Contribution à l'ordonnancement d'ateliers agroalimentaires utilisant des méthodes d'optimisation hybrides

Asma Karray

N° d'ordre : 182

ECOLE CENTRALE DE LILLE
UNIVERSITE DE TUNIS EL MANAR
ECOLE NATIONALE D'INGENIEURS DE TUNIS

THESE

présentée en vue d'obtenir le grade de

DOCTEUR
en

Spécialité : Automatique et Informatique Industrielle

par

Asma KARRAY

DOCTORAT DELIVRE CONJOINTEMENT
par l'Ecole Centrale de Lille et l'Ecole Nationale d'Ingénieurs de Tunis

Titre de la thèse :

CONTRIBUTION A L'ORDONNANCEMENT D'ATELIERS AGROALIMENTAIRES UTILISANT DES METHODES D'OPTIMISATION HYBRIDES

soutenue le 05 Juillet 2011, devant le Jury d'Examen composé de :

MM.	**Pr.**	**Noureddine**	**ELLOUZE**	**Président**
	Pr.	**Naceur**	**BELHADJ BRAIEK**	**Rapporteur**
	Pr.	**Abdallah**	**EL MOUDNI**	**Rapporteur**
	Pr.	**José**	**RAGOT**	**Examinateur**
	Pr.	**Mohamed**	**BENREJEB**	**Examinateur**
	Pr.	**Pierre**	**BORNE**	**Examinateur**

Thèse en cotutelle préparée au Laboratoire d'Automatique, Génie Informatique et Signal
de l'Ecole Centrale de Lille et à l'Unité de Recherche LARA Automatique
de l'Ecole Nationale d'Ingénieurs de Tunis

Ecole Doctorale SPI 072 (EC Lille)
PRES Université Lille Nord-de-France

AVANT PROPOS

Ce présent travail a été effectué au sein de l'Unité de Recherche LARA Automatique de l'Ecole Nationale d'Ingénieurs de Tunis (ENIT) et du Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS) de l'Ecole Centrale de Lille (EC-Lille).

Je tiens à exprimer notre vive gratitude à Monsieur Noureddine ELLOUZE, Professeur et Directeur de l'Unité de Recherche Traitement du Signal, Traitement de l'Image et Reconnaissance des Formes de L'ENIT, pour m'avoir fait le grand honneur d'accepter de présider le Jury d'Examen. Qu'il trouve ici l'expression de ma reconnaissance et de mon profond respect.

C'est un agréable devoir d'exprimer ma très vive reconnaissance à Monsieur le Professeur Mohamed BENREJEB, Directeur de l'Unité de Recherche LA.R.A. Automatique, et à Monsieur Pierre BORNE, Professeur à l'Ecole Centrale de Lille pour m'avoir guidé durant toute l'élaboration de ce mémoire avec le sérieux et la compétence qui les caractérisent. Qu'ils trouvent ici l'expression de ma profonde gratitude pour l'intérêt qu'ils ont porté à mes travaux, les conseils éclairés qu'ils m'ont prodigués et leurs encouragements. Qu'il trouve aussi ici le témoignage de mon profond respect pour leur rigueur scientifique et leur constante disponibilité.

Je tiens à remercier très sincèrement Monsieur Naceur BELHADJ BRAIEK, Professeur et Directeur de l'Ecole Supérieur de Sciences et Technologies de Tunis et Monsieur Abdellah EL MOUDNI, Professeur à l'Université de Technologie Belfort-Monbéliard, d'avoir bien voulus accepté de rapporter sur mes travaux de recherche. Qu'ils trouvent ici, le témoignage de ma profonde reconnaissance.

Mes remerciements vont également à Monsieur José RAGOT, Professeur à l'Institut National Polytechnique de Lorraine, pour avoir accepté de juger ce travail, qu'il soit grandement remercié.

Je remercie enfin tous les chercheurs de l'Unité de Recherche LARA Automatique et du Laboratoire d'Automatique, Génie Informatique et Signal de l'EC-Lille et toute autre personne qui, par leur aide, par leur amicale présence et par leur sympathie, ont rendu ce travail agréable. Je leur exprime ici toute ma gratitude.

TABLE DES MATIERES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

L'environnement actuel des entreprises est caractérisé par des marchés soumis à une grande concurrence et sur lesquels les attentes des clients deviennent de plus en plus exigeantes sur les plans qualité, coût et délais de mise à disposition. Ainsi, la mondialisation et le développement rapide de nouvelles technologies d'information et de communication renforcent cette évolution du marché [Lopez, 01].

Dans un tel contexte, rester toujours performant, passe obligatoirement par une gestion de production plus fiable. En effet, l'amélioration de la gestion de production vise à organiser le fonctionnement du système de production, et à mieux gérer ses différentes opérations. Ainsi, l'amélioration de la production dépend étroitement de l'ordonnancement d'ateliers.

L'ordonnancement d'ateliers consiste à prévoir l'enchaînement de toutes les opérations élémentaires nécessaires à la réalisation des ordres de fabrication sur les ressources de production tout en tenant compte des contraintes internes et externes.

La résolution d'un problème d'ordonnancement passe par une phase d'identification et de modélisation et par une phase de recherche de la méthode de résolution adéquate. Dans la première phase, les contraintes à respecter et les critères à optimiser, lors de la résolution sont identifiés et décrits.

Dans le cadre de notre travail, ils sont spécifiques à l'industrie agroalimentaire, domaine qui présente certaines particularités dues à la nature des produits manipulés et fabriqués, ayant des durées de vie relativement courtes. Chacun des composants primaires, des produits semi-finis et des produits finis est caractérisé par une date limite de validité avant laquelle il doit être consommé. Cette particularité détermine le mode d'organisation dans les ateliers de produits agroalimentaires. Ainsi, les contraintes et les objectifs retenus, dans nos travaux, sont fortement liés à cet aspect. Conjointement aux contraintes cumulatives de ressources et aux contraintes de précédence ordinairement considérées dans les problèmes d'ordonnancement, le respect des dates de validités des composants requis pour la réalisation d'une tâche constitue aussi une contrainte à respecter.

Chaque tâche doit être entamée avant la péremption de ses composants. En effet, la péremption d'un composant génère une perte du prix du revient du produit fabriqué et engendré des retards de livraison. Dans ce contexte, le lancement de production au plus tôt est conseillé.

Le produit fini a aussi une durée de vie limitée. Ainsi, le consommateur souhaite en disposer immédiatement après la fin de fabrication et de le consommer avant sa péremption. Dans cette politique de marché, les grandes surfaces de distribution imposent aux industriels une pénalité, dite de discount de distribution, par jour de stockage de produit fini avant livraison. De ce fait, le prix de vente d'un produit fini diminue proportionnellement à la durée de stockage avant son expédition aux surfaces de ventes. L'un des objectifs à atteindre est de minimiser l'intervalle de temps entre la date de fin de fabrication et la date d'expédition pour minimiser le discount de distribution.

Notre travail s'articule autour de cette problématique dans le but de développer un outil d'ordonnancement qui respecte ces contraintes spécifiques à l'agroalimentaire et vise à maîtriser le discount de distribution et la péremption des composants.

La deuxième phase de résolution d'un problème d'ordonnancement, est la recherche et le choix de la méthode de résolution adéquate. Dans cette optique, nos travaux proposent des solutions pour le problème d'ordonnancement multi-objectifs relatif aux industries agroalimentaires.

L'ordonnancement multi-objectifs est un problème NP-difficile. Sa résolution optimale s'avère, dans la plupart des cas, impossible à obtenir à cause de son caractère fortement combinatoire. Les méthodes exactes requièrent un effort calculatoire qui croît exponentiellement avec la taille du problème. Ainsi, les méthodes approchées ont été proposées pour résoudre ce problème ou obtenir un résultat satisfaisant en temps raisonnable. Parmi ces méthodes, les métaheuristiques ont prouvé leur efficacité pour la résolution de problèmes combinatoires.

