Méthodes hybridées pour l'équilibrage de charges de lignes d'assemblage robotisées

Slim Daoud, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, Philippe Duperray

9e Conférence Internationale de Modélisation, Optimisation et SIMulation - MOSIM'12
06 au 08 Juin 2012 - Bordeaux - France
« Performance, interopérabilité et sécurité pour le développement durable »

METHODES HYBRIDEES POUR L'EQUILIBRAGE DE CHARGES DE LIGNES D'ASSEMBLAGE ROBOTISEES

Slim DAOUD, Philippe Duperray

ARIES Packaging, Technopole de l'Aube en Champagne, 10430 Rosières, France
{slim.daoud, philippe.duperray}@mwv.com

Farouk YALAOUI, Lionel AMODEO, Hicham CHEHADE

Institut Charles Delaunay (STMR UMR CNRS 6279), LOSI, Université de Technologie de Troyes, 12 rue de Marie Curie, 10010 Troyes, France
{farouk.yalaoui,lionel.amodeo,hicham.chehade}@utt.fr

RESUME : *Cet article porte sur l'optimisation de lignes d'assemblage robotisées équipées de robots de prise et dépose. De ce fait, les robots sont utilisés pour saisir les composants défilant sur le convoyeur de prise et les positionner dans les positions du produit final sur le convoyeur de dépose. Ainsi, le problème d'optimisation consiste à améliorer les performances de ces lignes et d'assurer la répartition et l'équilibrage de charges des lignes robotisées en respectant des contraintes technologiques. Pour répondre à un besoin industriel, on propose des méthodes d'optimisation couplées avec des automates programmables. Ce type de couplage nécessite un temps de réponse très court. De ce fait, on propose des métaheuristiques pour résoudre ce problème, comme des colonies de fourmis, un algorithme par essaim particulaire et un algorithme génétique. Ces algorithmes sont hybridés avec une recherche locale guidée. Les performances des méthodes développées ont été évaluées par comparaison avec des solutions optimales obtenues à l'aide du modeleur Cplex et la méthode d'énumération complète.*

MOTS-CLES : *Equilibrage de charges, Optimisation, métaheuristiques, recherche locale guidée.*

1 INTRODUCTION

Le problème étudié dans cet article apparaît à l'étape de la conception ou à l'étape de reconfiguration de lignes d'assemblage robotisées afin d'optimiser leurs performances et de répondre aux besoins du marché qui demande des lignes innovantes et à haute cadence pour faire face à la concurrence. En effet, les robots de prise et de dépose présentent un rôle important dans les lignes de production et en particulier dans les lignes d'assemblage. Cela est dû aux avantages qu'ils présentent notamment au niveau de la productivité. Dans ce travail, nous nous intéressons au problème d'équilibrage de lignes d'assemblage robotisées afin d'améliorer et d'optimiser leurs performances.

(Rubinovitz et Bukchin, 1991) sont les premiers qui ont formulé le problème d'équilibrage de lignes robotisées (RALB) en cherchant à équilibrer les stations de la ligne, tout en attribuant le robot le plus efficace et disponible aux stations. Ils ont proposé une méthode basée sur la procédure de séparation et évaluation pour équilibrer la charge pour différents robots. (Levitin et *al.*, 2006) ont proposé un algorithme génétique pour l'équilibrage de lignes d'assemblage robotisées de type II afin de trouver les meilleures combinaisons des tâches et robots à affecter pour chaque station. En général, l'objectif du problème d'équilibrage de lignes d'assemblage est l'affectation de tâches aux différentes stations de la ligne de manière à respecter les contraintes de précédence avec une fonction objectif à optimiser. Le terme Simple Assembly Line Balancing Problem (SALBP) a été introduit par (Baybars, 1986). L'objectif de ce problème est d'équilibrer une ligne d'assemblage fabriquant un seul type de produit. (Boysen *et al.*, 2008) ont présenté les différents types de SALBP qui peuvent être de type 1, type 2, type E ou de type F. SALBP de type 1 consiste à minimiser le nombre de stations en gardant le temps de cycle inversement à celui du type 2. SALBP-F consiste à étudier la faisabilité de l'opération à affecter pour un temps de cycle donné et un nombre de stations fixe. Concernant le SALBP-E,qui est similaire à notre problème, (Nai-Chieh et Chao, 2011) ont développé une méthode qui permet de résoudre ce type de problèmes tout en combinant les deux types SALBP-1 et SALBP-2. (Dolgui et Ihnatsenka, 2009) ont traité de nouvelles classes de problèmes d'équilibrage de lignes de transfert (TLBP). L'objectif de ce problème est de trouver la meilleure répartition des tâches par bloc et station afin de minimiser le coût de la ligne. Une série de publications est consacrée à l'optimisation ce type de ligne comme (Guschinskaya et Dolgui, 2009) où ils ont comparé la performance d'une méthode exacte et d'une autre approchée.

La problématique traitée dans ce travail est issue d'un cas industriel. Pour la résoudre, nous proposons plusieurs méthodes hybridées avec une recherche locale guidée qui seront couplées avec des automates programmables (PLC). Ces algorithmes permettent d'affecter les tâches et les composants à chaque robot afin de maximiser l'efficacité des lignes robotisées et de définir les politiques de saisie des composants. Nous

comparons les performances des méthodes développées avec les solutions exactes obtenues à l'aide du modeleur Cplex et la méthode d'énumération complète.

