CIGI 2011
Heuristiques efficaces pour la conception de réseaux d'assemblage/désassemblage

NABIL NAHAS[1,2 3], MUSTAPHA NOURELFATH[1,3], MICHEL GENDREAU[2,3]

[1] UNIVERSITE LAVAL
Faculté des sciences et de génie, Département de génie mécanique, Québec (QC), Canada

[2] ÉCOLE POLYTECHNIQUE DE MONTREAL
Département de mathématiques et génie industriel, Québec (QC), Canada

[3] CENTRE INTERUNIVERSITAIRE DE RECHERCHE SUR LES RESEAUX D'ENTREPRISE, LA LOGISTIQUE ET LE TRANSPORT (CIRRELT)
Québec (QC), Canada

Nabil.Nahas.1@ulaval.ca; Mustapha.Nourelfath@cirrelt.ca; Michel.Gendreau@cirrelt.ca

Résumé **- Cet article formule un problème de conception optimale des systèmes en structures de réseaux d'assemblage/désassemblage, avec des machines non fiables et des stocks intermédiaires à capacités finis. L'objectif est de maximiser l'efficacité (taux de production) du réseau sous une contrainte de budget. L'approche proposée se distingue des approches existantes par son pouvoir de sélection à la fois des technologies des machines et des capacités des stocks intermédiaires, et ce, dans le contexte d'une structure de réseau. Les machines sont caractérisées par leurs coûts, leurs taux de panne, leurs taux de réparation et leurs taux de traitement. Chaque stock intermédiaire est caractérisé par une capacité maximale et un coût unitaire associé au stockage. Une méthode de décomposition est utilisée pour évaluer le taux de production. Deux metaheuristiques basées sur l'algorithme de recherche d'harmonie et l'algorithme génétique sont développées pour la résolution du problème d'optimisation combinatoire résultant.**

Abstract **- This article formulates a new optimal design problem of an assembly/disassembly manufacturing network. The objective is to maximize production rate subject to a total cost constraint. Machines are chosen from a list of products available in the market and buffers' sizes are chosen within a range. The buffers are characterized by their cost coefficients associated with the buffer size. The machines are characterized by their cost, failure rate, repair rate and processing time. To estimate assembly/disassembly network throughput, a decomposition-type approximation is used. The optimal design problem is formulated as a combinatorial optimization one where the decision variables are buffers and types of machines. To solve this problem, Harmony search algorithm and Genetic algorithm are proposed.**

Mots clés **– Réseaux d'assemblage/désassemblage, méthode de décomposition, algorithme de recherche d'harmonie, algorithme génétique.**

Keywords **- Assembly/disassembly network, decomposition, Harmony search algorithm, Genetic algorithm.**

1 INTRODUCTION

1.1. Contexte et motivation

Le secteur manufacturier joue un rôle primordial dans l'économie du Canada, en raison de son apport important au produit intérieur brut, à l'emploi, à la formation brute de capital fixe et aux exportations de marchandises, et en raison de son utilisation des technologies de pointe. Comme l'industrie manufacturière vit actuellement de profonds bouleversements et rencontre de nombreuses difficultés, il est devenu plus important que jamais d'y diminuer les coûts et augmenter la performance des équipements (disponibilité, taux de production, fiabilité, etc.). Les entreprises manufacturières doivent ainsi développer des moyens de production fiables, maintenir un niveau élevé de la qualité des produits et optimiser l'utilisation des équipements. Parmi les techniques généralement utilisées pour augmenter la performance dans les systèmes manufacturiers, on retrouve : l'augmentation de la fiabilité des machines, l'optimisation de la planification et de l'ordonnancement des activités, l'augmentation du niveau de redondance, l'utilisation des stocks tampons, et la mise en place des plans de maintenance. Les systèmes manufacturiers étudiés dans ce projet utilisent la redondance et/ou des stocks tampons, et opèrent selon une structure réseau d'assemblage/désassemblage. Ce type de systèmes est souvent rencontré dans des secteurs manufacturiers clés de l'économie, tels que l'aérospatiale et l'automobile. Pour ce type de systèmes manufacturiers, l'un des problèmes importants est le développement de méthodes d'analyse et d'optimisation de performance rapides et précises à utiliser pendant les phases de conception et d'exploitation.

Plusieurs chercheurs ont examiné le cas des lignes manufacturières séries. Il existe aussi plusieurs travaux traitant de l'analyse et de l'optimisation des réseaux d'assemblage/désassemblage. Ces derniers sont généralement plus difficiles à étudier.

1.2. Description du problème

Un réseau d'assemblage et de désassemblage (AD) est un système manufacturier dans lequel les machines peuvent effectuer des opérations d'assemblage et de désassemblage. A chaque machine d'assemblage, deux pièces ou plus, provenant de ses stocks intermédiaires (SI) amont, sont assemblées et la pièce résultante est ensuite stockée dans son SI aval. Une machine de désassemblage effectue l'opération de désassemblage d'une pièce provenant du SI amont et stocke chacune des pièces résultantes dans les SI aval.

