International Journal of Business & Economic Strategy (IJBES)
Vol.5, Issue 2, pp. 145-151


# Méthode de Recuit Simulé pour l'optimisation de l'affectation d'opérateurs sur une ligne de production

Sana Bouajaja, Najoua Dridi

*OASIS Laboratory, National Engineering School of Tunis (ENIT)*
*University of Tunis Elmanar, BP. 37 le Belvédère, 1002, Tunis, Tunisia*
Sana.bouajaja@enit.utm.tn, najoua.dridi@enit.rnu.tn

***Abstract*— Ce travail traite le problème d'équilibrage des lignes d'assemblage, l'objectif étant la minimisation du temps de cycle. Nous traitons le cas où le facteur humain est impliqué, c'est à dire la durée d'une opération dépend de l'opérateur qui la réalise. Nous considérons des opérations reliées par des contraintes de précédence et nous cherchons à affecter les opérations et les opérateurs aux différentes stations. Notre but consiste à optimiser l'affectation de ces deux éléments aux stations de travail de la ligne. Ce problème est NP-difficile. Un algorithme de recuit simulé a été proposé et adapté à ce problème. Une étude expérimentale est réalisée pour évaluer les résultats de cette méthode, auparavant, des simulations permettant d'optimiser les paramètres de l'algorithme proposé ont été effectuées. Les résultats obtenus sont plutôt satisfaisants.**

***Keywords*— Ressource humaine; Affectation; Recuit Simulé; Temps de cycle; Ligne d'assemblage**

## I. INTRODUCTION

Gérer ses ressources humaines est aujourd'hui une tâche nécessaire qui préoccupe toute entreprise de services ou de production, afin de réaliser ses objectifs stratégiques. Particulièrement, pour les systèmes industriels où les lignes de production sont manuelles, la ressource humaine est un facteur à ne pas négliger puisqu'elle présente la clé de succès de la production. Pour ces raisons, les industriels doivent prendre en compte le facteur humain dans leur processus de décision. En fait, bien optimiser l'affectation des ressources humaines tenant compte de leurs compétences, pour des lignes de production en grande masse, permet de réduire le temps et le coût de la production. Etant donné l'effet d'échelle (un nombre important de pièces fabriquées pendant toute la durée de vie de la ligne ou pendant la période entre deux rééquilibrages, pour une ligne flexible ou reconfigurable), le gain peut être très important.

C'est dans ce contexte que s'inscrit notre travail, ayant pour objectif d'optimiser l'affectation des opérations d'assemblage aux opérateurs et aux stations de travail d'une ligne d'assemblage manuelle, afin d'améliorer sa productivité, en réduisant son temps de cycle. Ce problème peut être vu comme une combinaison de deux problèmes, connus dans la littérature par leur caractère NP-difficile. Le premier est le problème d'équilibrage des lignes d'assemblage de type 2, Simple Assembly Balancing Problem ou SALBP-2 [1],[2], dont l'objectif est de minimiser le temps de cycle de la ligne pour un nombre de stations donné. Le deuxième est le problème d'allocation des ressources humaines tenant compte de leurs compétences. Un problème composite en découle, nous pouvons lui attribuer l'appellation Assembly Line Worker Assignment and Balancing Problem-2 (ALWABP-2), qui a été initialement proposé par [3], dans un cadre particulier où les opérateurs sont des personnes handicapés, travaillants dans des centres de travail protégé qui leur sont réservés, avec des lignes d'assemblage. L'objectif est de les intégrer tous aux lignes d'assemblage en maximisant le taux de production. Un état de l'art détaillé sur les problèmes d'affectation des ressources humaines dans différents domaines est présenté dans [4], et un autre traitant en particulier le cas des handicapés a fait l'objet de [5].

Cette appellation reste donc valable dans notre cas d'étude, où on traite un ensemble d'opérateurs hétérogènes ayant différentes compétences, et un ensemble d'opérations avec des contraintes de précédence, à assigner à une ligne d'assemblage manuelle constituée par des stations de travail disposées en série. L'objectif à optimiser dans ce travail est la minimisation du temps de cycle, en respectant les différentes contraintes techniques et économiques.

Ce problème d'optimisation combinatoire est NP-difficile. Par conséquent, les méthodes de résolution exactes restent limitées aux instances de petite taille, ce qui justifie notre recours à une méthode approchée qui est le recuit simulé.