La première méthode développée pour ce type de problèmes qui est basée sur les colonies de fourmis cette méthode a été proposée par (Dorigo, 1992). Cette métaheuristique est basée sur le comportement de fourmis dont le déplacement représente une solution partielle. Le mouvement des fourmis est basé sur deux paramètres qui sont le taux de phéromones et la désirabilité. (Ze-qiang *et al.*, 2007) ont amélioré pour résoudre un problème d'équilibrage de lignes d'assemblages robotisés de type I. (Chehade *et al.*,2009) ont utilisé les colonies de fourmis pour les problèmes de dimensionnement des buffers.

La deuxième méthode abordée dans cet article est l'algorithme génétique. Ce dernier a été défini par (Hollande, 1975) et fait partie de la famille des algorithmes évolutionnaires inspirés par l'évolution biologique des espèces. (Gao *et al.*, 2009) ont développé un algorithme hybridé avec une recherche locale afin d'affecter les robots aux différentes stations. Le dernier algorithme est basé sur l'essaim particulaire. L'algorithme d'essaim particulaire est une technique d'optimisation basée sur le concept d'intelligence par essaim. Initiée par (Eberhart et Kennedy, 1995), cette méthode est inspirée du comportement des essaims d'abeilles ou des oiseaux. (Dong yun *et al.*, 2010) ont développé un nouvel algorithme qui change dynamiquement l'inertie et le poids du PSO. Nous couplons alors les algorithmes développés avec une recherche locale guidée (GLS). (Voudouris et Tsang, 1996) ont été les premiers à introduire cette technique d'optimisation générale utilisable à un grand nombre de problèmes d'optimisation combinatoire. La recherche locale guidée est utilisée pour éviter les optimums locaux. Elle a été appliquée avec succès à plusieurs problèmes pratiques comme les tournées de véhicule ou le problème d'affectation quadratique tels que développé par (Mils *et al.*, 2003).

Nous rappelons que le problème étudié dans cet article consiste à affecter tous les composants dans le produit final par l'intermédiaire des robots de prise et dépose de façon à maximiser l'efficacité des lignes d'assemblage robotisées RALB. La majorité des travaux de recherche concernant l'équilibrage de lignes d'assemblage robotisées se résument sur ceux du RALB type I et RALB type II où l'objectif est de minimiser le temps de cycle ou le nombre de station. De ce fait, notre contribution est de proposer une application sur un cas industriel par l'intermédiaire d'un couplage entre les métaheuristiques hybridées et des automates programmables. Ce couplage nécessite un temps de réponse qui est suffisamment court. De plus, cette méthode de recherche locale (GLS), à notre connaissance, n'a été jamais utilisée pour l'équilibrage de lignes robotisées (RALB).

Ce papier est organisé comme suit : la deuxième partie présente la formulation du problème d'optimisation. Les méthodes de résolution sont présentées dans la troisième partie. Les réglages des paramètres et les tests numériques sont développés en quatrième partie afin de terminer par une conclusion et des perspectives.

2 FORMULATION DU PROBLEME D'OPTIMISATION

L'objectif de notre travail est de résoudre un problème de configuration et de reconfiguration de lignes robotisées afin d'améliorer la performance de ces lignes et d'équilibrer la charge des différents robots. Cette ligne d'assemblage robotisée, modélisée par la figure 1, est constituée de I robots de prise et dépose. Le rôle de ces robots est de saisir les J composants de produits et les déposer sur un deuxième convoyeur pour assembler le produit final constitué de K positions. Ce dernier est composé de n couches qui contiennent chacune K_α composants ou emplacements. On note aussi k_α la position du composant dans la couche α du produit final. Concernant les deux convoyeurs qui sont présentés par la figure 1, le premier est appelé convoyeur de prise. Sur ce convoyeur, destiné aux transports des composants du produit, les composants défilent sur plusieurs voies (bandes). Le deuxième convoyeur, appelée convoyeur de dépose, est utilisé par les robots pour déposer les composants dans les emplacements du produit final. On note aussi que dans la suite de l'article le mouvement de prise et dépose des produits par les robots est appelé tâche d'assemblage. Par ailleurs, l'assemblage du produit final requiert l'exécution de K tâches. L'ordre dans lequel les tâches doivent être exécutées est assuré par les contraintes de précédence qui sont représentées par un graphe cyclique de précédence comme l'illustre la figure 2. Dans ce dernier, toutes les tâches sont présentées comme des nœuds. Les contraintes de précédence traduisent le fait qu'une position d'une couche ne peut être complétée que si la même position dans la couche précédente est complétée. La ligne considérée dans cet article est constituée de I stations de travail en série, chacune équipée d'un robot.

Afin de pouvoir formuler le problème d'optimisation qui vise à maximiser la productivité de la ligne d'assemblage robotisée, nous avons émis les hypothèses qui sont basées sur celles développées par (Rubinovitz et Buckchin, 1991) :

(1) La durée de la tâche est déterministe et ne peut pas être subdivisée

(2) La durée d'une tâche dépend du robot et de l'emplacement affecté

(3) L'affectation des tâches pour chaque robot est déduite par la capacité du robot et la cadence élémentaire de la ligne

(4) Un seul robot est affecté à chaque station

(5) La ligne est équilibrée pour un seul type de produit

(6) Les relations de précédences sont imposées par les contraintes technologiques et industrielles.