En supposant que les machines ont le même temps de traitement, nous considérons le problème pratique où :

- Pour chaque machine, plusieurs technologies sont disponibles dans le marché ; chacune de ces technologies étant caractérisée par un taux de production, un taux de panne, un taux de réparation et un coût.
- Les stocks intermédiaires peuvent être dimensionnés en tenant compte de leur capacité maximale et du cout unitaire associé au stockage.

Dans notre approche, la conception optimale du réseau AD consiste à faire une sélection des technologies des machines *e*t à dimensionner les stocks intermédiaires.

1.3. Travaux existants et approche utilisée

Il existe de nombreuses publications consacrées à la revue des méthodes analytiques de modélisation et d'analyse des lignes de production. Une revue complète et classique est celle de Dallery et Gershwin [Dallery et Gershwin, 1992]. Il y a également une littérature assez importante sur l'optimisation des stocks intermédiaires. Une revue de littérature sur l'optimisation des stocks peut être trouvée par exemple dans [Gershwin et Schor, 2000]. Le but des travaux existants est de dimensionner les tailles des stocks intermédiaires, en supposant que les machines sont déjà sélectionnées. Ainsi, les seuls paramètres à trouver sont les capacités des stocks intermédiaires.

Les travaux relatifs aux réseaux AD ont été aussi traités dans la littérature. Bulgak et al. [Bulgak et al., 2000] ont étudié les systèmes d'assemblage asynchrones. Ils ont proposé une approche analytique pour la conception des systèmes d'assemblage. Paik et al. [Paik et al., 2002] ont proposé des méthodes approximatives d'évaluation du taux de production pour les systèmes de production à boucle fermée. [Tan, 2001] a développé un modèle d'évaluation pour un système à 3 stations et un stock intermédiaire. Le même modèle a été étudié par [Helber et Mehrtens, 2003] où une approche analytique utilisant les équations différentielles pour déterminer les probabilités de transition a été proposée. [Bulgak, 2006] a présenté un modèle de simulation basé sur des réseaux de neurones couplé avec l'algorithme génétique pour le dimensionnement des stocks intermédiaires.

Le but des travaux existants est de dimensionner les tailles des stocks intermédiaires, en supposant que les machines sont déjà sélectionnées. Ainsi, les seuls paramètres à trouver sont les capacités des stocks intermédiaires. L'approche proposée se distingue des approches existantes par son pouvoir de *sélection à la fois des technologies des machines et des capacités des stocks*. Cette sélection conjointe des machines et des stocks conduit à un problème d'optimisation combinatoire encore plus difficile que celui d'allocation de stocks. Afin de résoudre ce problème, nous développons deux algorithmes à base de metaheuristiques recherche d'harmonie [Geem et al., 2001] et l'algorithme génétique [Holland, 1975]. L'efficacité du réseau AD sera déterminée en utilisant la méthode de décomposition développée par [Di Mascolo et al., 1991].

Le reste de l'article est organisé comme suit. Dans la section 2, le problème de conception optimale est formulé. La section 3 résume la méthode de décomposition utilisée pour évaluer l'efficacité du réseau AD. Les deux algorithmes proposés sont détaillés dans la section 4. Dans la section 5, des résultats numériques sont présentés. Des conclusions et des perspectives sont formulées dans la section 6.

2 PROBLÈME DE CONCEPTION OPTIMALE

Le réseau AD est représenté comme un graphe connecté dans lequel les nœuds représentent les machines et les arcs représentent les stocks intermédiaires. Chaque SI est connecté à exactement deux machines. Pour chaque machine M_i, l'ensemble des SI amont est noté *U(i)* et l'ensemble des SI aval est noté *D(i)* (graphe 1).

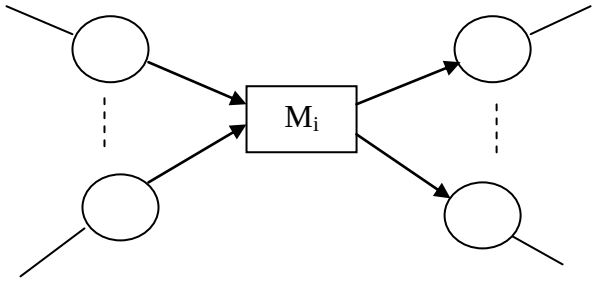

Figure 1. Réseau d'assemblage/désassemblage [Gerswin, 1994]

Notation

M	: nombre de machines
NV_i	: nombre de technologies disponibles pour la machine M_i
$x(i,j)$	: variable binaire. $x(i,j)$=1, si la techonologie j de la machine M_i est choisie
$N(i)$	: espace de stockage du SI i
N_i^{max}	: espace de stockage maximale autorisé pour le SI i
N	: $\{N(i), i=1,..,m-1\}$, $N(i) \in \{1,.., N_i^{max}\}$
x	: $\{x(i,j), i=1,..,m; j=1,..,M_j\}$, $x(i,j) \in \{0,1\}$
$C_m(i,j)$	: cout de la technologie j de la machine M_i
$C_b(i)$	: cout associé à une unité de stockage du SI i
B	: Budget disponible

Le problème de conception optimal peut être formulé comme suit: trouver la configuration du système N et x qui maximise le taux de production total P tel que le cout total associé est inferieur ou égal au budget B. En tenant compte de ces notations, notre problème de conception optimale est exprimé comme suit :

$$Maximiser \quad P[N(i), x(j,k)] \qquad i = 1, \dots, m-1; j = 1, \dots, m; k = 1, \dots, NV_j$$