Le reste du papier est organisé comme suit : Dans la deuxième section, présentons les contraintes et les hypothèses de notre problème et nous décrivons dans la section 3 le principe de l'algorithme de recuit simulé et ses différentes étapes. La section 4 présente les résultats d'une étude expérimentale permettant d'évaluer la méthode de résolution proposée. Nous terminons ce travail par la conclusion et les perspectives envisageables.

## II. PRESENTATION DU PROBLEME

Nous avons comme données : $N$ opérations à affecter à $H$ opérateurs et à $M$ stations, un graphe de précédence définissant les relations de précédence entre les opérations,

modélisé par une matrice d'adjacence *Prec*, et une matrice de compétence présentant le temps opératoire $T_{ih}$ de chaque opération $i$ ($i$=1…$N$), si elle est exécutée par l'opérateur $h$ ($h$=1…$H$). La résolution de ce problème doit minimiser le temps de cycle $T_c$ de la ligne, tenant compte des hypothèses et contraintes suivantes:

1. la ligne est conçue pour un seul produit,
2. les stations de la ligne sont en série et sans stock tampon,
3. les temps opératoires sont déterministes,
4. les temps opératoires dépendent de la compétence de l'opérateur qui va les exécuter,
5. les relations de précédence doivent être respectées,
6. toutes les opérations doivent être exécutées,
7. une opération est exécutée sur une seule station
8. un opérateur est assigné à une seule station,
9. une station ne comporte qu'un seul opérateur.

Etant donné qu'on a un seul opérateur par station c'est-à-dire $H = M$, nous assimilons chaque opérateur à une station et nous passons à une effectuer une simple affectation des opérations aux opérateurs tout en établissant des relations de précédence entre les opérateurs après chaque affectation, en se basant sur les contraintes de précédence entre les opérations.

## III. Recuit Simule pour le Probleme ALWABP-2

Le recuit simulé est une métaheuristique qui est généralement connue par deux caractéristiques : la facilité de son implémentation, et sa rapidité d'exécution comparée à d'autres métaheuristiques. En outre, elle fournit de bons résultats pour de nombreux problèmes d'optimisation combinatoire.

Par analogie avec la thermodynamique, le recuit simulé repose sur le principe d'accepter une transformation locale qui engendre une dégradation de la solution courante, dans l'attente d'avoir une amélioration ultérieurement, ce qui permet d'éviter d'être piégé dans des minima locaux. En métallurgie, des transformations élémentaires sont appliquées sur le système thermodynamique en réduisant lentement la température le long du processus de recuit, afin d'atteindre un état d'équilibre qui correspond à une énergie minimale. Dans l'algorithme de recuit simulé, cette énergie correspond à la fonction objectif à optimiser et l'état d'équilibre à la solution du problème. A chaque itération de l'algorithme, une modification locale faisant varier l'énergie du système est directement acceptée si elle permet d'améliorer la fonction objectif ($\Delta E \leq 0$). Autrement, si ($\Delta E > 0$) c'est-à-dire la nouvelle solution est moins bonne que la solution courante, alors elle est acceptée avec une probabilité $P$, calculée en fonction de la température $T$ et de l'énergie $E$.

$$P = e^{\frac{-\Delta E}{T}}$$

Cette règle d'acceptation est appelée critère de Metropolis. Un nombre $r$ est généré aléatoirement $r \in [0,1]$. Si $P \leq r$ alors la nouvelle solution est acceptée, sinon elle sera refusée.

Nous remarquons que la température est un paramètre contrôlant ce critère d'acceptation. En effet, lorsque $T$ est élevée alors $P \cong 1$, ainsi toute solution voisine sera forcément acceptée, mais quand $T$ est petite, on aura $P \cong 0$, alors les transformations dégradant la fonction objectif auront moins de chance d'être acceptées. Le long du processus, la température décroit lentement, en suivant une loi de décroissance jusqu'à ce qu'elle tend vers zéro. Soit qu'on garde la même valeur de température et on la diminue après un certain nombre d'itérations, on définit ainsi des paliers de température, ou bien elle est réduite à chaque itération. Généralement, la fonction de réduction utilisée est une suite géométrique tel que: $T_{i+1} = \alpha\, T_i$. Avec $\alpha$ est le coefficient de refroidissement, avec $\alpha \in [0,1]$.