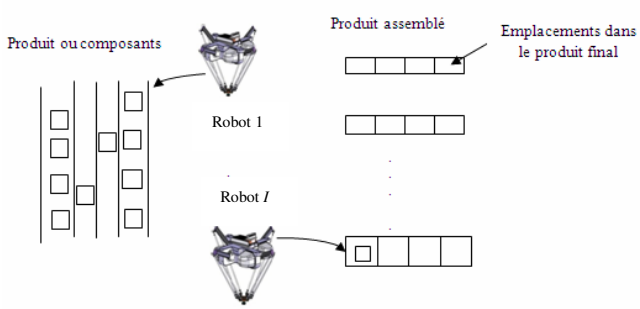

Figure 1 : Description de la ligne d'assemblage robotisée

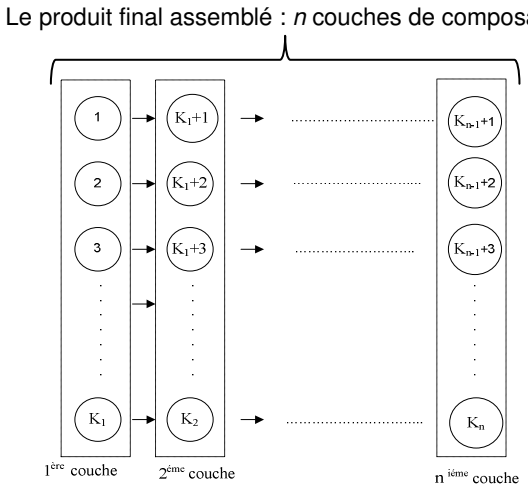

Figure 2 : Graphe de précédence entre les positions du produit final

Comme défini dans l'introduction, plusieurs travaux ont été réalisés pour la problématique de l'équilibrage des lignes robotisées. Plusieurs modèles ont été développés essentiellement pour RALB-I et RALB-II. L'analyse de l'état de l'art a permis aussi de noter qu'il n'existe pas de travaux dédiés à notre problème concret qui est RALB-E. Un modèle mathématique a été proposé dans nos travaux antérieurs (Daoud *et al.*, 2012) et qui est basé sur la programmation linéaire.

3 METHODES DE RESOLUTION

Afin de répondre à un besoin industriel qui nécessite l'optimisation de la configuration des lignes d'assemblage robotisées de type E, on a proposé un couplage des méthodes d'optimisation avec des automates programmables (PLC). De ce fait, dans cette section, on présente les méthodes proposées pour résoudre le problème d'optimisation hybridées avec une recherche locale guidée à savoir : les colonies de fourmis, l'algorithme génétique et l'optimisation par essaim particulaire. Le choix de ces méthodes s'est fait par rapport à leur aptitude à résoudre des problèmes en temps court ainsi que leur capacité de résolution des problèmes de grandes tailles. En outre, on présente des méthodes exactes pour évaluer les méthodes proposées. Concernant le critère d'arrêt, deux types de critères d'arrêt existent dans la littérature : soit le nombre de générations est fixé ou le temps CPU, soit le processus de recherche s'arrête lorsque la meilleure fonction objectif n'est pas améliorée pour un nombre donné de générations. Dans cette étude, afin de donner des chances égales à toutes les méthodes, le critère de temps CPU est retenu.

3.1 Méthodes approchées

Cette section présente les méthodes approchées proposées (ACO, PSO et GA) couplés avec les automates programmables (PLC) pour résoudre le problème d'équilibrage de lignes robotisées de type E issu d'un besoin industriel. Ensuite, afin d'améliorer la performance de ces méthodes, nous les avons hybridées avec une recherche locale guidée afin d'éviter les minimums locaux.

Cette méthode comme expliquée dans (Hani *et al.*,2006) est basée sur l'augmentation des pénalités à la fonction objectif modifiée. Ces pénalités permettent d'éviter les minimums locaux et donc augmente les chances de trouver l'optimum global. A notre connaissance, cette méthode de recherche locale n'a été jamais utilisée pour les problèmes d'équilibrage de lignes robotisées. De plus, les avantages qu'elle propose au niveau de la rapidité de son temps d'exécution et l'efficacité des solutions proposées.

3.1.1 L'algorithme de colonies de fourmis (ACO)

Dans cette section, nous décrivons l'algorithme de colonie de fourmis pour résoudre le problème d'équilibrage de lignes d'assemblage robotisées.

Codage

Afin d'appliquer l'algorithme de colonies de fourmis, nous avons adopté le codage présenté dans la figure 3 pour les lignes d'assemblage robotisées avec *KM* points de dépose à affecter pour *J* composants à travers les *I* robots.

Construction de la solution

Des fourmis parallèles sont utilisées pour l'affectation des composants et des robots pour les points de dépose du produit final. Dans un premier temps, chaque fourmi est déposée aléatoirement sur un point de départ correspondant aux composants et robots à attribuer au premier point de dépose du produit final. Ensuite, chaque fourmi construit son chemin en choisissant le point à visiter à chaque étape. Chaque étape correspond à un robot et un composant à affecter. Le nombre total de points pouvant être visités à chaque étape correspond aux différentes solutions de notre problème. Le choix des points à visiter est basé sur l'application d'une règle de transition d'état.