Les métaheuristiques, apparues dans les années 80, sont des algorithmes d'optimisation stochastiques et progressent vers un optimum par un échantillonnage d'une fonction objectif dans le but est la résolution de problèmes d'optimisation difficiles. Ces approches sont inspirées par des analogies avec la physique (le Recuit Simulé), avec la biologie (la Recherche Tabou, les Algorithmes Evolutionnaires) ou avec l'éthologie (les Colonies de Fourmis, les Essaims Particulaires).

Les métaheuristiques peuvent être groupées en deux classes : les métaheuristiques à parcours comme la recherche tabou ou le recuit simulé et les métaheuristiques à base de population à l'instar des algorithmes évolutionnaires et les essaims particulaires.

La conception de métaheuristiques pour la résolution de problèmes d'optimisation combinatoires est généralement une question d'équilibre entre intensification et diversification. En règle générale, les métaheuristiques à parcours sont connues pour leur pouvoir d'exploitation de meilleures solutions trouvées, alors que les métaheuristiques à base de population sont plutôt efficaces pour l'exploration de l'espace de recherche. Ainsi, au lieu d'essayer d'améliorer une méthode en termes de diversification ou une autre en termes d'intensification, une nouvelle approche consiste à faire coopérer les deux méthodes et à bénéficier de leurs caractéristiques respectives.

L'idée de faire coopérer différentes métaheuristiques n'est pas neuve, puisqu'il est apparu que toutes les méthodes n'avaient pas les mêmes propriétés. On a donc cherché à profiter des avantages de ces différentes approches afin de rendre les métaheuristiques plus performantes et plus robustes. Une métaheuristique hybride est une approche résultant de la combinaison de plusieurs méthodes d'optimisation (métaheuristiques, heuristiques ou méthodes exactes) dont au moins une métaheuristique.

La coopération entre les méthodes d'optimisation prend généralement l'une des formes suivantes :

- hybride séquentielle relative à deux algorithmes sont impliqués l'un après l'autre ; les résultats fournis par le premier étant, aussi, les solutions initiales du second à la manière d'un pipeline ;
- hybride parallèle synchrone relative à un algorithme de recherche utilisé à la place d'un opérateur ;
- hybride parallèle asynchrone relative à plusieurs algorithmes de recherche travaillant en parallèle et s'échangent des informations.

Dans notre travail, nous nous intéressons, exclusivement, à l'hybridation séquentielle constituée de métaheuristiques.

Le premier chapitre de ce mémoire propose, dans un premier temps, une vue d'ensemble sur les problèmes d'ordonnancement des systèmes de production. Nous rappelons d'abord les différents éléments qui composent un problème d'ordonnancement et nous présentons ensuite une typologie des problèmes d'ordonnancement en distinguant les différents types d'ateliers et proposons différentes méthodes de représentation des problèmes d'ordonnancement. Cette classification nous amène à étudier la complexité des problèmes d'ordonnancement. Nous nous sommes intéressés, par ailleurs, à la présentation des différentes méthodes

d'optimisation développées dans la littérature, classées en méthodes exactes et en méthodes approchées. A la fin de ce chapitre, la problématique relative à l'amélioration des méthodes d'optimisation approchées est posée.

Après l'introduction des spécificités et des différents problèmes rencontrés dans un atelier en industries agroalimentaires, la résolution du problème multi-objectifs est posé, dans le deuxième chapitre, en utilisant trois nouvelles approches basées sur les algorithmes génétiques.

La première approche est basée sur l'idée d'utiliser les algorithmes génétiques de façon séquentielle et la deuxième sur la notion du parallélisme alors que la troisième approche correspond à une coopération des deux approches précédemment citées.

Un codage spécifique est recherché pour permettre la meilleure représentation possible du problème traité.

Deux exemples relatifs à l'industrie agroalimentaire et les résultats obtenus par leur comparaison avec les approches dites classiques, à savoir l'approche Pareto-Optimalité et l'approche d'agrégations, sont, par la suite, traités.

Dans le troisième chapitre, après une vue d'ensemble sur l'hybridation des métaheuristiques et une classification hiérarchique, sont présentées les différentes métaheuristiques utilisées pour l'hybridation. Nous développons, mettons en œuvre et testons, ensuite, deux approches hybrides proposées. La première est basée sur une coopération entre les algorithmes génétiques et la recherche tabou et la deuxième approche sur une coopération entre les algorithmes génétiques et le recuit simulé.

CHAPITRE I

INTRODUCTION A L'ORDONNANCEMENT DE SYSTEMES DE PRODUCTION

I.1. Introduction

L'ordonnancement est une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. Les problèmes d'ordonnancement sont présents dans tous les secteurs

d'activités de l'économie, depuis l'industrie manufacturière [Pinedo, 55] jusqu'à l'informatique [Blazewicz, 96].

Résoudre un problème d'ordonnancement consiste à organiser des tâches, c'est-à-dire à déterminer leurs dates de démarrage et à leurs attribuer des ressources matérielles ou humaines nécessaires, de telle sorte que des contraintes soient respectées, afin d'optimiser un certain objectif préalablement défini [Gotha, 93], [Lopez, 01]. Cette planification des ressources pour réaliser une grande variété de produits finis dans des délais impératifs, est un garant de la compétitivité d'une entreprise et c'est une des raisons pour lesquelles les problèmes d'ordonnancement sont devenus plus complexes et plus importants.

Dans ce chapitre, quelques généralités, sur les problèmes d'ordonnancement dans les ateliers de production, sont d'abord introduites. Une description des principales méthodes d'optimisation utilisées dans la littérature est ensuite réalisée, nous permettant ainsi de préciser celles que nous utiliserons dans la suite de ce mémoire.

I.2. Ordonnancement dans un atelier de production

Les problèmes d'ordonnancement apparaissent dans tous les domaines de l'informatique, de l'industrie, de la construction et de l'administration, etc. [Carlier, 88].

I.2.1. Ordonnancement - Définitions

Ordonnancer le fonctionnement d'un système industriel de production consiste à gérer l'allocation des ressources au cours du temps, tout en optimisant au mieux un ensemble de critères [Rodammer, 88]. C'est aussi programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution [Carlier, 88].

L'ordonnancement se déroule en trois étapes qui sont:

- la planification qui vise à déterminer les différentes opérations à réaliser, les dates correspondantes et les moyens matériels et humains à y affecter,
- l'exécution qui consiste à mettre en œuvre les différentes opérations définies dans la phase de planification,
- le contrôle qui consiste à effectuer une comparaison entre la planification et l'exécution, soit au niveau des coûts, soit au niveau des dates de réalisation.

Ainsi, le résultat d'un ordonnancement est un calendrier précis de tâches à réaliser qui se décompose en trois caractéristiques importantes:

- l'affectation qui attribue les ressources nécessaires aux tâches,
- le séquencement qui indique l'ordre de passage des tâches sur les ressources,
- le datage qui indique les temps de début et de fin d'exécution des tâches sur les ressources.

I.2.2. Les éléments d'un problème d'ordonnancement

Les principaux éléments d'un problème d'ordonnancement sont les tâches, les ressources, les contraintes et les critères.

Résoudre un problème d'ordonnancement consiste à programmer l'exécution des tâches et à allouer les ressources requises afin d'optimiser un ou plusieurs critères, tout en respectant un ensemble de contraintes.

I.2.2.1. *Les tâches*

La fabrication de produits dans un atelier de production nécessite l'exécution d'un ensemble d'opérations élémentaires ou tâches. Une tâche est localisée dans le temps par une date de début d'exécution t_i et/ou par une date de fin c_i et une durée d'exécution p_i. Certaines contraintes techniques ou économiques peuvent associer aux tâches des dates de début au plus tôt r_i ou des dates de fin au plus tard d_i. La figure I.1 illustre les caractéristiques temporelles d'une tâche.