### *A. Principe de fonctionnement de l'algorithme proposé*

Nous appliquons la méthode de recuit simulé à notre problème en partant d'une solution initiale $S_0$ ($S = S_0$), obtenue par une méthode heuristique simple. A cette solution correspond un temps de cycle $T_c$, qui est assimilé à l'Energie, dans le problème de métallurgie. Une température initiale $T_0$ ($T = T_0$) est également définie au départ.

La valeur de cette température décroît par paliers, selon un schéma de décroissance géométrique. Cette règle de refroidissement, définissant l'évolution de la température dépend de trois paramètres :

- La température initiale $T_0$,
- Le coefficient de refroidissement $\alpha$,
- La longueur du palier, qui est le nombre d'itérations $N_{iter}$ pour lequel la température est constante.

L'ajustement de ces paramètres sera effectué via une étude expérimentale permettant de déterminer leurs valeurs de façon empirique.

Enfin, l'algorithme s'arrête lorsque la température atteint une valeur quasiment nulle. Ces différentes étapes sont illustrées par l'organigramme de Fig. 1.

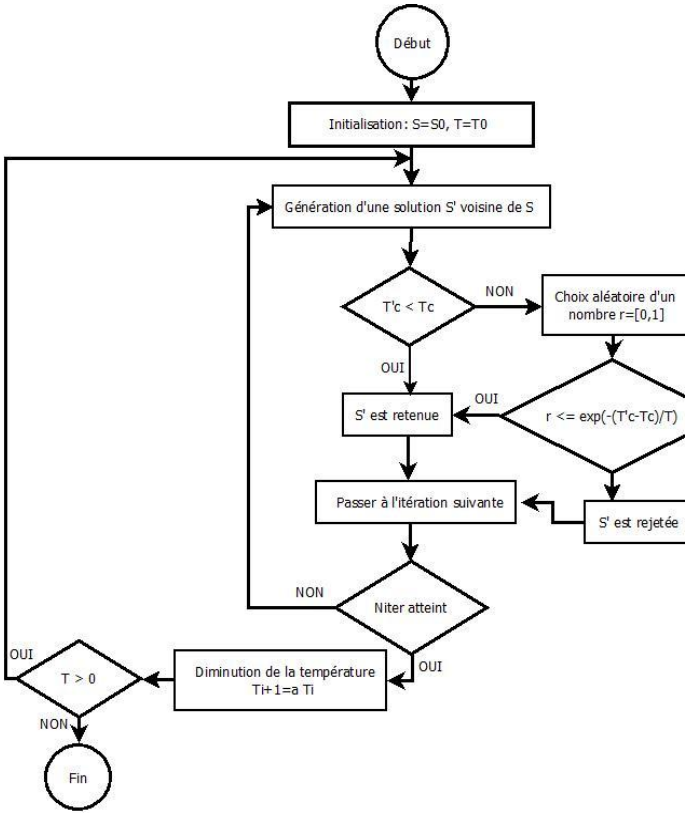


Fig. 1. Organigramme de l'algorithme du Recuit simulé

### *B. Construction du voisinage*

A partir d'une solution *S* donnée, nous effectuons des modifications élémentaires permettant d'avoir une nouvelle solution « voisine » $S'$ qui peut être de meilleure qualité ($T_c' \leq T_c$) ou de moins bonne qualité ($T_c' > T_c$).

Soit $\Delta T_c = (T_c' - T_c)$, si on souligne une amélioration de la solution, c'est-à-dire $\Delta T_c \leq 0$, la solution voisine trouvée est systématiquement acceptée. En cas de dégradation ($\Delta T_c < 0$), la nouvelle solution est acceptée avec la probabilité *P*, qui sera plus grande pour une dégradation faible que pour une forte dégradation de la qualité de la solution.

Pour définir le voisinage d'une solution *S*, nous procédons tout d'abord à définir pour cette solution la station goulot, occupé par l'opérateur le plus chargé, noté $h_g$. Nous trions les opérations affectées à cet opérateur dans l'ordre croissant de leurs temps opératoires. Si le déplacement de la première opération vers l'un des opérateurs adjacents à $h_g$ satisfait les différentes contraintes du problème, ce déplacement est effectué et la nouvelle solution est évaluée. Dans le cas contraire, on passe à l'opération suivante de la liste triée et on reprend la même procédure jusqu'à trouver l'opération à déplacer.