Point de dépose 1	Numéro du composant	1	2	…	*J*
	Numéro du robot	1	2	…	*I*
Point de dépose 2	Numéro du composant	1	2	…	*J*
	Numéro du robot	1	2	…	*I*
. . .	. . .	. . .	. . .	. . .	. . .
Point de dépose *KM*	Numéro du composant	1	2	…	*J*
	Numéro du robot	1	2	…	*I*

Figure 3 : Codage appliqué pour l'algorithme de colonies de fourmis

Une fourmi *f* étant sur un point *r* choisit le prochain point *p* à visiter en se basant sur l'équation (1).

$$p=\begin{cases} \arg\max_{u\in j_k(r)} \left\{[\tau_{r,u}]^{\alpha}.[\eta_{r,u}]^{\beta}\right\} & si \quad q \le q_0 \\ P^* & \text{Sinon} \end{cases} \quad (1)$$

Dans l'équation (1), *q* est une valeur aléatoire générée entre 0-1, q_0 est un paramètre ($0 \le \rho \le 1$) qui donne plus d'importance soit à l'intensification soit à la diversification. *P** est une variable aléatoire sélectionnée selon une probabilité donnée par l'équation (2). $\tau^0_{r,t}$ est la quantité initiale de phéromone entre *r* et *p*. Les paramètres *α, β* sont utilisés pour donner plus d'importance à la quantité de phéromone. $J_k(r)$ est l'ensemble des points non encore visités par la fourmi *f*.

$$P^*=\begin{cases} \dfrac{[\tau_{r,s}]^{\alpha}.[\eta_{r,s}]^{\beta}}{\sum_{u\in J_k(r)} [\tau_{r,s}]^{\alpha}[\eta_{r,u}]^{\beta}} & si \quad t \in J_k(r) \\ 0 & \text{Sinon} \end{cases} \quad (2)$$

Mise à jour locale et globale

En construisant son chemin, une fourmi change la quantité de phéromones en se basant sur l'équation (3) avec ρ le taux d'évaporation de phéromones $0 \le \rho \le 1$ et τ_0 est la quantité initiale de phéromone.

$$\tau_{r,t}= (1-\rho)\,\tau_{r,t}+ \rho\,\tau_0 \quad (3)$$

Une fois que les fourmis ont fini leurs tours, la quantité de phéromone est améliorée par l'intermédiaire de l'équation (4).

$$\tau_{r,t}= (1-\rho)\,\tau_{r,t}+ \rho\,\Delta\tau_{r,t} \quad (4)$$

Où $\Delta\tau_{r,s}$ est un facteur qui consiste à favoriser les meilleures solutions. Il est calculé selon l'équation (5).

$$\Delta\tau_{r,t} = \tau^0_{r,t} + X\left(1-\left(\frac{L_{gb}}{U_{gw}}\right)\right) \quad (5)$$

Où $\tau^0_{r,t}$ est la valeur initiale de phéromone, *X* est une constante, L_{gb} est la meilleure solution globale pour une itération et U_{gw} est la mauvaise solution proposée pour une itération.

3.1.2 L'algorithme d'Essaim particulaire (PSO)

Dans cette section, nous décrivons l'algorithme d'essaim particulaire (PSO) pour résoudre notre problème. Comme nous l'avons déjà mentionné dans l'introduction, cet algorithme consiste à refléter le déplacement et le comportement des essaims particulaires. L'algorithme commence par l'initialisation aléatoire des positions et des vitesses des particules. Ensuite, les particules évoluent vers la solution optimales à l'aide de P_{best} et G_{best}. La différence entre P_{best} et G_{best} est que G_{best} est utilisé sur toutes les particules et P_{best} est utilisé pour la particule elle-même.

Codage

Afin d'appliquer l'algorithme PSO, nous avons adopté le codage présenté dans la figure 4. Comme le montre la figure, la particule P_e est composée de différent point de dépose, composant et robot. A chaque élément de cette particule, on affecte un robot *i* tels que *i*=1..*I* et composant (*j*=1,…*J*) tout en respectant les contraintes technologiques et de précédence.

Particule *Pe*	Point de dépose	1	2	…	*KM*	Fitness
	Numéro du robot	1…*I*	1...*I*	…	1...*I*	
	Numéro du composant	1…*J*	1…*J*	…	1…*J*	

Figure 4 : codage appliqué pour l'optimisation par essaim particulaires

Construction de la solution

Une particule dans la population représente une solution relative au nombre de produits saisis par les différents robots. Chaque particule est composée de trois indices qui reflètent le mouvement du robot *i* du composant *j* vers le point de dépose *k* du produit final *m*. De ce fait, la taille de la particule est la somme de tous les points de dépose qui est égale à *KM*. La solution est obtenue principalement à l'aide de la position (x_{ii}^{kk}), qui est le robot et composant pour chaque point de dépose. La vitesse maximale (DomaineLimit) est égale au nombre de robots *I*. La valeur de la vitesse représente les mouvements qui se produisent pour chaque particule pour changer le numéro du robot et composant affecté pour les différents points de dépose. Cela se fait suivant l'équation (6). La mise à jour des positions se fait selon l'équation (7).

$$V_d^{e+1} = (w*V_d^e + C_p*(RND)*(Pbest_d - x_d^e) + C_g*(Rnd)*(Gbest_d - x_d^e) \bmod ulo(DomaineLimit) \quad (6)$$

$$x_d^{e+1} = (round(x_d^e + V_d^{e+1}) \bmod ulo(DomaineLimit) \quad (7)$$

V_d^e la vitesse de l'individu *d* dans *l'*itération *e*
X_d^e la position actuelle de l'individu *d* dans *l'* itération *e*
w l'inertie
C_{ii} les paramètres d'importances
Rnd nombre aléatoire
$Pbest_d$ la meilleure position d'une particule
$Gbest_d$ la meilleure position globale d'une particule
DomaineLimit la vitesse maximale qui est égale aux nombres de robots

3.1.3 L'algorithme génétique (GA)

Dans cette section, on décrit l'algorithme génétique pour résoudre le problème d'équilibrage de lignes d'assemblages robotisées de type E.

Codage

Nous avons adopté le même codage appliqué pour l'algorithme essaim particulaire qui présenté dans la figure 4. Chaque chromosome dans la population définit une solution pour le problème. Il est composé d'une matrice comportant *KM* gènes. Chaque gènes contient trois entier qui représente respectivement l'indice du point de dépose, du robot (1,…, *I*) et du composant (1,..., *J*). Ceci reflète le mouvement de prise et dépose de chaque composant vers les points de dépose par l'intermédiaire d'un robot.

