x_0 + \bar{c}\lambda \\
S.C. \quad A_0 x_0 + \bar{A}\lambda \triangleright\triangleleft_0 \; b_0 \\
\qquad\qquad \Delta\lambda = 1 \\
x_0 \in R^{n_0}_{\Diamond_0}; \quad \lambda \in R_+^{\left|E_!\right| + \ldots + \left|E_K\right|}
\end{array}
\right\}
$$

Équation 6

où $\Delta := diag(\delta_1, ..., \delta_K)$.

Algorithme de la génération de colonnes

Algorithme : La génération de colonnes

Etape 1 : Initialisation

Sélectionner $E'_k \subseteq E_k$, $k = 1, \ldots, K.$ et construire le problème maître restreint.

Etape 2 : Résoudre le problème maître restreint :

$$Z = \left\{ \begin{array}{l} \max \ \ c_0 x_0 + \bar{c}' \lambda' \\ S.C. \ A_0 x_0 + \bar{A}' \lambda' \triangleright\triangleleft_0 \ b_0 \\ \qquad\qquad \Delta' \lambda' = 1 \\ x_0 \in R^{n_0}_{\Diamond_0} ; \lambda' \in R_+^{\left|E'_1\right|+\cdots+\left|E''_K\right|} \end{array} \right\},$$

et déterminer les valeurs duales (π, μ).

Etape 3 : Résoudre le problème de génération de colonnes :

$$Z_k^* = \left\{ \begin{array}{l} \max \ (c_k - \pi * A_k) x_k - \mu \\ S.C. \ B_k x_k \triangleright\triangleleft_k \ b_k \\ \\ x_k \in R^{n_k}_{\Diamond_k} \end{array} \right\}.$$

Si la solution optimale du problème de génération de colonnes Z_k* est positive, les variables correspondantes λ_k du master problème doivent entrer dans la base du problème maître restreint courant. Retourner à l'étape 2.

Sinon il n'y a pas de variables qui peuvent améliorer la solution optimale trouvée pour le problème maître. La solution courante du problème maître restreint est la solution optimale du problème maître. Le processus est donc arrêté.

La génération de colonnes est une méthode de Simplexe spécialisée basée sur les itérations qui consiste à résoudre un problème de programmation linéaire de grande taille. D'abord, le problème original est transféré à un problème maître qui emploie moins de contraintes et plus de variables que le problème original. Dans chaque itération, la génération de colonnes résout un problème maître restreint qui prend en compte un petit nombre de variables du problème maître. Les variables restantes sont efficacement fixées à 0. Une fois que le problème maître restreint est résolu, un problème de génération de colonnes est construit. Alors les variables qui doivent entrer dans le problème maître restreint dans la prochaine itération sont déterminées par le problème de génération de colonnes. Ce processus est répété jusqu'à ce qu'il n'y a plus de variable intéressante à faire entrer dans le problème maître restreint.

Résumé : Ce travail traite du problème de l'équilibrage de lignes d'assemblage. C'est un problème d'optimisation combinatoire qui permet de définir la répartition des opérations et leur affectation aux stations de la ligne d'assemblage tout en respectant différentes contraintes de façon à optimiser un critère d'efficacité donné. Dans nos travaux, nous ne considérons que le problème de type II (SALBP-2) qui consiste à minimiser le temps de cycle déterminé par le temps de station le plus grand avec un nombre donné de stations. Dans un premier temps, nous nous intéressons au SALBP-2 avec des temps opératoires déterministes. Pour ces problèmes, nous proposons une heuristique et deux méta-heuristiques : l'algorithme basé sur le mécanisme d'électromagnétisme (EM) et l'estimation de distribution (ED) que nous adaptons à notre problème. Les résultats de simulation sont comparés avec ceux du recuit simulé (SA). Ensuite, nous cherchons à résoudre les SALBP-2 avec des temps opératoires stochastiques. De part la meilleure performance de EM pour le problème déterministe, EM est choisie pour équilibrer les lignes stochastiques. Dans le premier cas, les temps de cycle sont minimisés de façon à assurer que la fiabilité de la ligne soit supérieure à une valeur donnée. Dans le deuxième cas, EM est utilisée pour définir un ensemble de solutions Pareto-optimales pour des problèmes de type multi-critère avec minimisation du temps de cycle et maximisation de la fiabilité de la ligne. Enfin, nous proposons une méthode basée sur la génération de colonnes pour déterminer une bonne borne inférieure du temps de cycle.

Mots clés : équilibrage de lignes d'assemblage, heuristiques, méta-heuristiques, algorithme basé sur le mécanisme d'électromagnétisme (EM), estimation de distribution (ED), génération de colonnes, optimisation du temps de cycle.

Abstract: The work presented in this PhD report deals with the assembly line balancing type II problem (SALBP-2). The aim is to minimize the cycle time given by the largest time station under the constraints of a given number of stations and the precedence of the operations. At first, we consider the SALBP-2 with determinist operation times. Then, we define a new heuristic and we adapt two meta-heuristics: electromagnetism-like mechanism algorithm (EM) and estimation of distribution (ED) to SALBP-2. The simulated results are compared with simulated annealing (SA). Thereafter, we solve the SALBP-2 with stochastic operations times. In this case, we search for a set of Pareto optimal solutions. Due to its better performance, EM is chosen to balance the stochastic lines with random operation times. As the performance of an heuristic or of a meta-heuristic usually depends on the initial solution we also propose a column generation based method to determine a good lower bound of the cycle time.

Keywords: assembly line balancing problem, heuristic and meta-heuristic, electromagnetism-like mechanism algorithm (EM), estimation of distribution (ED), column generation, cycle time optimization.