Notons qu'on a opté, dans le choix de cette opération, pour l'ordre croissant des temps opératoires, afin de réduire le risque de créer par ce déplacement un nouveau poste goulot avec une charge plus importante que celle du précédent, et la solution sera ainsi dégradée. Ce cas de figure se pose notamment lorsque la variabilité des temps opératoires entre les différents opérateurs pour une même opération est faible, et l'écart entre les charges des opérateurs n'est pas important.

Nous justifions également ce déplacement de l'opération choisie vers les opérateurs adjacents de $h_g$ par le fait de ne pas nuire au respect des relations de précédence entre les opérations.

Ce mouvement élémentaire, impliquant une modification du temps de cycle, est effectué sur la solution courante à chaque itération de l'algorithme, tout en gardant la température constante. La nouvelle solution obtenue permet soit d'améliorer notre critère à optimiser ($T_c$), soit de le dégrader. Nous présentons cette étape de résolution dans l'Algorithme 1.

**Algorithme 1. Construction du voisinage**

**Données :** *N*, *H*, *Prec*, *T*;
**Entrée :** S
**Sortie :** *S' //Solution voisine de S*

1. Déterminer l'opérateur goulot $h_g$ de la solution *S ;*
2. Trier les opérations de $h_g$ dans l'ordre croissant de leurs temps opératoires
3. Déplacer de $h_g$ vers l'un de ses voisins adjacents, l'opération $i_g$ qui respecte la précédence et la compétence et parmi celles ayant le plus petit temps opératoire sur $h_g$
4. Calculer la nouvelle solution *S'*
5. **Retourner** *S'*

L'Algorithme 2 permet de regrouper toutes les étapes précédemment exposées toutes en détails.

**Algorithme 2. Algorithme de Recuit Simulé proposé**

1. **Données :** *N*, *H*, *Prec*, *T*;
2. **Initialisation :** $T = T_0$ ; // *Température initiale*
3. $S = S_0$ ; // *Solution initiale obtenue par une heuristique*
4. **Tant Que** la température $T > 0$ **Faire**
5. **Pour** chaque itération *i* de 1 à $N_{iter}$ **Faire**
   *// Chercher une solution voisine S'de S*
6. Appeler l'algorithme « Calcul d'une solution voisine »;
7. Calculer $\Delta T_c = T_c' - T_c$ ;
   *// Appliquer la règle de Metropolis*
8. **Si** $\Delta < 0$ **Alors**
9. Accepter S' ; $S = S'$ ;
10. **Sinon Si** $P = e^{(\frac{-\Delta}{T})} > random[0,1]$ **Alors**
11. Accepter S' ; $S = S'$;
12. **Sinon** Rejeter *S'* ;
13. **Fin Si**
14. **Fin Si**
15. **FinPour**
16. Diminuer la température $T_{i+1} = \alpha\ T_i$
17. **Fin Tant Que**
18.

19. **Retourner** (la meilleure solution);

Afin d'optimiser et d'ajuster les différents paramètres de notre méthode, d'étudier sa performance et d'évaluer la qualité des solutions obtenues, nous menons une étude expérimentale.

## IV. Etude Experimentale et Resultats

Notre algorithme a été programmé et testé sur Windows avec l'environnement de développement Dev C++, en utilisant le langage C sur un ordinateur portable équipé d'un processeur Intel(R) Core(TM) i3 CPU M370 @2.40GHz 2.40 GHz.

Dans cette étude, un ensemble d'expérimentations numériques a été réalisé en recourant à un jeu de données de la littérature présenté dans [6]. Ces instances sont formées par 4 grandes familles (Roszieg, Heskia, Tonge et Wee-mag) ayant chacune son propre graphe de précédence qui se distingue par le nombre d'opérations *N* et la densité du graphe *D*. Chaque famille comporte 8 groupes de 10 instances (80 instances par famille), qui sont formées en se basant sur 3 facteurs à savoir: le nombre d'opérateurs *H*, la variabilité des temps opératoires *Var* et le pourcentage d'infaisabilité *Inf*. Notons que les instances d'un même groupe diffèrent par la matrice de compétence. Pour chaque facteur (*N*, *H*, *D*, *Var*, *Inf*), on considère deux valeurs: faible ou grande, d'où on a 320 instances à tester.