Fonction objectif

La valeur fournie par la fonction objectif qui vise à maximiser le nombre de produits pris par chaque robot.

Reproduction

La reproduction consiste à obtenir deux enfants en échangeant un ou plusieurs gènes de leurs parents. Nous avons adopté un test sur les chromosomes afin d'assurer la faisabilité des solutions. En se basant sur des travaux antérieurs, nous adaptons pour ce problème le croisement en un seul point vu qu'un point de croisement est généré aléatoirement sur les point de dépose présentés en gras. Ensuite les composants et robot sont réattribués aux points de dépose pour chaque produit final. Un exemple de croisement entre deux parents et les nouveaux chromosomes (enfants) est présenté dans figure 5 et figure 6.

Parent 1	**1**	**2**	**3**	**4**	**5**
	1	1	2	2	3
	1	3	4	2	5
Parent 2	**1**	**2**	**3**	**4**	**5**
	1	1	2	2	3
	1	3	4	2	5

Figure 5 : Exemple de parents sélectionnés

Figure 6 : Enfants : nouveau chromosome

Enfant 1	**1**	2	3	4	5
	1	1	2	2	3
	1	3	4	2	5
Enfant 2	**1**	2	3	4	5
	1	1	2	2	3
	1	3	4	2	5

Mutation

La mutation est utilisée pour diversifier les solutions. Nous avons choisit deux gènes aléatoirement et on échange leurs affectations. Un test de vérification de contraintes technologique et industriel suit chaque opération de mutation afin d'avoir des solutions cohérentes.

3.1.4 La recherche locale guidée

Afin d'améliorer la performance de ces algorithmes, nous avons hybridé ces derniers avec une recherche locale guidé qui permet d'éviter les minimums locaux. Nous avons opté pour cette méthode de recherche locale vu les avantages qu'elle propose au niveau de la rapidité de son temps d'exécution et l'efficacité des solutions proposées. La recherche locale guidée est considérée comme une amélioration des solutions en évitant les minimums locaux. Cette méthode s'opère en modifiant la fonction objectif et en intégrant des pénalités associées à des optimums locaux (Voudouris et Tsang, 1996). Chaque caractéristique de la solution f_t doit avoir les composants suivants :

- Une fonction d'indication $I_t(l)$ pour préciser si la caractéristique est présente dans la solution actuelle ($I_t(l)$ =1) ou pas ($I_t(l)$=0).
- Une fonction coût $c_t(l)$ qui donne le coût d'avoir f_t dans l qui est donc le coût de déplacer un composant j par le robot i vers la position k du produit final m.
- Une pénalité p_t initialisée à 0, utilisée pour pénaliser f_t puisque f_t est une caractéristique de la solution l

$$util(s, f_t) = I_t(l)\frac{c_t(l)}{1+p_t} \tag{8}$$

La variation de la fonction objectif est due à la modification de la pénalité. La recherche locale guidée vise à incrémenter les paramètres de pénalité p_t pour les caractéristiques qui maximisent la fonction utilitaire définie dans l'équation (8). De ce fait, l'objectif est de pénaliser les caractéristiques ayant des coûts élevés. Par l'intermédiaire d'une fonction objectif augmentée, la recherche locale guidée permet de sortir de l'optimum local en rendant le coût de la solution plus cher. Cette fonction objectif augmentée est définie par :

$$h(l) = g(l) + \lambda' \sum_{t=1}^{n} I_t(l)\, p_t \tag{9}$$

Où $g(l)$ représente la fonction objectif du problème et λ' est un paramètre utilisé pour contrôler les contraintes par rapport à la solution actuelle. L'application de la recherche locale guidée au problème d'équilibrage des lignes d'assemblage robotisées se fait en considérant les analogies suivantes : la caractéristique f_t d'une solution s correspond à l'affectation du composant j au robot i pour l'emplacement k du produit final m. Le coût relatif à la caractéristique est la charge du robot i. Ce coût est calculé dans l'équation 10 :

$$c(i) = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{m=1}^{M} t_{ijkm}\, x_{ijkm} \tag{10}$$

La valeur de λ' adaptée à notre problème est définit dans l'équation 11 où J est le nombre de composants et K est le nombre de positions dans le produit final :

$$\lambda' = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{m=1}^{M} x_{ijkm}}{2\max(J,K)} \tag{11}$$

L'algorithme de recherche locale guidée est résumé dans l'algorithme 1.

Algorithme 1 : Recherche locale guidée :

Calculer λ

Considérer la solution initiale l comme la meilleure l'

Lancer *la recherche locale en utilisant les fonctions objective modifiées*

***Considérer** la solution l* comme étant la solution avec le coût modifié le moins élevé*
Si le coût (l) < coût (l') alors remplacer l' par l**
Fin si
***Trouver** la caractéristique de l* ayant la fonction utilitaire maximale et incrémenter la pénalité $p_t=p_t+1$*
***Retour** à l'étape de recherche locale jusqu'à la satisfaction du critère d'arrêt.*
l' est la meilleure solution du problème.

3.1.5 L'algorithme de colonies de fourmis hybridé avec une recherche locale guidée (ACO-GLS)

Dans cette section, nous présentons l'algorithme de fourmis hybridé avec la recherche locale guidée (ACO-GLS). L'algorithme de colonies de fourmis couplé avec la recherche locale guidée, est donné dans l'algorithme 2.