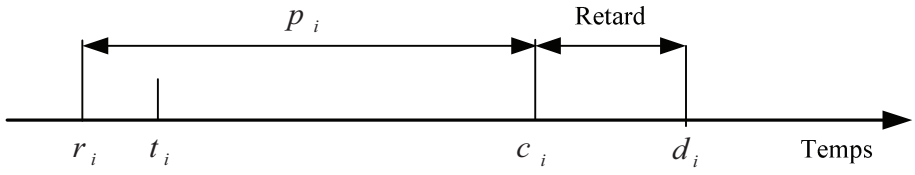

Figure I.1. Caractéristiques temporelles d'une tâche

On peut classer les tâches comme suit :

- *les tâches préemptives* qui peuvent se réaliser par morceaux facilitant ainsi la résolution de certains problèmes,
- *les tâches non-préemptives,* qui sont exécutées en une seule fois, ne peuvant pas être interrompues avant qu'elles soient complètement terminées.

I.2.2.2. ***Les ressources***

Une ressource est définie comme un moyen technique ou humain requis pour la réalisation d'une tâche. Dans un atelier, plusieurs types de ressources sont distingués :

- les ressources *renouvelables* toujours disponibles en même quantité (les hommes, les machines, l'espace, l'équipement en général, etc.), la quantité de ressources utilisables à chaque instant étant limitée,
- les ressources *consommables* ou *non renouvelables* (matières premières, budget, etc.), La consommation globale (ou cumul) au cours du temps est limitée,
- les ressources *disjonctives* (ou *non partageables*) allouées à une tâche à la fois (machine-outils, robot manipulateur),
- les ressources *cumulatives* (ou *partageables*) peuvent être utilisées par plusieurs tâches simultanément (équipe d'ouvriers, poste de travail).

I.2.2.3. ***Les contraintes***

Les contraintes sont les conditions à respecter lors de la construction d'un ordonnancement pour qu'il soit réalisable. Il existe deux classifications possibles des contraintes. La première, classique [Harrath, 03], comporte les contraintes temporelles et celles liées directement aux ressources. La deuxième présentée dans [Kacem, 03], distingue les contraintes endogènes indépendantes du système.

- Les contraintes temporelles [Harrath, 03]: Certaines opérations ne peuvent s'exécuter qu'après une date de début au plus tôt, satisfaisant ainsi l'expression suivante : $c_i - p_i \geq r_i$ et ne finissent qu'avant une date de fin au plus tard ; ce qui revient à vérifier l'intégralité suivante : $r_i + p_i \leq r_i$. Ceci exprime la non disponibilité continue des ressources et la nécessité de délivrer les produits en respectant les délais. Ce type de contraintes peut aussi définir des relations de précédence entre les opérations. La relation de précédence entre deux tâches i et j (i précède de j) s'écrit sous la forme $t_i + p_i \leq t_j$;
- Les contraintes liées aux ressources : Ces contraintes traduisent le fait que les ressources sont disponibles en quantité limitée. On distingue deux types de contraintes de ressources, liés à la nature disjonctive ou cumulative des ressources.
- Les contraintes endogènes : Elles constituent des contraintes directement liées au système de production et à ses performances dont les dates de disponibilité ou les capaci-

tés des machines et des moyens de transport et les séquences des actions à effectuer ou les gammes des produits.

- Les contraintes exogènes : Imposées extérieurement, elles sont indépendantes du système de production ; on distingue :
    - les dates de fin de fabrication au plus tard du produit, imposées généralement par les commandes,
    - les priorités de quelques commandes et de quelques clients,
    - les retards possibles accordés pour certains produits.

I.2.2.4. ***Les critères***

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi [Tangour, 07]. On peut classer les critères en critères réguliers et en critères irréguliers [Kacem, 03].

- Les critères réguliers : Ils sont dits réguliers car ils constituent des fonctions croissantes des dates d'achèvement des opérations. Nous citons, à titre d'exemple :
    - la minimisation des dates d'achèvement des actions ;
    - la minimisation du maximum des dates d'achèvement des actions ;
    - la minimisation de la moyenne des dates d'achèvement des actions ;
    - la minimisation des retards sur les dates d'achèvement des actions ;
    - la minimisation du maximum des retards sur les dates d'achèvement des actions ;
    - la minimisation de la moyenne des retards sur les dates d'achèvement des actions ;
    - la minimisation du temps du cycle (dans le cas d'un ordonnancement cyclique).
- Les critères irréguliers : Ils ne sont pas des fonctions monotones des dates de fin d'exécution des opérations. Parmi ces types de critères, on cite:
    - la minimisation des encours ;
    - la minimisation du coût du stockage des matières premières ;
    - l'équilibrage des charges des machines.

I.2.3. Typologie des ateliers

Les différents problèmes que l'on rencontre en ordonnancement dépendent principalement des machines et de l'enchaînement des opérations. On distingue trois catégories. La première regroupe les problèmes pour lesquels chaque tâche nécessite une seule machine (ressources). Dans la deuxième, chaque tâche demande plusieurs machines pour son exécution. Dans la troisième catégorie, les machines sont regroupées en étages distincts.

I.2.3.1. *Les ateliers à une ressource*

Dans certains ateliers, on ne dispose que d'une ressource pour traiter un ensemble de tâches. Celle-ci ne peut effectuer le traitement à un instant donné qu'une tâche à la fois. Ce type de problème consiste à déterminer la séquence optimale de passage de *n* tâches sur une machine, afin d'optimiser un ou plusieurs critères donnés.

Parmi les problèmes rencontrés dans un environnement à une machine, on peut citer les problèmes de minimisation du retard maximum, du nombre de tâches en retard, ou de la somme des retards.

Malgré leur apparente simplicité, les problèmes à une machine sont NP-difficiles au sens fort [Lenstra, 77], [Garey, 79].

I.2.3.2.*Les ateliers à plusieurs ressources*

Ces ateliers sont composés d'un ensemble de *m* machines (ou ressources) dont chacune ne peut réaliser, à elle seule, l'ensemble des opérations. Suivant le mode de passage sur les différentes machines, trois types d'ateliers sont distingués.

I.2.3.2.1. Flowshop

Dans un tel atelier, appelé aussi atelier en ligne ou à cheminement unique, toutes les opérations passent par toutes les machines dans le même et unique ordre. La figure I.2 illustre un atelier de type flowshop : l'ensemble des produits initiaux {Pi} passe successivement sur toutes les machines M1…M5 pour donner un ensemble de produits finis {Pf}.

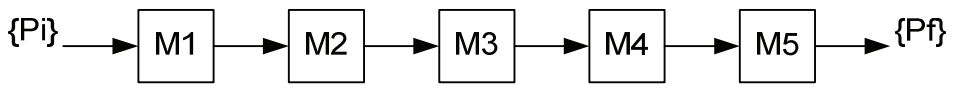

Figure I.2. Atelier Flowshop

I.2.3.2.2. Jobshop

Dans cette classe d'ateliers, appelés aussi ateliers à cheminements multiples, chaque tâche possède son propre mode de passage sur les machines. La figure I.3 présente un atelier de type job shop.

Le Jobshop flexible est une extension du modèle jobshop classique. Sa principale particularité réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations. Une opération est associée, plus précisément à un ensemble contenant toutes les machines pouvant effectuer cette opération.

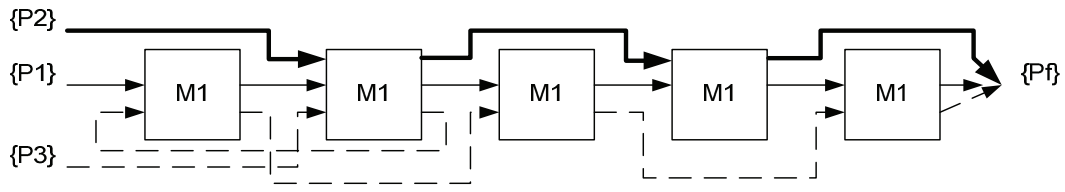

Figure I.3. Atelier Jobshop

I.2.3.2.3. Open shop

C'est un atelier à cheminement libre, l'ordre des opérations n'étant pas fixé a priori; le problème d'ordonnancement consiste, d'une part, à déterminer le cheminement de chaque produit et, d'autre part, à ordonnancer les produits en tenant compte des gammes trouvées, ces deux problèmes pouvant être résolus simultanément. Ce type d'ateliers n'est pas couramment utilisé dans les entreprises.