### *A. Réglage des paramètres de l'algorithme de recuit simulé*

L'algorithme de recuit simulé (RS) est contrôlé par différents paramètres ($T_0$, $\alpha$, $N_{iter}$). Avant d'évaluer cet algorithme, et afin de bien choisir les valeurs des paramètres, nous avons effectué des tests numériques permettant d'étudier l'évolution du système en fonction de chaque paramètre.

L'algorithme RS étant randomisé, nous résolvons chaque instance 10 fois. Nous retenons pour chacune des instances la meilleure solution trouvée, la valeur moyenne des 10 solutions obtenues (*Moy* $T_c$), et la moyenne des temps de calcul (*Moy t(s)*).

Nous procédons dans notre étude numérique à considérer tous les paramètres successivement. Chaque fois, on varie la valeur du paramètre visé et on fixe les autres. Les résultats obtenus sont illustrés sous forme de graphiques.

Dans Fig. 2, nous étudions l'impact de $N_{iter}$ sur l'évolution de la qualité de l'algorithme, en considérant la moyenne des solutions pour la totalité des 320 instances. Nous varions $N_{iter}$ sur l'intervalle [5,100]. Cette figure révèle l'existence de deux pics sur cet intervalle. La première correspond à la valeur minimale 271.3 pour $N_{iter}$ = 10 et la deuxième correspond à la valeur maximale 279.36 pour $N_{iter}$ = 50. Nous fixons alors la valeur de $N_{iter}$ ($N_{iter}$ = 10). Notons également qu'après un certain seuil, l'augmentation de $N_{iter}$ ne permet pas d'améliorer les solutions cherchées, bien au contraire.

Il est évident que lorsque $N_{iter}$ augmente, la somme des temps d'exécution croît également. Toutefois, ce temps reste de l'ordre de quelques millisecondes pour chaque instance même celles de grandes tailles, comme le montre Table 1.

Nous fixons les valeurs des paramètres $T_0$ et $N_{iter}$ et nous varions celui du coefficient de refroidissement, afin d'analyser son effet sur les résultats. Fig. 3 et Fig. 4 présentent respectivement l'impact de $\alpha$ sur la moyenne des solutions et sur la somme des temps de calcul. Il est clair que les meilleurs résultats sont obtenus pour $\alpha$ = 0.8, en termes de solutions et de temps de résolution. De même, en testant $T_0$ nous trouvons que $T_0$ = 0.4 est la valeur optimale à garder pour ce paramètre. Par conséquence, la combinaison suivante des paramètres ($N_{iter}$ = 10, $\alpha$ = 0.8, $T_0$ = 0.4) forme le choix qu'on a fait, étant donné qu'il fournit les résultats les plus satisfaisants, avec une résolution très vite du problème.

Ce choix a permis d'améliorer les résultats obtenus au début par un choix quelconque. Ces résultats finaux, donnés par les valeurs retenues des trois paramètres peuvent être utilisés ultérieurement pour évaluer la qualité de notre métaheuristique en la comparant à d'autres méthodes de résolution.

Notons que Table 1 permet de visualiser les résultats numériques obtenus, pour chaque groupe relatif à une famille particulière, regroupant 10 instances qui diffèrent uniquement par la matrice de compétence. Nous présentons: Moy Tc qui est la moyenne des temps de cycle de chaque groupe. Moy t(s) qui est la moyenne des temps de calcul du programme (en secondes), pour les 10 instances de ce groupe. Ces valeurs sont en fonction de la taille de l'instance du problème (*N*, *H*), de la densité du graphe, de la variabilité des temps opératoires et de l'infaisabilité. Il est clair que le recuit simulé permet d'obtenir l'ensemble des solutions rapidement mais afin d'évaluer leur qualité il s'avère nécessaire de les comparer soit à une borne inférieure ou à une solution optimale. Ces résultats serviront par la suite à effectuer une étude comparative permettant de mieux juger la performance de notre algorithme par rapport à ce qui existe dans la littérature.