Sujet à

$$N(i) \leq N_i^{max} \qquad \forall\, i = 1,..,m-1 \qquad (1)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{NV_i} x(i,j) \times C_m(i,j) + \sum_{i=1}^{m-1} C_b(i) \times N(i) \leq B \qquad (2)$$

$$\sum_{j=1}^{NV_i} x(i,j) = 1 \qquad \forall\, i = 1,\dots,m \qquad (3)$$

$$x(i,j) \in \{0,1\} \qquad \forall\, i = 1,\dots,m \quad et\ \forall\, j = 1,\dots,NV_i \qquad (4)$$

$$N(i) \in \{1,\dots,N_i^{max}\} \qquad \forall\, i = 1,\dots,m-1 \qquad (5)$$

La contrainte (1) représente la contrainte de la taille maximale du SI. La contrainte (2) représente la contrainte de budget; sans perte de généralité, nous considérons que B est un entier. La relation (3) traduit la contrainte de choix multiples. (4) et (5) définissent les variables de décision. Une solution qui satisfait toutes les contraintes, est dite faisable.

3 ÉVALUATION DE L'EFFICACITÉ

Chaque machine M_i est décrite par son taux de panne λ_i, son taux de réparation μ_i et son temps de traitement T_i. Nous supposons que le réseau est homogène (temps de traitement des machines égaux). L'efficacité du réseau est calculée en utilisant la méthode proposée par [Di Mascolo et al., 1991]. Cette méthode est basée sur une décomposition du système original en un ensemble de sous-systèmes qui sont plus faciles à étudier. Plus précisément, il s'agit de diviser le réseau en un ensemble de lignes virtuelles dont chacune est composée de deux machines virtuelles séparées par un stock intermédiaire (voir figure 2).

Les solutions exactes de lignes de production constituées de deux machines séparées par un stock intermédiaire existent dans la littérature [Dubois et Forestier, 1981]. Les deux machines virtuelles $M_u(i,j)$ et $M_d(i,j)$ modélisent les portions de la ligne se trouvant respectivement en amont et en aval du stock intermediaire $B_{i,j}$.

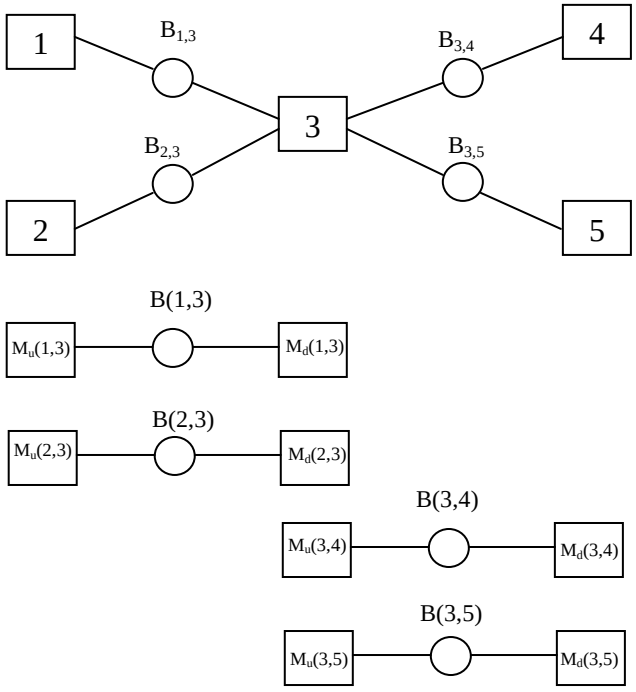

Figure 2. Méthode de décomposition [Di Mascolo et al., 1991]

Comme toutes les techniques de type décomposition, la méthode proposée par les auteurs [Di Mascolo et al., 1991] passe ainsi par les trois étapes suivantes :

- La caractérisation des sous-systèmes.
- La définition des équations reliant les paramètres inconnus des sous-systèmes décomposés.
- La mise au point d'un algorithme de résolution des équations.

Pour plus de détails sur la méthode de décomposition, le lecteur est référé à [Di Mascolo et al., 1991].

4 ALGORITHMES DE RÉSOLUTION

4.1 Algorithme de recherche d'harmonie

L'algorithme de recherche d'harmonie (RH) est une metaheuristique développée par [Geem et al., 2001]. Elle est basée sur le processus de performance musical qui consiste à trouver l'harmonie parfaite dans un orchestre musical où chaque musicien joue une note pour trouver une meilleure harmonie. D'une manière analogue, chaque variable de décision dans le processus d'optimisation a une valeur pour trouver la meilleure solution. L'algorithme RH a été appliqué avec succès sur plusieurs problèmes comme le problème de voyageur de commerce [Geem et al. 2001], le problème de tournée de véhicule [Geem et al. 2005a] et le problème de conception des structures [Geem et al. 2005b]

Les principales étapes de l'algorithme RH sont présentées comme suit :

- Étape 1. Initialisation des paramètres

  Dans cette étape, les paramètres de l'algorithme sont initialisés : nombre HMS de solutions générées, taux de sélection HMCR, taux d'ajustement PAR et le critère d'arrêt.