I.2.3.3. ***Les ateliers à ressources parallèles***

Dans ces ateliers, les machines sont regroupées en étages distincts. Chaque machine appartient à un seul étage et toutes les machines d'un même étage peuvent effectuer la même opération.

Ce type d'atelier peut être divisé en trois sous-catégories selon la vitesse d'exécution des machines :

- les ateliers à machines identiques : toute tâche peut s'exécuter sur n'importe quelle machine avec une même durée opératoire,
- les ateliers à machines uniformes : chaque machine possède sa propre vitesse, indépendamment de la durée de la tâche à exécuter,
- les ateliers à machines indépendantes : la vitesse de chaque machine dépend de l'opération à effectuer.

I.2.4. Complexité des problèmes d'ordonnancement

La difficulté des problèmes d'ordonnancement peut être spécifiée en fonction des données dont le nombre de machines et le nombre de tâches. Elle peut être définie aussi suivant la complexité des méthodes de résolution et celle des algorithmes utilisés [Garey, 79], [Carlier, 84] [Carlier, 88], [Charon, 96]. On distingue deux types de complexité: algorithmique et problématique.

I.2.4.1.*Complexité algorithmique*

Le temps et l'espace mémoire sont les paramètres déterminant la performance des algorithmes pouvant résoudre un problème donné. La performance d'un algorithme se mesure par rapport au temps de calcul nécessaire.

Définition 1 *: On appelle complexité en temps d'un algorithme la fonction* $f(n)$ *qui représente le nombre maximum d'opérations élémentaires effectuées pour résoudre un problème de taille n (n étant le nombre de variables décrivant le problème). On associe ainsi une unité de temps à chaque opération élémentaire.*

Définition 2 *: On dit que* $f(n) \in O(g(n))$*, s'il existe une constante* $c \succ 0$ *et un entier* n_0 *tels que* $\forall n \geq n_0, |f(n)| \leq c|g(n)|$.

Définition 3 *: Un algorithme est dit polynômial si sa fonction de complexité* $f(n) \in O(p(n))$*, p étant un polynôme en n, c'est-à-dire, s'il existe une constante* $k \succ 0$ *telle que* $f(n) \in O(n^k)$.

Si la fonction de complexité f n'est pas majorée par un polynôme, on dit alors que c'est une fonction Non Polynomiale (NP) ou exponentielle.

I.2.4.2.*Complexité problématique*

La complexité problématique dépend de la difficulté du problème à résoudre et du nombre des opérations élémentaires qu'un algorithme peut effectuer pour trouver l'optimum en fonction de la taille du problème.

Selon son degré de complexité, un problème peut appartenir à l'une des quatre classes suivantes [Sakarovitch, 84] :

- *les problèmes les plus difficiles* : sont des problèmes pour lesquels il n'existe aucune méthode de résolution; ils sont dits indécidables,

- *les problèmes de la classe P* : Ce sont des problèmes polynomiaux s'il existe un algorithme de complexité polynomiale pour leur résolution,
- *les problèmes de la classe NP* : Ce sont des problèmes NP-difficiles, qui ne peuvent à priori être résolus en un temps polynomial que par des méthodes approchées (heuristiques),
- *les problèmes NP-Complets* : un problème de décision A est dit NP-Complet s'il appartient à la classe NP et si pour tout A′ de NP :
    - il existe une application polynomiale qui transforme toute instance I′ de A′ en une instance I de A,
    - A′ admet une réponse "oui" pour l'instance I′, si et seulement si A admet une réponse "oui" pour l'instance I.

Autrement dit, s'il existe un algorithme polynomial pour résoudre A, alors, pour tout le reste des problèmes de la classe NP-Complets, il existe des algorithmes polynomiaux pour les résoudre.

I.2.5. Modélisation et représentation des ordonnancements

I.2.5.1. *Modélisation*

✓ *Les méthodes mathématiques*

Dans le cas des méthodes mathématiques, les données, les contraintes et la fonction d'évaluation des critères sont écrites sous forme d'équations et d'inéquations mathématiques. Ces méthodes, couramment utilisées, ont l'avantage d'être simples et directement exploitables par les algorithmes de résolution [Lopez, 99].

✓ *Les méthodes graphiques*

Un problème d'ordonnancement peut être modélisé par un graphe disjonctif (ou graphe potentiel – tâches) constitué :

- de nœuds représentant les tâches,
- d'arcs conjonctifs qui connectent deux opérations, consécutives d'un même travail, décrivant ainsi la contrainte de précédence liant ces opérations,
- d'arcs disjonctifs qui connectent deux opérations appartenant à des travaux différents qui utilisent la même machine indiquant ainsi les contraintes de ressources.

Chaque arc est pondèré par la durée de l'opération de l'extrême droite. La figure I.4 représente le graphe disjonctif d'un problème d'ordonnancement de 7 tâches sur 3 ressources, tableau I.1, les contraintes sur les ressources étant représentées par des arêtes pointillées.

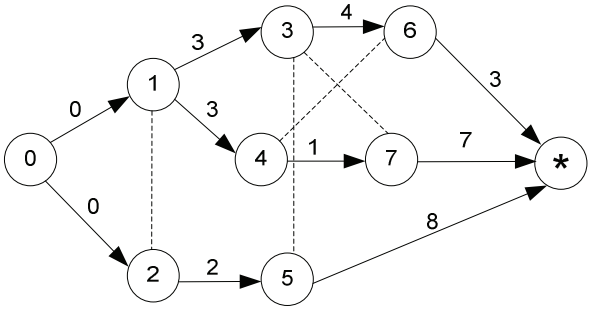

Figure I.4. Graphe Potentiel-Tâche

Tableau I. 1 Données d'un problème d'ordonnancement

Tâches	Durée opératoire	Contrainte de précédence	Ressources
1	3	-	M1
2	2	-	M1
3	4	1	M2
4	1	1	M3
5	8	2	M2
6	3	3	M3
7	7	4	M2

I.2.5.2. ***Représentation des solutions***

Le diagramme de Gantt, du nom de son développeur Henry Gantt, est un outil permettant de modéliser la planification des tâches nécessaires à la réalisation d'un projet. Etant donné la facilité relative de lecture des diagrammes de Gantt, cet outil est utilisé à la fois dans la littérature et par la quasi-totalité des chefs de projets dans tous les secteurs. Il permet de représenter graphiquement l'avancement du projet et constitue également un bon moyen de communication entre les différents acteurs d'un projet. Le diagramme de Gantt présente en ordonnées la liste des tâches, notées Ti à exécuter par les machines, notées Mj, et en abscisses l'échelle du temps, comme le montre la figure I.5, dans le cas du problème considéré dans le paragraphe précédent.