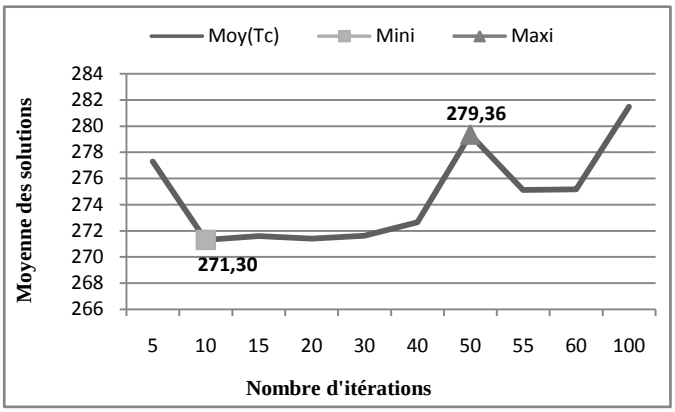


Fig. 2. Evolution de la valeur moyenne des solutions en fonction de Niter

| ille | Groupe | H | Var | Inf | *Moy Tc* | *Moy t( s)* |
|---|---|---|---|---|---|---|
| Roszieg | 1 | 4 | [1, *Ti*] | 10% | 26,4 | 0,00245 |
| | 2 | | [1,3*Ti*] | 20% | 26,4 | 0,00259 |
| | 3 | | [1, *Ti*] | 10% | 43,8 | 0,00254 |
| | 4 | | [1,3*Ti*] | 20% | 39,7 | 0,00287 |
| | 5 | 6 | [1, *Ti*] | 10% | 20,8 | 0,00233 |
| | 6 | | [1,3*Ti*] | 20% | 23,6 | 0,00218 |
| | 7 | | [1, *Ti*] | 10% | 36,3 | 0,00275 |
| | 8 | | [1,3*Ti*] | 20% | 37 | 0,00255 |
| | | | | | **31,75** | **0.00253** |
| Heskia | 1 | 4 | [1, *Ti*] | 10% | 346,9 | 0,00264 |
| | 2 | | [1,3*Ti*] | 20% | 360,9 | 0,00261 |
| | 3 | | [1, *Ti*] | 10% | 352 | 0,00257 |
| | 4 | | [1,3*Ti*] | 20% | 380,3 | 0,003 |
| | 5 | 7 | [1, *Ti*] | 10% | 369,2 | 0,00401 |
| | 6 | | [1,3*Ti*] | 20% | 369,8 | 0,00339 |
| | 7 | | [1, *Ti*] | 10% | 404,4 | 0,00381 |
| | 8 | | [1,3*Ti*] | 20% | 360,5 | 0,00364 |
| | | | | | **368** | **0.00321** |
| Tonge | 1 | 10 | [1, *Ti*] | 10% | 463,7 | 0,00861 |
| | 2 | | [1,3*Ti*] | 20% | 483,2 | 0,00852 |
| | 3 | | [1, *Ti*] | 10% | 592,6 | 0,01149 |
| | 4 | | [1,3*Ti*] | 20% | 595,4 | 0,01044 |
| | 5 | 17 | [1, *Ti*] | 10% | 463,8 | 0,01074 |
| | 6 | | [1,3*Ti*] | 20% | 478,5 | 0,01012 |
| | 7 | | [1, *Ti*] | 10% | 502,3 | 0,01103 |
| | 8 | | [1,3*Ti*] | 20% | 493,7 | 0,01081 |
| | | | | | **509,15** | **0.01022** |
| Wee-Mag | 1 | 11 | [1, *Ti*] | 10% | 129,6 | 0,01265 |
| | 2 | | [1,3*Ti*] | 20% | 149,4 | 0,00962 |
| | 3 | | [1, *Ti*] | 10% | 223,7 | 0,01161 |
| | 4 | | [1,3*Ti*] | 20% | 229,2 | 0,01099 |
| | 5 | 19 | [1, *Ti*] | 10% | 113,5 | 0,01328 |
| | 6 | | [1,3*Ti*] | 20% | 108,5 | 0,01188 |
| | 7 | | [1, *Ti*] | 10% | 213 | 0,01328 |
| | 8 | | [1,3*Ti*] | 20% | 212,6 | 0,01168 |
| | | | | | **172,43** | **0.01187** |
| | | | | | **270,33** | **2,23** |