- Étape 2. Génération des solutions initiales (appelées mémoire de l'harmonie HM)

  Dans cette étape, un ensemble de HMS solutions sont aléatoirement générées et pour chaque solution i (i=1,..,HMS), la fonction objectif f_i est calculée. La figure 3 présente la structure générale de HM. Cette mémoire peut être considérée comme une matrice contenant un ensemble d'harmonies ou solutions.

- Etape 3. 'Improvisation' d'une nouvelle harmonie (solution) à partir de la matrice HM.

  Une nouvelle solution x=(x_1,..,x_n) est générée à partir de la matrice HM avec une probabilité HMCR. En utilisant le paramètre HMCR, chaque variable x_i (i=1,..,n) est choisie aléatoirement du vecteur (y_{1i},..,y_{HMSi}) de la matrice HM avec un taux HMCR.

- Étape 4. Si la nouvelle solution est faisable et meilleure que la plus mauvaise solution dans la matrice HM, inclure la nouvelle solution dans HM et exclure la plus mauvaise.

- Étape 5. Si le critère d'arrêt n'est pas satisfait, aller à l'étape 2.

  Dans leurs expérimentations, les auteurs [Geem et al., 2001] ont introduit un autre paramètre appelé taux d'ajustement (PAR), utilisé essentiellement pour échapper aux locaux minimaux et améliorer les solutions obtenus. Le PAR est utilisé après qu'une nouvelle solution est construite et peut être appliqué à chaque variable de la solution. Dans ce cas, la valeur de chaque variable est modifié avec une probabilité PAR.

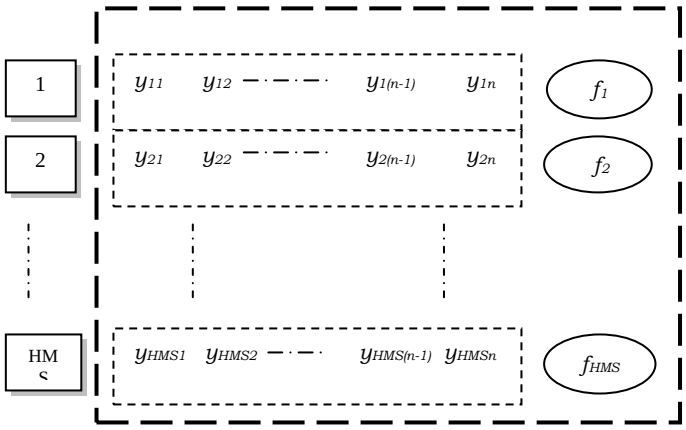

Figure 3. Structure générale de la matrice HM

4.1.1 L'algorithme RH pour la conception optimale des réseaux AD

Matrice HM

Pour appliquer l'algorithme RH à notre problème, il est nécessaire de représenter adéquatement la matrice HM. Dans l'algorithme proposé, chaque rangée i de la matrice HM est composée de (2.m-1) variables. Les m premières variables représentent les technologies affectées aux m machines. Les (m-1) variables restantes représentent la taille des (m-1) stocks intermédiaires. La figure 3 montre la structure de la matrice HM adoptée où les variables sont représentées par y_{ij} (i = 1,.. , HMS, j = 1,.., n = 2.m-1).

Construction de la nouvelle solution

Une nouvelle solution $\mathbf{x} = (x_1, \ldots, x_{(2xm-1)})$ est générée à partir de la matrice HM en utilisant les paramètres HMCR et PAR. Ces paramètres aident l'algorithme à obtenir des solutions localement ou globalement améliorées. La solution est construite de la façon suivante :

Pour chaque variable x_k (k = 1, . . . , 2.m-1):

- Si $k \leq m$ (i.e. x_k est associé à la machine k)

$$x_k \in \begin{cases} \{y_{1k}, y_{2k}, \ldots, y_{HMS,k}\} & \text{avec la probabilité HMCR} \\ \{1, \ldots, NV_i\} & \text{avec la probabilité (1-HMCR)} \end{cases}$$

- si $k > m$ (i.e. x_k est associé au SI k-m)

$$x_k \in \begin{cases} \{y_{1k}, y_{2k}, \ldots, y_{HMS,k}\} & \text{avec la probabilité HMCR} \\ \{1, \ldots, N_i^{\max}\} & \text{avec la probabilité (1-HMCR)} \end{cases}$$

Avec une probabilité PAR, certaines variables de la nouvelle solution peuvent être modifiées de la façon suivante :

- Etape 1. choisir aléatoirement une variable x_i de la solution construite.
- Etape 2. Si x_i représente un stock intermédiaire :
  - Choisir une autre variable x_j représentant un autre stock intermédiaire.
  - Echanger les contenus des deux variables x_i et x_j.
- Etape 3. Si xi représente la technologie d'une machine :
  - Incrémenter ou décrémenter le niveau de technologie d'une façon aléatoire.