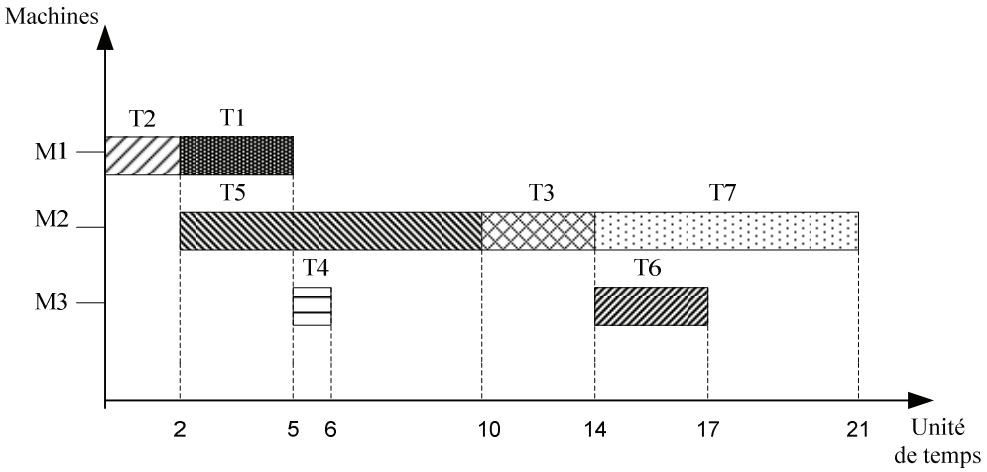

Figure I.5. Diagramme de Gantt

I.2.5.3. ***Notations***

Dans la littérature, un système de notation a été proposé [Gar79], [Bru94] pour représenter un problème d'ordonnancement d'une manière simple. Il consiste à représenter le problème par trois champs α, β et γ $(\alpha/\beta/\gamma)$:

- le champ α, permettant de décrire le type d'atelier, le nombre de jobs à réaliser et le nombre de machines disponibles,
- le champ β, caractérisant les conditions d'exécution des jobs ainsi que les états des ressources présentes dans l'atelier. Il indique en particulier la présence ou non des contraintes de précédence entre les tâches et la possibilité de tolérer la préemption,
- le champ γ, dédié au(x) critère(s) à optimiser.

Pour l'exemple suivant :

$$J, 10, 6|Prec, r_i|C_{max}$$

il s'agit d'un problème d'ordonnancement d'un atelier de type job-shop, ayant dix jobs et six machines disponibles. Le deuxième champ indique que les jobs présentent une contrainte de précédence, *Prec*, et une contrainte r_i de dates de début au plus tôt. En plus, la préemption est interdite (puisqu'elle n'est pas mentionnée dans le champ *prem*). Le dernier champ montre que l'objectif est de minimiser le makespan, C_{max}.

I.3. Problèmes d'optimisation et méthodes de résolution

I.3.1. Problèmes d'optimisation

Les problèmes d'ordonnancement sont définis, en général, comme des problèmes d'optimisation puisque leur objectif est de minimiser (ou maximiser) une fonction objectif tout en respectant certains critères. Ce besoin d'optimisation vient de la nécessité de fournir à l'utilisateur un système qui réponde au mieux à ses attentes. Mais parfois, il s'avère nécessaire d'optimiser plusieurs fonctions, et de recourir, dans ce cas, à une optimisation multi-objectifs.

I.3.1.1. *Formulations*

Un problème d'optimisation se présente de la façon suivante :

$$\begin{cases} \text{minimiser} & f(\vec{x}) & \vec{x} \in \mathbb{R}^n \\ \text{avec} & \vec{g}(\vec{x}) \leq 0 & \vec{g}(\vec{x}) \in \mathbb{R}^m \\ \text{et} & \vec{h}(\vec{x}) = 0 & \vec{h}(\vec{x}) \in \mathbb{R}^p \end{cases}$$

où f est la fonction objectif, le vecteur $\vec{x}$ regroupe les variables de décision qui sont les paramètres d'optimisation. L'ensemble des conditions $\vec{h}(\vec{x}) = 0$ représente les contraintes d'égalités du problème.

L'ensemble des conditions $\vec{g}(\vec{x}) \leq 0$ représente les contraintes d'inégalités du problème.

Il existe deux types d'extrema: les extrema locaux et les extrema globaux.

- *Minimum global*

Un point $\overrightarrow{x^*}$ est un minimum global de la fonction *f*, si on a :

$f(\overrightarrow{x^*}) \prec f(\vec{x})\ \forall \vec{x}$, tel que $\overrightarrow{x^*} \neq \vec{x}$

- *Minimum local*

Un point $\overrightarrow{x^*}$ est un minimum local de la fonction *f*, si on a :

$f(\overrightarrow{x^*}) \leq f(\vec{x})\ \forall \vec{x} \in V(\overrightarrow{x^*})$ et $\overrightarrow{x^*} \neq \vec{x}$

$V(\overrightarrow{x^*})$ définissant un voisinage de $\overrightarrow{x^*}$

La figure I.7 illustre le cas d'une fonction à cinq minima locaux, *M1, M2, M4, M5* et *M6*, et un minimum global, *M3*, sur un intervalle d'étude.

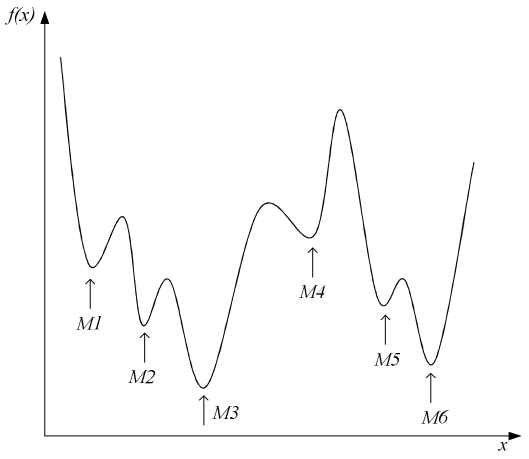

Figure I.6. Exemple de différents minima

I.3.1.2. ***Optimisation mono-objectif***

Le problème à traiter ne possède, dans ce cas, qu'un objectif unique à optimiser. Les objectifs les plus régulièrement considérés sont la minimisation de la durée totale de l'ordonnancement, le respect des dates au plus tard, la minimisation des retards, la minimisation d'un coût, la minimisation de la date effective de fin de toutes les opérations, …

I.3.1.3. ***Optimisation multi-objectifs***

Dans le cas d'une optimisation multi-objectifs, on chercher souvent à satisfaire plusieurs objectifs. D'un point de vue mathématique, un tel problème, dans le cas où f regroupe k fonctions objectif, se présente de la façon suivante [Collette, 02]:

$$(P1)\ \begin{cases} \text{Minimiser } \vec{f}(\vec{x}), \quad \vec{x} \in \Re^n, \quad \vec{f}(\vec{x}) \in \Re^k \text{ (fonction à optimiser)} \\ \text{avec } \quad \vec{g}(\vec{x}) \le 0, \quad \vec{g}(\vec{x}) \in \Re^m \text{ (m contraintes d'inégalités)} \\ \text{et } \quad \vec{h}(\vec{x}) = 0, \quad \vec{h}(\vec{x}) \in \Re^p \text{ (p contraintes d'égalités)} \end{cases}$$

Il existe plusieurs méthodes pour résoudre ces problèmes, on peut, par exemple, simplifier les différentes fonctions objectif pour faciliter leur traitement [Collette, 02].

I.3.2. Méthodes de résolution

La principale difficulté à laquelle est confronté un décideur, en présence d'un problème d'optimisation est celui du choix d'une méthode efficace capable de produire une solution optimale en un temps de calcul raisonnable.

Dans la littérature, on distingue essentiellement deux classes de méthodes : les méthodes exactes assurant la résolution des problèmes en un temps polynômial et les méthodes approchées (ou heuristiques) permettant de trouver une solution proche de l'optimale en un temps tolérable [Harrath, 03].

I.3.2.1.*Méthodes exactes*

Ces méthodes sont généralement utilisées pour résoudre des problèmes de petite taille. Dans ce cas, le nombre de combinaisons possibles est suffisamment faible pour pouvoir explorer l'espace de solutions en un temps raisonnable. On distingue trois sous-classes de méthodes exactes [Kacem, 03] : la procédure de séparation et d'évaluation (Branch and Bound) [Le Pape, 95], [Baptiste, 96], la programmation dynamique [Bellman, 86] et la programmation linéaire [Sakarovitch, 84].