Algorithme 2 : Les colonies de fourmis couplés avec la recherche locale guidée

Initialisation des paramètres
Pour les N_f fourmis faire
***Placer** aléatoirement les N_f fourmis sur les points représentant les robots et composants affectés pour le premier point de dépose k=1, j=1*
Pour k = 2 jusqu'au le nombre de point de dépose
Pour j= 2 jusqu'au le nombre de composants
***Choisir** les prochains points à visiter (robot et composant à affecter au point de dépose) en se basant sur les équations (13) et (14)*
***Vérifier** si point de dépose[k] = Robot i alors point de dépose [k+1] = {Robot i || Robot i+1}*
Sinon mettre à jour la valeur de point de dépose [k+1]= {Robot i || Robot i+1}
Fin pour
Fin pour
***Appliquer** les mises à jour locales des phéromones*
***Appliquer** la recherche locale guidée GLS*
***Appliquer** les mises à jour globales des phéromones en se basant sur la meilleure solution trouvée*
Fin pour
***Reconstruction** de nouveaux tours des fourmis jusqu'à la satisfaction d'un critère d'arrêt (temps de CPU)*

3.1.6 Essaim particulaire hybridé avec une recherche locale guidée (PSO-GLS)

Dans cette section, nous présentons l'algorithme d'essaim particulaire hybridé avec la recherche locale guidée (PSO-GLS). L'algorithme d'optimisation par essaim de particulaire couplé avec la recherche locale guidée, est donné dans l'algorithme 3.

Algorithme 3 : essaim de particulaire couplé avec la recherche locale guidée

Générer les particuliers aléatoirement
Générer x_d^1 : solution initiale
Générer V_d^1 : vitesse initiale
***Vérifier** si point de dépose[k] = Robot i alors point de dépose [k+1] = {Robot i || Robot i+1}*
*Sinon **mettre à jour** la valeur du point de dépose [k+1]= {Robot i || Robot i+1}*
***Affecter** à Pbest de chaque particule la solution initiale*
Trouver la meilleure solution G_{best} parmi toutes les solutions
***Tant que** (le temps CPU < temps nécessaire pour avoir la solution)*
***Mettre à jour** la vitesse*
***Mettre à jour** les nouvelles positions des particules*
***Mettre à jour** P_{best}*
Appliquer la recherche locale guidée
Si** G_{best} est amélioré **alors** mettre à jour **G_{best}
Fin Si
Fin tant que
Retourner la solution de G_{best}

3.1.7 L'algorithme génétique hybridé avec une recherche locale guidée (GA-GLS)

Dans cette section, nous présentons l'algorithme génétique hybridé avec la recherche locale guidée (GA-GLS). L'algorithme génétique couplé avec la recherche locale guidée, est donné dans l'algorithme 4.

Algorithme 4 : L'algorithme génétique couplé avec la recherche locale guidée

Générer aléatoirement une population de Ng chromosomes
***Vérifier** si point de dépose[k] = Robot i alors point de dépose [k+1] = {Robot i || Robot i+1}*
*Sinon **mettre à jour** la valeur du point de dépose [k+1]= {Robot i || Robot i+1}*
Pour** chaque itération **faire
***Tant que** (le temps CPU < temps nécessaire pour avoir la solution)*
Sélectionner aléatoirement les chromosomes
Appliquer un croisement orienté avec une probabilité P_c
Appliquer la mutation sélectionnée avec une probabilité P_m
Appliquer la recherche locale guidée
Insérer les nouveaux enfants avec leurs parents
Trier en décroissance les chromosomes selon leurs fitness
Fin tant que
Afficher les meilleures solutions si le critère d'arrêt est d'atteint
Fin pour.

3.2 Méthodes exactes

Afin d'obtenir une solution exacte à notre problème, nous avons utilisé le modèle developé par (Daoud *et al.*, 2012) qui est basé sur un modèle linéaire. Cette solution permet d'évaluer les solutions proposées par les méthodes développées pour résoudre le besoin industriel. Pour sa résolution, nous avons utilisé le solveur Cplex. Une autre méthode exacte basée sur l'énumération complète des solutions a été implémentée (FEM). Cette méthode parcourt tout l'espace de recherche et teste la qualité de toutes les solutions possibles puis choisit la meilleure solution. La FEM permet de confirmer les solutions proposées par Cplex.

4 REGLAGES DES PARAMETRES

Le réglage des paramètres a été déduit en effectuant plusieurs tests afin de dégager les meilleures valeurs des paramètres pour chaque algorithme appliqué.

4.1 Paramètres de l'ACO

Vu la performance de ses résultats obtenus par (Chehade *et al.*, 2008), nous adoptons ici les paramètres qu'ils ont utilisés. Ces paramètres sont : le nombre de fourmis N_f=20, le taux d'évaporation ρ= 0.6, q_0=0.75, α = 0.7, β = 0.3 et enfin la constante X = 7.

4.2 Paramètres du PSO

En se basant sur des récents travaux dans le domaine d'optimisation par essaim particulaire, nous adoptons les paramètres du PSO développé par (Daoud *et al.*, 2010). Ces paramètres sont : le nombre de particules P_e = 20, l'inertie w = 0.8, C_{pbest} = 0.75 et C_{gbest} = 0.6.

4.3 Paramètres de GA

Les valeurs des paramètres de l'algorithme génétique sont fixées en réalisant plusieurs tests.

- la probabilité de croisement P_c =90%.
- probabilité de mutation P_m = 10% et *Ng=100*.

4.4 Paramètres de GLS

Le critère d'arrêt pour la recherche locale guidée adaptée à notre problème est le temps de CPU égale à 0.5 (s).