4.2 Algorithme Génétique

L'algorithme génétique (AG) est un algorithme d'optimisation stochastique basé sur les principes de la sélection naturelle et de la génétique. Introduit par [Holland, 1975], l'AG est une approche robuste et puissante de développement d'heuristiques pour les problèmes d'optimisation combinatoire de grande taille. L'AG a prouvé son efficacité dans la résolution des problèmes NP-difficile comme par exemple le problème d'affectation quadratique [Wu et Ji, 2007] ou le problème d'ordonnancement des job-shop [Pezzella et al, 2008]. Pour une brève introduction aux algorithmes génétiques, le lecteur est référé à [Austin, 1990].
Dans cet article, l'algorithme génétique proposé est basé sur la version de [Whitley, Kauth, 1988]. Les principales étapes de l'algorithme génétique adopté sont les suivantes :

- Étape 1. Générer aléatoirement une population initiale de N_s solutions.
- Étape 2. Sélectionner aléatoirement deux solutions et produire une nouvelle solution en utilisant une procédure de croisement.
- Étape 3. Permettre à la nouvelle solution de muter avec une certaine probabilité p_m. La mutation permet un léger changement de la solution et maintient la diversité des solutions. Cette procédure diminue la possibilité de convergence prématurée à un optimum local.
- Étape 4. Décoder la nouvelle solution pour obtenir les valeurs de la fonction objectif. Ces valeurs sont une mesure de qualité utilisées pour comparer les différentes solutions.
- Étape 5. Appliquer une procédure de sélection qui compare la nouvelle solution avec la plus mauvaise solution dans la population. La meilleure solution des deux joindra la population et la plus mauvaise est écartée. Si la population contient des solutions identiques suivant cette procédure, éliminer les redondances. La taille de la population diminuera alors.
- Étape 6. Générer de nouvelles solutions aléatoires à la place des solutions manquantes pour renouveler la population après avoir exécuté les étapes 2-5 P_{rep} fois ou si la taille de la population est réduite à 1.
- Étape 7. Exécuter un nouveau cycle génétique (aller à l'étape 2).
- Étape 8. Terminer l'AG après N cycles génétiques.

4.2.1 L'algorithme génétique pour la conception optimale des réseaux AD

Représentation de la solution

Pour appliquer l'algorithme génétique à notre problème, il est nécessaire de représenter adéquatement la solution. Dans l'algorithme proposé, comme dans le cas de l'algorithme RH, la solution est composée de (2.m-1) variables. Les m premières variables représentent les technologies affectées aux m machines. Les (m-1) variables restantes représentent la taille des (m-1) stocks intermédiaires.

Procédure de croisement

Trois procédures de croisement ont été testées sur ce problème:

- Croisement à un point : tous les éléments se trouvant avant une position choisie aléatoirement sont copiés du premier parent et le reste est copié du second parent.
- Croisement à deux points : deux positions sont sélectionnées aléatoirement. Les éléments se trouvant à l'intérieur de ses positions sont copiés du premier parent et les autres éléments sont copiés du second parent.
- Croisement uniforme : chaque élément est copié d'une façon aléatoire soit du premier parent, soit du second parent.

Les résultants obtenus des tests effectués sur ces trois procédures ont montré que le croisement à un point donne les meilleurs résultats.

5 EXEMPLE ET RÉSULTATS NUMÉRIQUES

La majorité des benchmarks qui existent dans la littérature concernent le dimensionnement des tailles de stocks intermédiaires, en supposant que les machines sont déjà sélectionnées. Rappelons que l'approche proposée dans ce travail est différente puisqu'elle vise la sélection à la fois des machines et des tailles des stocks. Comme il n'y a pas de benchmarks correspondant exactement à notre problème, nous allons proposer un exemple de grande taille pour comparer les deux algorithmes.

Dans cet exemple, nous considérons un réseau d'assemblage/désassemblage à 15 machines. La figure 4 montre la structure de ce système. La table 1 contient les données de ce réseau. Pour cet exemple, le budget disponible est varié de 350 à 450$. Tous les stocks intermédiaires ont une capacité maximale de 20 unités et le cout associé à une unité de stockage est de 1$.

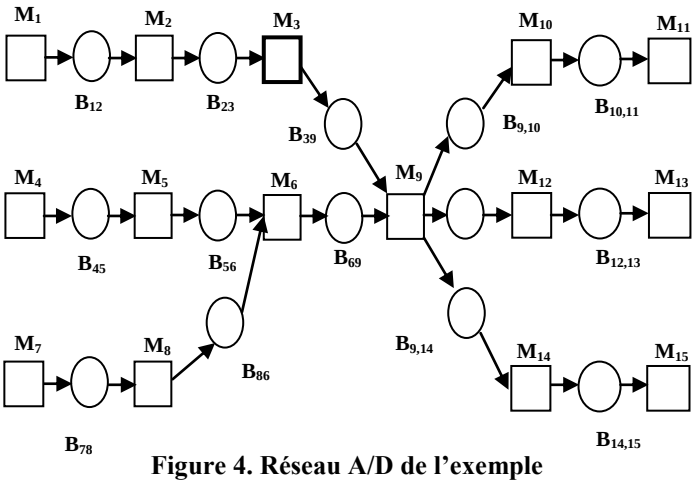

Figure 4. Réseau A/D de l'exemple

Une analyse du comportement de l'algorithme génétique montre que la taille de la population est très variable. La figure 5 montre un exemple de cette variation. La figure 6 montre que toutes les solutions de la matrice HM de l'algorithme RH convergent vers une même solution (optimum local) plusieurs fois durant l'exécution du programme.