I.3.2.2.*Méthodes approchées*

La résolution d'un problème d'optimisation combinatoire, de taille comparable à ceux rencontrés dans la pratique, se heurte à des tailles mémoire et des temps de calcul trop importants. L'objectif n'est plus alors d'obtenir systématiquement l'optimum mais plutôt d'obtenir une solution proche de l'optimum ou de « bonne qualité » en un temps minimal. Ainsi, au lieu d'effectuer une recherche exhaustive, les méthodes approchées échantillonnent l'espace de recherche, n'en considèrent qu'une partie, et fournissent ainsi, en un temps raisonnable, la meilleure configuration rencontrée.

On distingue deux types de méthodes : les heuristiques et métaheuristiques.

I.3.2.2.1. Les heuristiques

Les heuristiques sont des méthodes empiriques basées sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de ces méthodes est d'intégrer des stratégies de décision pour construire une solution proche de l'optimum, tout en essayant de l'obtenir en un temps de calcul raisonnable [Bel, 01], [Lopez, 01], [Kacem, 03].

On distingue [Chu, 96] :

- **FIFO** (First In First Out) : la première tâche qui vient est la première tâche ordonnancée,

- **SPT** (Shortest Processing Time) : la tâche ayant le temps opératoire le plus court est traitée en premier lieu,

- **LPT** (Longest Processing Time) : la tâche ayant le temps opératoire le plus important est ordonnancée en premier lieu,

- **EDD** (Earliest Due Date) : la tâche ayant le date due la plus petite est la plus prioritaire,

- **SRPT** (Shortest Remaining Processing Time) : cette règle, servant à lancer la tâche ayant la plus courte durée de travail restant à exécuter, est très utilisée pour minimiser les encours et dans le cas des problèmes d'ordonnancement préemptifs,

- **ST** (Slack Time) : à chaque point de décision, l'opération ayant la plus petite marge temporelle est prioritaire. Faute de disponibilité des ressources de production, cette marge peut devenir négative.

I.3.2.2.2. Les métaheuristiques

Les métaheuristiques, telles que le Recuit Simulé [Kirkpatrick, 83], la Recherche Tabou [Glover, 89], les Algorithmes Génétiques [Holland, 75], les Algorithmes de Colonies de Fourmis [Colorni & al, 91], [Colorni & al, 92], constituent des méthodes générales de recherche dédiées aux problèmes d'optimisation difficiles [Tangour, 08], [Borne, 10]. Elles sont, en général, présentées sous forme de concepts.

a. Le Recuit Simulé

Le Recuit Simulé (Simulated Annealing) est une métaheuristique basée sur la recherche locale qui a prouvé son efficacité grâce à des résultats pratiques obtenus sur de nombreux problèmes NP-difficiles [Dréo, 03] [Karray, 07].

Le recuit simulé [Kirkpatrick, 83] trouve ses origines dans le phénomène thermodynamique de recuit des métaux. Cette méthode imite une procédure utilisée par les métallurgistes qui, pour obtenir un alliage exempt de défauts, chauffent d'abord à blanc leur morceau de métal et laissent ensuite l'alliage se refroidir très lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement. Cependant, lorsque la baisse progressive de la tempéra-

ture est trop rapide (comme pour la méthode de la Trempe), le solide présente alors des défauts, et un minimum local d'énergie est obtenu.

Se basant sur l'évolution d'un système thermodynamique, l'algorithme du recuit simulé accepte, pour sortir d'un minimum local, une dégradation de la fonction de coût avec une certaine probabilité.

Algorithme: Recuit Simulé

Début

Déterminer aléatoirement une solution initiale s_0 et une température initiale T_0 ;

Poser $s_{\min} = s_0$ et $T_i = T_0$;

Calculer $f(s_0)$;

Répéter pour chaque itération *i/ i=1,..., n*

Choisir $s' \in N(s_i)$;

Calculer $\Delta f = f(s') - f(s_i)$;

Si $\Delta f \prec 0$ **alors** $s_{\min} = s_i$;

Sinon

Trier p dans [0, 1] suivant une distribution uniforme ;

Si $p \leq e^{(-\Delta f / T)}$ **alors** $s_{\min} = s_i$;

Sinon s_i est rejeté ;

Jusqu'à T_i est proche de 0

Calculer $T = \gamma T$;

Retourner $s_{\min}$;

Fin

Figure I.7. Algorithme général du Recuit Simulé

L'algorithme, illustré par la figure I.7, se base sur deux résultats de la physique statistique [Dréo, 03].

1) Lorsque l'équilibre thermodynamique est atteint à une température donnée *T*, la probabilité, pour un système physique, de posséder une énergie donnée *E*, est proportionnelle au facteur de Boltzmann : $\exp^{(-E/K_B T)}$, K_B étant la constante de Boltzmann.
2) Le deuxième résultat s'appuie sur l'algorithme de Metropolis [Metropolis & al, 53] pour simuler l'évolution d'un système physique vers son équilibre thermodynamique à une température *T* donnée. Le principe de cet algorithme est le suivant : partant d'une configuration donnée, nous faisons subir au système une modification élémentaire ; si cette transformation a pour effet de diminuer la fonction objectif *f* (ou « énergie ») du

système, elle est acceptée ; si elle provoque au contraire une augmentation Δf de la fonction objectif, elle est acceptée tout de même, avec la probabilité $\exp^{(-f/K_B T)}$.

Cette méthode est présentée, avec plus de détails, dans le troisième chapitre de ce mémoire.

b. La Recherche Tabou

La Recherche Tabou (Tabu Search) est une métaheuristique originalement développée par Glover [Glover, 89], [Glover, 90]. Elle a été appliquée avec succès pour résoudre de nombreux problèmes NP-difficiles, comme les problèmes de routage de véhicules, les problèmes d'affectation quadratique, aussi les problèmes d'ordonnancement et les problèmes de coloration de graphes.

Le principe de la Recherche Tabou est similaire à celui des méthodes de recherche locales itératives. Il est basé sur la création et l'évaluation d'un voisinage de solutions [Dréo, 03], [Kammarti, 05], [Karray, 08].

Comme le Recuit Simulé, la Recherche Tabou fonctionne avec une seule configuration courante qui est actualisée au cours des itérations. A chaque itération, figure I.8, le mécanisme de passage d'une configuration à une autre comporte deux étapes :

- la construction d'un ensemble de solutions voisines de la solution courante,
- l'évaluation de la fonction objectif *f* du problème en chacune des configurations appartenant à l'ensemble de solutions voisines afin de choisir la meilleure solution, même si ce choix entraîne une augmentation de la fonction objectif à minimiser.

En acceptant de détériorer la valeur de la solution courante, le minimum local peut être évité mais, en contre partie, des parcours répétitifs sont évités.

Aussi, pour éviter la présence de cycle, la Recherche Tabou utilise une mémoire afin de conserver pendant un moment les informations sur les solutions déjà visitées.

Cette mémoire constitue la liste Tabou, ainsi, une nouvelle solution n'est acceptée que si elle n'appartient pas à cette liste. Ce critère d'acceptation d'une nouvelle solution évite les cycles et dirige l'exploration de la méthode vers des régions du domaine de solutions non encore visitées.

Un *critère d'aspiration,* également utilisé, permet de lever l'interdiction et de revenir à une solution déjà visitée pour redémarrer la recherche dans une autre direction. Cette idée est développée dans [Glover, 90], [Glover, 95] et [Glover, 97].

Algorithme : Recherche Tabou

Début

Poser x =solution initiale aléatoire

Poser $x_{\min} = x$, $f_{\min} = f(x)$ et $TL = \varnothing$ (TL : liste tabou);

Répéter

Générer un sous-ensemble N de voisinage tel que :

$s_i(x) \subset S(x)$ et $(x, s_i(x)) \notin TL$;

Trouver $\underset{1 \le i \le N}{f}(s(x)) = \min\left[f(s_i(x))\right]$;

Ajouter $\{x, s_i(x)\}$ dans TL ;

Remplacer x par $s(x)$ tel que $s(x)$ est la meilleure solution de $s_i(x)$;

Si $f(x) \prec f_{\min}$ **alors**

$f_{\min} = f(x)$

$x_{\min} = x$

Jusqu'à conditions d'arrêt satisfaites ;

Fin

Figure I.8. Algorithme général de la Recherche Tabou

L'intensification et la diversification sont deux stratégies développées afin d'améliorer l'efficacité de la méthode tabou [Widmer, 01].