## V. CONCLUSIONS

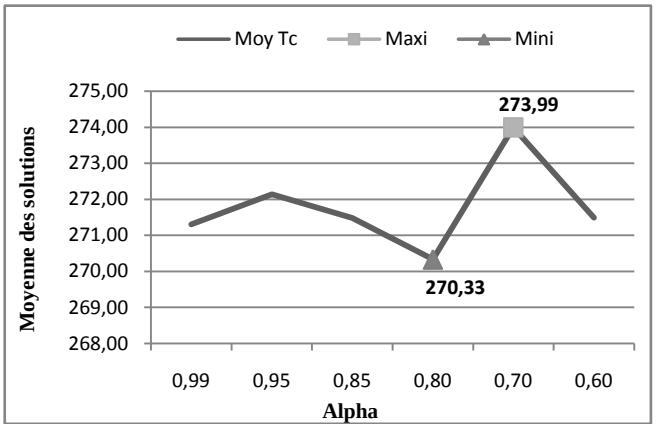


Fig. 3. Evolution de la valeur moyenne des solutions en fonction de α

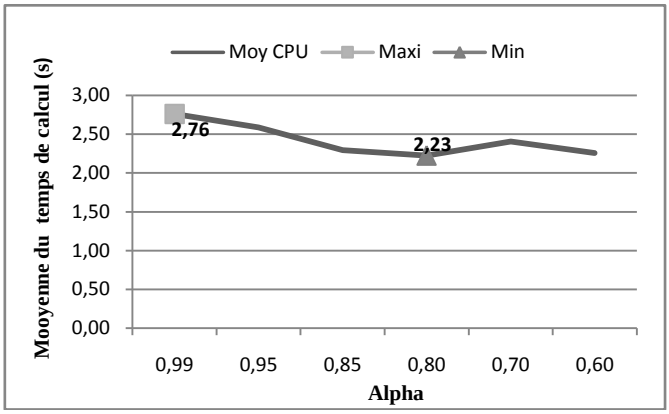


Fig. 4. Evolution du temps de calcul en fonction de α

TABLE I RESULTATS NUMERIQUES DU RECUIT SIMULE

Nous avons présenté dans ce papier un problème d'affectation des ressources humaines aux lignes d'assemblage, dans l'objectif d'améliorer la productivité, en minimisant son temps de cycle.

Le recours à une métaheuristique dans sa résolution est justifié par le caractère NP difficile de ce problème. L'utilisation de la méthode recuit simulé pour ce fin nous a permis de tirer profit de ses avantages telle que la facilité d'implémentation, la résolution du problème en un temps de calcul réduit, donner des résultats satisfaisants même pour les instances de grandes tailles. Néanmoins, un réajustement délicat de ses paramètres était nécessaire à faire, en réalisant de nombreux tests de calcul. Ces essais expérimentaux nous ont permis de déterminer les valeurs optimaux des différents paramètres et de les utiliser pour déterminer les solutions des instances testées de la littérature.

Les résultats obtenus peuvent être améliorés en hybridant cette méthode avec une autre méthode exacte, une heuristique ou une métaheuristique ce qui permet d'exploiter les avantages cumulés des ces méthodes hybridées.

## REFERENCES


[1] B. Rekiek, A. Dolgui, A. Delchambre, and A. Bratcu, "State of art of optimization methods for assembly line design". *Annual Reviews in control*,26(2), p.163-174, 2002.

[2] C. Becker and A. Scholl, "A survey on problems and methods in generalized assembly line balancing". *European journal of operational research*,168(3), 694-715, 2006.

[3] C. Miralles, J. P. Garcia-Sabater, C. Andres, and M. Cardos, "Advantages of assembly lines in sheltered work centres for disabled. A case study". *International Journal of Production Economics*, 110(1), 187-197, 2007.

[4] S. Bouajaja S. et N. Dridi , "Survey on Human Resource Allocation Problem and its Applications", *Operational Research: International Journal*, 1-31, DOI: 10.1007/s12351-016-0247-8.

[5] S. Bouajaja S. et N. Dridi , "Intégration des Handicapés dans le Milieu Industriel et Optimisation de Leur Affectation aux Lignes d'Assemblage: Etude de la Littérature", in *Proc. 25th IBIMA*, 2015, p.2379.

[6] A. A. Chaves, C. Miralles, and L. A. N. Lorena, "Clustering search approach for the assembly line worker assignment and balancing problem". In *Proc. 37th International conference on computers and industrial engineering*, p.1469-1478, 2007.