Nous avons implémenté les deux algorithmes et exécuté plusieurs essais pour régler les différents paramètres (10 essais pour chaque valeur de paramètre). Les paramètres considérés ici sont ceux qui ont un effet direct ou indirect sur la performance des algorithmes (i.e. les paramètres *HMCR,PAR,HMS* pour l'algorithme RH et P_{rep}, procédures de sélection, de croisement, et de mutation pour l'algorithme génétique). Les valeurs des différents paramètres sont les suivants :

- Algorithme RH : HMCR=0.915, PAR=0.1, HMS=50.
- Algorithme génétique : p_m=0.9, P_{rep}=1000.

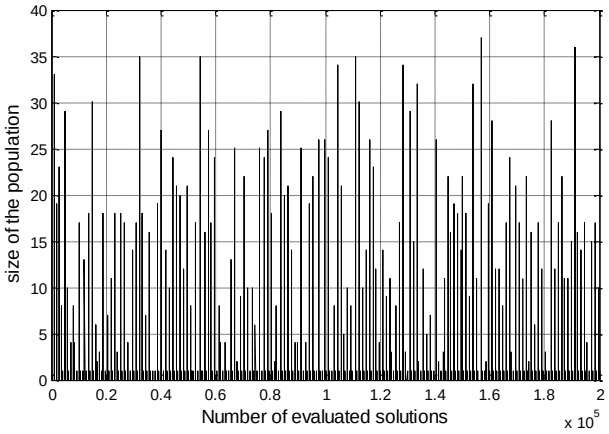

Figure 5. Changement de la taille de la population (AG)

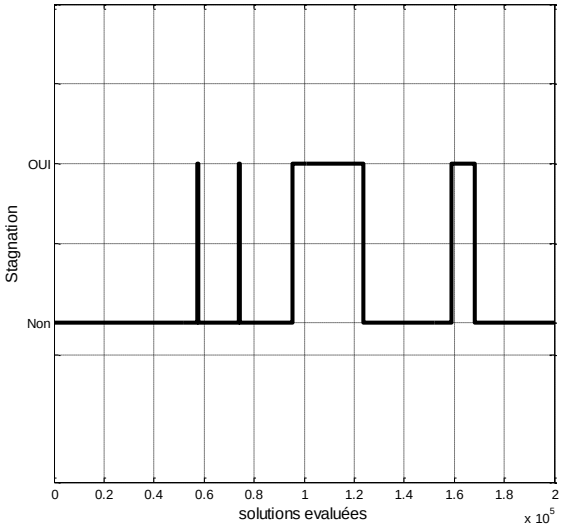

Figure 6. Stagnation des solutions (RH)

Pour comparer les deux algorithmes, le nombre maximal de solutions évaluées a été fixé à 2.10^5 pour toutes les exécutions. Les résultats obtenus après 10 essais sont présentés dans la table 2. La table 2 montre que l'algorithme génétique présente les meilleurs résultats dans les trois cas du problème. Par contre, en se qui concerne les valeurs moyennes et l'écart-type, c'est l'algorithme RH qui a les meilleurs résultats. Il est à noter aussi que l'algorithme génétique converge relativement plus rapidement que l'algorithme RH (figure 7).