5 TESTS NUMERIQUES

Dans cette section, nous présentons les tests numériques des algorithmes développés à savoir ACO-GLS, PSO-GLS et GA-GLS pour résoudre le problème d'équilibrage des lignes d'assemblage robotisées. Afin d'évaluer l'efficacité des algorithmes, nous comparons nos résultats avec ceux obtenus en appliquant deux méthodes exactes. La première est l'énumération complète FEM et l'autre déduite par le solveur Cplex. Cette comparaison concerne la qualité des solutions obtenues et les temps d'exécution des algorithmes. Tous les algorithmes sont testés sur des lignes d'assemblages robotisées qui diffèrent par le nombre de robots *I*, la position dans le produit final *K* et le nombre de couches dans le produit final *n*. La première structure (S1) représente un système avec 2 robots et un produit final composé de 4 ou 8 positions et 2 ou 4 couches de composants. En outre, (S2), (S3) et (S4) présentent respectivement des systèmes avec 4, 6 et 8 robots et les mêmes configurations pour le produit final. Quatre configurations différentes de lignes d'assemblages robotisées sont utilisées pour nos applications numériques. La structure de ces configurations diffère du nombre de positions dans le produit final *K* et par le nombre de couches de composants *n*. Les relations de précédence sont présentées dans la figure 2. Les durées opératoires des tâches d'assemblage sont générées par une loi uniforme. Cette génération de données dépend de la couche du produit final. Dans cet article, la première couche suit la loi uniforme de [1, 1.2] et pour le reste suit aussi la loi uniforme [0.8, 1]. Afin de répondre à un besoin industriel pour optimiser la configuration d'une ligne d'assemblage robotisée, on a adopté les paramètres suivants :

- La proportion de la charge moyenne des différents robots δ= 20%
- Le temps nécessaire pour assembler 104 produits : C_{max} = 43.2 (s) pour les configurations de 2 et 4 robots et C_{max} = 20 (s) pour les configurations 6 et 8 robots.

Ces paramètres correspond à une phase de démarrage d'une ligne d'assemblage robotisée dont nous sommes censé de définir en temps réel les stratégies de prise et dépose des robots. Le tableau 1 présente la comparaison des résultats trouvés par les méthodes exactes et les méthodes approchées hybridées alors que le tableau 2 présente une comparaison entre les temps d'exécutions. Tous les méthodes développées pour le problème d'équilibrage de lignes d'assemblage robotisées sont testés pour quatre structures qui différent par le nombre de robots ainsi le nombre de positions et couches pour le produit final. Le critère d'arrêt pour la recherche guidée adopté à notre problème pour les méthodes développées est le temps CPU égale à 0.5(s). Le critère d'arrêt pour les algorithmes est le temps CPU déduit par le besoin industriel. Ce temps reflète le temps nécessaire pour avoir la solution pour les automates programmable afin de mieux gérer un fonctionnement réel des lignes d'assemblage robotisées. Prenons l'exemple de la structure du (S1) du tableau 1. Cette structure correspond à une configuration avec 2 robots et un produit final composé de 4 ou 8 positions et 2 ou 4 couches. Pour un exemple de 2 robots, un produit de 4 positions et 2 couches et un temps nécessaire pour assembler 104 produits, le nombre de produits saisis par les robots avec les méthodes exactes et hybridées égales à 72. Le temps d'exécution pour trouver cette solution optimale trouvé par Cplex égale à 1.63 (s), par FEM égale à 0.103(s) et par les méthodes approchées hybridées égale au temps fixé par l'industriel **1 seconde**.

Pour les 4 structures testées et pour toutes les instances, les résultats montrent que ACO-GLS, PSO-GLS et GA-GLS convergent respectivement pour 13, 11 et 11 instances testées sur 16 vers la solution optimale trouvée par Cplex et FEM. A noter qu'en prenant en considération les temps d'exécution, les méthodes approchées hybridées présentent un avantage intéressant par rapport aux méthodes exactes avec une réduction moyenne du temps de 90 %. Ceci l'avantage de l'ACO-GLS, PSO-GLS et GA-GLS pour répondre à un besoin industriel qui demande un temps de réponse rapide puisque la solution est installée sur des automates programmables (PLC). D'où l'avantage de nos méthodes approchées pour manager le fonctionnement réel d'une ligne d'assemblage robotisée. On remarque pour cette étude que le ACO-GLS est le meilleur en tenant compte des critères de temps d'exécution et la convergence vers la solution optimale.

6 CONCLUSION

Le problème d'équilibrage de lignes d'assemblage robotisées de type E a été étudié dans cet article. Pour répondre à un problème industriel, un couplage avec des automates programmables (PLC) est considéré pour gérer en temps réel le fonctionnement d'une ligne d'assemblage. De ce fait, nous avons développé des méthodes approchées améliorées avec une recherche locale guidée. Deux méthodes exactes ont été developé afin d'évaluer la performances des solutions déduit par les méthodes developées. Les résultats prouvent l'efficacité des méthodes développées en terme de temps d'exécution et de solutions trouvées. Plusieurs pistes peuvent être considérées dans nos futurs travaux, un contrôleur de logique flou peut être intéressant pour mieux régler les paramètres des algorithmes développés. D'autres travaux peuvent être menés en développant d'autres méthodes de recherche locale adaptées à notre problème.

REMERCIEMENTS

Cette recherche a été soutenue par l'entreprise Aries Packaging (France) et l'association nationale de recherche technique (ANRT).