- *L'intensification* consiste à explorer en détails une région de l'espace de recherche, jugée prometteuse. Sa mise en œuvre consiste, le plus souvent, en un élargissement temporaire du voisinage de la solution courante dans le but de visiter un ensemble de solutions partageant certaines propriétés.

- *La diversification* a pour objectif de diriger la procédure de recherche vers des régions inexplorées de l'espace de recherche. La stratégie de diversification la plus simple consiste à redémarrer périodiquement le processus de recherche à partir d'une solution, générée aléatoirement ou choisie judicieusement, dans une région non encore visitée de l'ensemble des solutions admissibles.

Cette méthode est présentée avec plus de détails dans le troisième chapitre de ce mémoire.

c. Les algorithmes génétiques

Selon l'une des théories de l'évolution et de la sélection naturelle (basée sur le néodarwinisme de Charles Darwin, XIX^{ème} siècle), les caractéristiques des êtres vivants se modifient progressivement, lors de la phase de reproduction, sous l'influence des conditions extérieures.

Les générations successives d'individus s'adaptent de mieux en mieux aux conditions complexes de leur environnement, maximisant ainsi leur probabilité de survie [Dupont, 05]. Pour passer d'une génération à la suivante, deux mécanismes fondamentaux ont été identifiés pour créer un nouvel individu à partir de ses parents :

- le croisement, qui consiste à combiner deux moitiés du patrimoine génétique de chacun des parents pour constituer le patrimoine génétique de l'enfant ;
- la mutation, ou la modification spontanée de quelques gènes de l'enfant.

Le nouvel individu ainsi créé est différent de chacun de ses parents mais partage certaines de leurs caractéristiques. Si le hasard fait que l'enfant hérite de «bonnes» caractéristiques, son espérance de vie est plus élevée et il aura, par conséquent, de plus grande chance de se reproduire.

L'analogie entre cette théorie de l'évolution et une métaheuristique pour l'optimisation combinatoire a été proposée par Holland en 1975 [Holland, 75].

Ce qui distingue les algorithmes génétiques des autres méthodes, peut être formulé selon quatre axes principaux [Lerman, 95] :

- les algorithmes génétiques utilisent un codage des paramètres, et non les paramètres eux-mêmes,
- les algorithmes génétiques travaillent sur une population de points, au lieu d'un point unique,
- les algorithmes génétiques n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée, ou une autre connaissance auxiliaire,
- les algorithmes génétiques utilisent des règles de transition probabilistes, et non déterministes.

L'utilisation d'un algorithme génétique nécessite la définition, au préalable, d'un espace de recherche dont les éléments de base sont les chromosomes et d'une fonction définie sur cet

espace (fonction fitness) dont la valeur optimale est évaluée en rapport avec les opérateurs de croisement et de mutation choisis [Iyer, 04].

L'enchaînement de ces différents éléments de l'algorithme est présenté dans la figure I.9. Cette méthode est présentée avec plus de détails dans le second chapitre de ce mémoire.

Algorithme : Algorithmes Génétiques

Début
  Génération d'une population aléatoire initiale de *k* individus ;
  Répéter
    Evaluation de la fonction objectif *f* pour chaque individu ;
    Sélection des meilleurs individus ;
    Croisement des individus sélectionnés ;
    Mutation de certains individus de la population ;
  Jusqu'à conditions d'arrêt satisfaites ;
  Retourner la meilleur solution ;
Fin

Figure I.9. Les Algorithmes Génétiques

I.4. Amélioration des méthodes de résolution - Position du problème

La plupart des problèmes industriels sont des problèmes complexes. Le nombre de solutions croît exponentiellement avec la taille du problème. Les méthodes exactes deviennent rapidement inutilisables pour des instances de grandes tailles. Il s'avère, ainsi, nécessaire d'utiliser les méthodes approchées pour résoudre ces problèmes en un temps raisonnable. Les méthodes approchées, et plus exactement les métaheuristiques, dont les algorithmes génétiques [Altay, 95] [Fonseca, 95], [Deb, 01], qui offrent l'avantage de ne parcourir qu'une faible fraction de l'espace de recherche pour parvenir à une solution acceptable.

Plusieurs méthodes ont été mises en œuvre pour le traitement de problèmes d'optimisation multiobjectif. Ces méthodes se décomposent en deux familles [collette, 02].

La première famille comporte les méthodes « agrégatives » dont le but est de se ramener à un problème d'optimisation mono-objectif en fusionnant les différentes fonctions objectif [Boukef, 09].

La deuxième famille comporte les méthodes « non agrégatives » [Kacem, 02]. Contrairement à la première famille, l'objectif ici n'est pas de se ramener à un problème d'optimisation mono-objectif.

Notre idée rentre dans ce cadre et consiste à développer une nouvelle approche pour la résolution d'un problème d'optimisation multiobjectif, utilisant les algorithmes génétiques pour la

résolution de ce type de problème. Pour cela, trois nouvelles approches basées sur les algorithmes génétiques sont, développées dans le deuxième chapitre.

La diversité des méthodes de résolution exactes ou approchées et les avantages qu'elles possèdent séparément, permettent d'envisager de les utiliser conjointement pour améliorer les résultats obtenus ou pour résoudre des problèmes complexes. Les techniques de l'intelligence artificielle sont basées sur le raisonnement et la logique formelle, alors que les techniques de la recherche opérationnelle adoptent un mode plus calculatoire. Il semble donc intéressant de faire coopérer ou d'hybrider différentes méthodes afin de combiner leurs qualités complémentaires et de mettre en œuvre des modèles hybrides basés sur la collaboration de plusieurs méthodes de résolution.

Nos travaux visent à dégager des résultats concernant l'utilisation des modèles hybrides dans le cadre de résolution des problèmes d'ordonnancement. Plus que les performances elles-mêmes, nous nous intéressons tout particulièrement à l'analyse de l'influence de la coopération de plusieurs méthodes de recherches sur la qualité des solutions engendrées.

Pour cela, trois méthodes sont utilisées: les Algorithmes Génétiques (AGs), la Recherche Tabou (RT) et le Recuit Simulé (RS).

I.5. Conclusion

Au cours de ce chapitre, après la présentation des problèmes d'ordonnancement et de leurs principales caractérisations, les méthodes exactes de résolution, à l'instar de la programmation linéaire et du Branch and Bound, sont définies.

Nous avons aussi introduit les méthodes approchées et plus exactement les métaheuristiques dont le Recuit Simulé, la Recherche Tabou et les Algorithmes Génétiques.

La résolution d'un problème d'ordonnancement en industries agroalimentaires par l'utilisation d'une nouvelle approche des algorithmes génétiques est proposée dans le prochain chapitre et les résultats obtenus seront comparés avec ceux fournis par les méthodes hybrides présentées dans le troisième chapitre.

CHAPITRE II

METHODES BASEES SUR LES ALGORITHMES GENETIQUES POUR LA RESOLUTION DE PROBLEMES D'ORDONNANCEMENT

II.1. Introduction

L'ordonnancement est un champ d'investigation qui a connu un essor important ces quarante dernières années [Blazewicz, 96], [Brucker, 98], [Lopez, 01], tant par les nombreux problèmes identifiés que par l'utilisation et le développement de nombreuses techniques de résolution.