Machine/ Techn.		V(1)	V(2)	V(3)	V(4)	V(5)	V(6)	V(7)	V(8)	V(9)	V(10)
M_1	λ_1	0.1000	0.0842	0.0699	0.0616	0.0539	0.0523	0.0426	0.0407	0.0274	0.0151
	μ_1	0.3870	0.3810	0.3540	0.3840	0.3915	0.4005	0.4110	0.4050	0.3810	0.3810
	c_1	4	7	9	13	17	23	26	27	32	33
M_2	λ_2	0.0878	0.0634	0.0536	0.0464	0.0398	0.0305	0.0286	0.0262	0.0154	0.0130
	μ_2	0.3848	0.3735	0.3465	0.3525	0.3405	0.3525	0.3832	0.3518	0.3510	0.3840
	c_2	6	9	10	14	18	20	22	24	25	29
M_3	λ_3	0.1210	0.1000	0.0860	0.0544	0.0506	0.0435	0.0309	0.0203	0.0166	0.0110
	μ_3	0.4028	0.3705	0.3435	0.3525	0.3525	0.3742	0.3772	0.3742	0.3480	0.3855
	c_3	8	12	18	19	22	24	26	29	32	36
M_4	λ_4	0.1460	0.1000	0.0543	0.0443	0.0336	0.0248	0.0200	0.0170	0.0107	0.0070
	μ_4	0.3668	0.4013	0.3398	0.3413	0.3420	0.4073	0.3915	0.3600	0.3840	0.3758
	c_4	6	8	12	15	21	24	26	32	36	38
M_5	λ_5	0.0939	0.1040	0.0920	0.0548	0.0516	0.0496	0.0505	0.0355	0.0284	0.0152
	μ_5	0.3622	0.4118	0.3780	0.3458	0.3503	0.3622	0.4005	0.3705	0.4013	0.3855
	c_5	5	7	9	13	16	19	25	31	36	41
M_6	λ_6	0.1500	0.1000	0.0524	0.0441	0.0367	0.0296	0.0190	0.0131	0.0110	0.0100
	μ_6	0.3638	0.4050	0.3450	0.3720	0.3450	0.3682	0.3503	0.3945	0.3765	0.3922
	c_6	7	12	16	18	20	24	27	32	35	39
M_7	λ_7	0.0700	0.0539	0.0439	0.0400	0.0354	0.0300	0.0274	0.0195	0.0130	0.0100
	μ_7	0.3592	0.3383	0.3532	0.3510	0.3413	0.3450	0.3428	0.3600	0.3622	0.3518
	c_7	6	8	11	16	18	23	24	28	33	38
M_8	λ_8	0.1000	0.1100	0.0883	0.0700	0.0713	0.0400	0.0324	0.0279	0.0150	0.0120
	μ_8	0.3585	0.4320	0.3990	0.3922	0.4095	0.3705	0.3892	0.3720	0.4402	0.3832
	c_8	7	9	10	15	20	25	28	33	36	41
M_9	λ_9	0.1200	0.1000	0.0628	0.0528	0.0504	0.0419	0.0339	0.0210	0.0122	0.0100
	μ_9	0.3930	0.3788	0.3900	0.3420	0.3465	0.3585	0.3735	0.3622	0.3765	0.3968
	c_9	6	11	17	22	23	25	26	32	34	40
M_{10}	λ_{10}	0.1000	0.0810	0.0600	0.0529	0.0422	0.0323	0.0224	0.0124	0.0110	0.0100
	μ_{10}	0.4103	0.4080	0.3855	0.3435	0.3473	0.3548	0.3810	0.3862	0.3458	0.3413
	c_{10}	7	10	14	19	21	22	24	28	34	38
M_{11}	λ_{11}	0.1010	0.0972	0.0800	0.0758	0.0628	0.0527	0.0428	0.0318	0.0286	0.0179
	μ_{11}	0.3855	0.3788	0.3645	0.3525	0.3818	0.3435	0.3488	0.3645	0.3435	0.3450
	c_{11}	5	6	10	11	15	17	19	23	28	30
M_{12}	λ_{12}	0.1200	0.0829	0.0690	0.0746	0.0503	0.0419	0.0361	0.0293	0.0179	0.0100
	μ_{12}	0.4073	0.3570	0.3443	0.3825	0.3555	0.3698	0.3450	0.3938	0.3960	0.3862
	c_{12}	8	13	15	19	21	26	28	31	32	35
M_{13}	λ_{13}	0.0992	0.0855	0.0795	0.0677	0.0617	0.0527	0.0426	0.0322	0.0259	0.0120
	μ_{13}	0.3375	0.3825	0.3690	0.3668	0.3398	0.3608	0.3735	0.3862	0.3473	0.3892
	c_{13}	4	8	13	15	19	21	25	31	33	36
M_{14}	λ_{14}	0.1040	0.0862	0.0600	0.0581	0.0441	0.0339	0.0271	0.0200	0.0130	0.0105
	μ_{14}	0.3772	0.3848	0.3473	0.3518	0.3428	0.3562	0.3510	0.3480	0.3608	0.3900
	c_{14}	7	12	16	18	20	25	28	31	33	36
M_{15}	λ_{15}	0.1500	0.1050	0.0890	0.0765	0.0548	0.0537	0.0375	0.0205	0.0166	0.0105
	μ_{15}	0.4110	0.3668	0.3885	0.3840	0.3840	0.3938	0.3503	0.3900	0.3608	0.3660
	c_{15}	5	8	9	15	18	20	23	26	32	35

Table 1. Données de l'exemple

Probleme	HS				AG			
	Min	Moy.	Max	Std	Min	Moy.	Max	Std
B=450	0.82906	**0.83030**	0.83118	**0.00065**	0.82687	0.83010	**0.83174**	0.00127
B=400	0.79690	**0.79928**	0.80028	**0.00118**	0.79562	0.79869	**0.80028**	0.00167
B=350	0.76615	**0.76693**	0.76750	**0.00048**	0.76121	0.76649	**0.76754**	0.00195

Table 2. Résultats obtenus (10 essais)

B ($)	*P*	C_T ($)	V	N
450	0.83174	450	{4,9,8,4,6,7,3,7,9,7,5,9,6,6,8}	{6,5,8,9,7,8,8,6,3,10,8,7,10,11}
400	0.80028	400	{4,3,7,4,4,7,3,4,9,7,5,9,2,5,5}	{8,9,9,7,9,9,5,9,7,10,11,8,7,11}
350	0.76754	350	{2,2,7,3,4,4,2,4,9,3,5,5,2,5,3}	{8,8,9,9,10,9,6,9,9,11,10,7,6,10}