Structures	Robots	Nombre de Positions dans le produit final	Nombre de couches	Nombre de produits saisis (Cplex)	Nombre de produits saisis (FEM)	Nombre de produits saisis (ACO-GLS)	Nombre de produits saisis (PSO-GLS)	Nombre de produits saisis (GA-GLS)
(S1)	2	4	2	**72**	**72**	**72**	**72**	**72**
			4	**72**	**72**	**72**	**72**	**72**
		8	2	**72**	**72**	**72**	**72**	**72**
			4	**72**	**72**	**72**	**72**	**72**
(S2)	4	4	2	**104**	**104**	**104**	**104**	**104**
			4	**104**	**104**	**104**	**104**	**104**
		8	2	**104**	**104**	103	**104**	100
			4	**104**	**104**	**104**	103	**104**
(S3)	6	4	2	**102**	**102**	**102**	**102**	**102**
			4	**102**	**102**	**102**	**102**	**102**
		8	2	**104**	**104**	**104**	103	102
			4	**102**	**102**	**102**	100	101
(S4)	8	4	2	**104**	**104**	**104**	**104**	**104**
			4	**104**	**104**	**104**	**104**	**104**
		8	2	**104**	**104**	102	100	101
			4	**104**	**104**	102	102	102

Tableau 1 : Résultats numériques pour le nombre de produits saisis avec les différentes instances testées.

Structures	Robots	Position dans produit final	Nombre de couches	Temps d'exécution (Cplex)	Temps d'exécution (FEM)
(S1)	2	4	2	1.63	0.103
			4	0.92	0.112
		8	2	1.53	0 .124
			4	3.88	0.138
(S2)	4	4	2	5.63	2.56
			4	5.88	2.67
		8	2	2.94	2.78
			4	3.78	2.89
(S3)	6	4	2	12.45	50.90
			4	5.86	52.3
		8	2	5.58	52.68
			4	13.7	53.12
(S4)	8	4	2	22.9	1053.57
			4	19.28	1054.6
		8	2	22.9	1055.78
			4	23.8	1056.8
moyenne				9,54	296,07

Tableau 2 : Temps d'exécution du Cplex et FEM

REFERENCES

Baybars. I, 1986. A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, vol. 32, p.909-932.

Boysen. N, M. Fliedner, A. Scholl, 2008 Assembly line balancing: Which model to use when?, *International Journal Production Economics*, 111(2), p.509–528.

Chehade. H, Yalaoui. F, Amodeo.L, De Guglielmo. P., 2008. Ant colony optimization for assembly lines design problem. *Proceedings of the 8th international FLINS'08 conference on computational intelligence in decision and control*, Madrid, Espagne, p.1135-1140.

Chehade. H, Yalaoui. F, Amodeo. L, De Guglielmo. P., 2009 Optimisation multiobjectif par colonies de fourmis pour le problème de dimensionnement de buffers, *Journal of Decision Systems*, 18(2),p.257-287.

Daoud. S., Yalaoui F., Amodeo L., Chehade H., Girard T.,2010. A particle swarm optimization algorithm for a pick and place robotic system. *In proceedings of 24th European Conference on Operational Research*, Portugal.

Daoud. S., Yalaoui F., Amodeo L., Chehade H., Duperray P.,2012. New mathematical model to solve robotic assembly lines balancing. *In proceedings of 14th IFAC Symposium on Information control problems in manufacturing*, Romania.

Dolgui. A and Ihnatsenka. I, 2009. Balancing modular transfer lines with serial–parallel activation of spindle heads at stations, *Discrete Applied Mathematics*, vol.157, p.68-89.

Dongyun. W, Zeng Ping, Li Luowei, Wang Kai, 2010. A Novel Particle Swarm Optimization Algorithm, *Software Engineering and Service Sciences (ICSESS)*, p.408-411.

Dorigo. M, 1992. "*Optimization Learning and Natural Algorithms*", PHD thesis, Politecnice Di Milano.

Gao. J, Sun. Linyan, Wang. Lihua, Gen. Mitsuo, 2009 An efficient approach for type II robotic assembly line balancing problems, *Computers & Industrial Engineering*, vol.56, p.1065-1080.

Guschinskaya. O and Dolgui, 2009. A Comparison of exact and heuristic methods for a transfer line balancing problem, *International Journal of Production Economics,* vol,120, p.276-286.

Hani.Y, Amodeo. L, Yalaoui. F, Chen. H, (2007), Ant colony optimization for solving an industrial layout problem, *European Journal of Operational Research*, vol, 183, p, 633-642.

Holland, J. H.,1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Kennedy. J and Eberhart R.C., 1995. "Particle swarm optimization", *in Proc. IEEE Int . Conf. Neural Netw*., Perth, Australia, vol,4, p.1942-1948.

Levitin. G, Jacob. Rubinovitz, and Boris. Schnits, 2006. A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research*, vol. 168, p.811-825.

Mills. P, E. Tsang, and J. Ford, 2003. Applying an extended guided local search to the quadratic assignment problem. *Annals of Operations Research*, vol. 118, p.121-135.

Nai-Chieh. W and Ming Chao. I, 2011. A solution procedure for type E simple assembly line balancing problem, *Computers&IndustrialEngineering*,vol. 61, p.824-830.

Rubinovitz. J and J. Bukchin, 1991. Design and balancing of robotic assembly lines. *In Proceedings of the fourth world conference on robotics research*, Pittsburgh, PA.

Voudouris. V and E. Tsang, 1996. *Partial constraint satisfaction problems and guided local search*. Technical report, p.337-356.

Ze-qiang. Z, Cheng.W, Lian-sheng. Zhong Bin. T , 2007. Ant Algorithm with Summation Rules for Assembly Line Balancing Problem, *Management Science and Engineering.ICMSE*,p.369-37