Table 3. Les meilleures structures obtenues du réseau

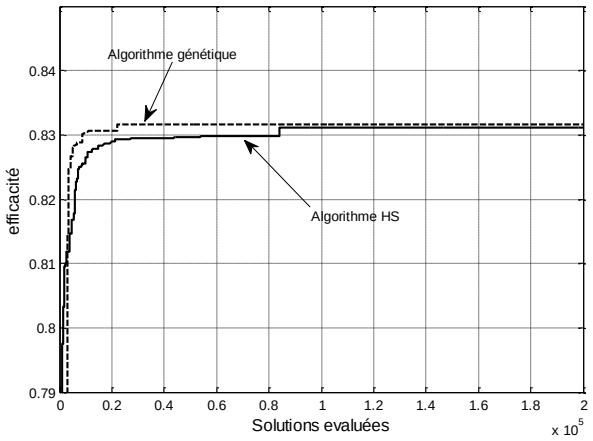

Figure 7. Évolution de la meilleure solution

6 CONCLUSION

Dans cet article, nous avons formulé un nouveau problème de conception optimale des réseaux A/D ayant des machines non fiables et des stocks finis. Dans les approches existantes, les variables de décision concernent seulement les tailles des stocks tampons. L'approche proposée étend les problèmes classiques d'allocation de stocks en formulant un problème plus compliqué où les variables de décision sont les stocks *et* les types de machines, et ce, dans le contexte d'une structure de réseau. Une méthode de décomposition a été utilisée pour estimer l'efficacité de chaque configuration possible. Comme le problème formulé est un problème d'optimisation combinatoire difficile, une énumération exhaustive de toutes les solutions possibles n'est pas réaliste. Nous avons développé deux approches heuristiques basées sur l'algorithme de recherche d'harmonie et l'algorithme génétique. Une comparaison des deux algorithmes a été réalisée sur un exemple numérique de grande taille. Les résultats des tests ont montré que les meilleurs résultats ont été obtenus par l'algorithme génétique. Par contre, en se qui concerne les valeurs moyennes et l'écart-type, c'est l'algorithme RH qui a les meilleurs résultats

7 RÉFÉRENCES

Dallery Y., Gershwin S.B. (1992). Manufacturing Flow Line Systems: A Review of Models and analytical Results. Queueing Systems theory and Applications, Special Issue on Queueing models of Manufacturing Systems, 12(1-2), pp. 3-94.

Gershwin SB., Schor JE. (2000). Efficient Algorithms for Buffer Space Allocation. Annals of Operations Research. 93, pp. 117-144.

Bulgak A.A, Tarakci Y., Verter V. (1999). Robust design of asynchronous flexible assembly systems. International Journal of Production Research. 37 (14), pp. 3169–3184.

Paik, C.H., Kim, H.G., Cho, H.S. (2002). Performance analysis for closed loop production systems with unreliable machines and random processing times. Comput. Ind. Eng., 42, pp. 207–220.

Tan, B. (2001). A three station merge system with unreliable stations and a shared buffer. Math. Comput. Model., 33, pp. 1011–1026.

Helber, S. and Mehrtens, N. (2003). Exact analysis of a continuous material merge system with limited buffer capacity and three stations. In Analysis and Modeling of Manufacuring Systems, hrsg.v., edited by S.B. Gershwin, Y. Dallery, C.T. Papadopoulos and J. MacGregor Smith, pp. 85–121 (Kluwer: Amsterdam).

Bulgak A.A (2006). Analysis and design of split and merge unpaced assembly systems by metamodelling and stochastic search. International Journal of Production Research, 44 (18–19), pp. 4067–4080.

Geem, Z.W., Kim, J.-H. and Loganathan, G.V., (2001). A new heuristic optimization algorithm: harmony search. Simulation, 76(2), pp. 60-68.

Holland JH (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology. *Control and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press.

Di Mascolo, David R. And Dallery Y (1991). Modeling and Analysis of assembly Systems with Unreliable Machines and Finite Buffers. IIE transactions, 23 (4), 315-330.

Gershwin, S.B., (1994). Manufacturing systems engineering (Prentice-Hall).

D. Dubois et J.P. Forestier (1981)**.** Productivité et en-cours moyens d'un ensemble de deux machines séparées par une zone de stockage**.** R.I.AR.O. Autom. Syst. Analysis and Control ; 16, pp.105-132.

Geem, Z.W., Lee, K.S. and Park, Y., (2005a). Application of Harmony Search to Vehicle Routing. American Journal of Applied Sciences, 2(12), pp. 1552-1557.

Geem, Z.W., Lee, K.S. and Tseng, C.-L., (2005b). Harmony Search for Structural Design in Proceedings of 2005 Genetic and Evolutionary Computation Conference (GECCO-2005). Washington, DC, USA, June 25-29 2005, pp. 651-652.

Wu Y and Ji P (2007). Solving the Quadratic Assignment Problems by a Genetic Algorithm with a New Replacement Strategy. International Journal of Humanities and Social Sciences, 1(3), pp. 1307-8046.

Pezzella F, Morganti G, Ciaschetti G (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. Computers and Operations Research, 35(10), pp. 3202-3212.

Whitley D, Kauth J (1988). GENITOR: a different genetic algorithm, Tech. Rep. CS-88-101, Colorado State University, Fort Collins, CO.

Austin S (1990). An introduction to genetic algorithms, AI Expert, Mfr. (1990) 49-53.