Les différents problèmes que l'on rencontre en ordonnancement dépendent principalement des machines et de l'enchaînement des opérations. Parmi ces problèmes, on distingue les problèmes d'ordonnancement à une machine ; Ceux-ci ont été utilisés comme modèle pour les autres catégories de problèmes et se sont avérés efficaces en pratique. Ainsi, plusieurs travaux se sont intéressés à l'étude des problèmes d'ordonnancement à une machine [Ying, 09].

Malgré leur apparente simplicité, les problèmes à une machine sont NP-difficiles [Lenstra, 77], [Garey, 79], [Carlier, 87]. Plusieurs méthodes d'optimisation, parmi lesquelles les méthodes exactes et les méthodes approchées, ont été utilisées pour résoudre les problèmes d'ordonnancement.

La programmation linéaire, la programmation dynamique et la méthode de « Branch & Bound » constituent les méthodes exactes qui se sont avérées peu efficaces pour la résolution des problèmes de grandes tailles, alors que les méthodes approchées telles que les métaheuristiques ont conduit à une solution proche de l'optimum [Borne, 07].

Les métaheuristiques ont prouvé leur efficacité pour la résolution de problèmes d'ordonnancement à une machine. Parmi ces métaheuristiques, on cite le recuit simulé [Jin, 09], la recherche tabou [Choobineh, 06], les algorithmes génétiques [Chou, 09], les colonies de fourmis [Hassine, 08] et les essaims particulaires [Anghinolfi, 09].

Les approches de résolution, proposées dans ce chapitre, sont basées sur l'utilisation des algorithmes génétiques qui, initialement développés par Holland [Holland, 75], constituent des algorithmes d'exploration fondés sur les mécanismes de la sélection naturelle et de la génétique.

Ils se sont illustrés par une grande efficacité face aux problèmes d'ordonnancement tels que le flow shop [Etiler, 04], le job shop [Mesghouni, 04] et l'open shop [Liaw, 00]. Aussi, ils ont

été largement utilisés, ces dernières années, pour la résolution de problèmes d'ordonnancement à une machine [Gupta, 93], [Mayer, 99], [Armentano, 00], [Madureira, 01], [Koksalan, 03], [Chang, 06], [Chang, 08], [Chou, 09].

Dans ce chapitre, un problème d'ordonnancement multi-objectifs d'un atelier de production agroalimentaire à une machine est présenté. Les différentes caractéristiques et les opérations le concernant sont introduites puis sa formulation et sa résolution par de nouvelles approches basées sur les algorithmes génétiques sont proposées.

II.2. Particularités de l'ordonnancement dans un atelier de production agroalimentaire

Avec la distribution et la restauration, les industries agroalimentaires constituent un intermédiaire indispensable entre les producteurs agricoles et les consommateurs finaux. Ces industries sont très dépendantes de l'évolution des habitudes de consommation et des différentes réglementations internationales. Les entreprises agroalimentaires doivent elles-mêmes évoluer en terme d'emploi : effectifs, structure de qualification, compétences, etc. L'évolution de ces entreprises dépend, essentiellement, des facteurs suivants [Gargouri, 03] :

- un facteur externe qui est directement lié au contexte sectoriel et qui dépend des évolutions, qualitative et quantitative, du marché, des circuits de distribution et des innovations technologiques,
- un facteur interne qui dépend de la stratégie ou de la politique de l'entreprise qui identifie les objectifs souhaités,
- un autre facteur interne qui concerne essentiellement les choix de l'entreprise en matière d'organisation de travail et de coordination entre les postes fonctionnels et opérationnels.

II.2.1. Les produits

Le produit mis à la disposition du consommateur passe du composant primaire, au produit semi-fini jusqu'au produit fini.

II.2.1.1. *Les composants primaires*

La fabrication d'un produit agroalimentaire fini P_i, composé par n opérations, est caractérisé par :

- r_i : la date de mise en fabrication au plus tôt,
- d_i: la date de fin de fabrication au plus tard,
- g_i : la gamme définissant l'ensemble des opérations à réaliser.

Ce produit est également défini par une nomenclature qui détermine les composants primaires nécessaires pour l'obtenir. Ces composants, souvent représentés par des produits agricoles et des matières crues périssables, ont des durées de vie limitées et sont transformés au fur et à mesure de la production.

Pour éviter la péremption des matières premières, chaque composant c_{ijk} doit être consommé avant sa date limite de validité V_{ijk} .

II.2.1.2. ***Les produits semi-finis***

A cause des risques de contamination, les produits semi-finis manipulés dans les industries agroalimentaires ont généralement des durées de vie très courtes. Ces produits doivent être transférés sur les postes de travail dans des délais très courts. En effet, la synchronisation entre les différents postes de travail de la ligne de fabrication est nécessaire pour éviter la péremption en ligne.

II.2.1.3. ***Les produits finis***

Contrairement à d'autres produits industriels, les produits agroalimentaires ont une durée de vie (*Dv*) bien déterminée à compter de leurs dates de fin de fabrication (*Dff*). A chaque produit fini, est alors associée une date limite de consommation (*Dlc*). Il est à noter que la vente de ces produits passe souvent par des surfaces de distribution qui imposent certaines conditions aux manufacturiers, à savoir :

- Le produit étant considéré invendable à partir d'un certain délai de sa date limite de consommation, appelé délai de retour (*Dr*), le responsable de la production doit alors, en cas de dépassement, reprendre son produit.

Considérons par exemple, figure II.1, le cas d'un produit fini ayant une durée de 21 jours. Si son délai de retour est fixé à 5 jours, ce produit doit être repris par le manufacturier 5 jours avant sa date limite de consommation [Gargouri, 03].

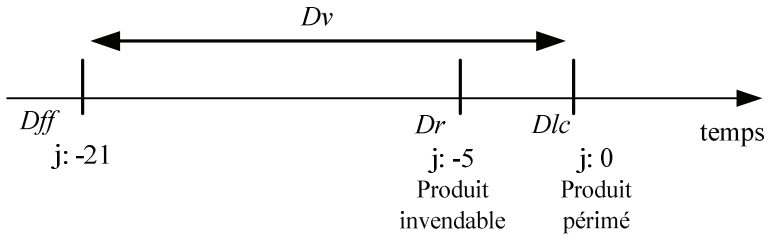

Figure II.1. Cycle de vie d'un produit agroalimentaire

- Le prix de vente du produit, noté P^{ven}, se dégrade proportionnellement à la dégradation de sa durée de vie avant la livraison. Le prix de vente est aussi diminué d'une valeur appelée discount de distribution Δ_i par jour de stockage, avant sa livraison aux distributeurs, on a :

$$\Delta_i = \frac{P^{ven}}{DV_{ij}} \Big/ jour\ de\ stockage \qquad \text{(II.1)}$$

(DV_{ij}) étant la durée de vie pouvant être définie pour chaque opération O_{ij} à compter de la date courante d_c :

$$DV_{ij} = \max(0, (Dff_{ij} - d_c)) \qquad \text{(II.2)}$$

II.2.2. Spécificités d'un problème d'ordonnancement en industries agro-alimentaires

La nature des produits et des procédés ainsi que les contraintes de qualité et de contrôle des coûts font que les besoins industriels de l'agroalimentaire en matière de gestion de données techniques sont assez différents de ceux des entreprises manufacturières [Treillon, 96], [Franchini, 98], [Gargouri, 02]. En effet, il faut tenir compte de ces particularités lors de la résolution du problème d'ordonnancement. Contrairement aux autres industries, la succession des opérations dans un même poste de travail dans un atelier de production agroalimentaire est caractérisée, généralement, par des opérations de nettoyage de durées variables qui peuvent interrompre le cycle de fonctionnement de la production et peuvent donc provoquer des retards.

Ces opérations, engendrant des temps improductifs assez importants, nécessitent l'optimisation des changements de recettes et des gammes de produits. D'autres opérations de nettoyage de durées d'exécution plus importantes, sont d'autre part, imposées par le service qualité pour éviter la contamination des ressources et assurer ainsi une qualité satisfaisante.

Dans ce type d'industrie, les produits semi-finis, résultant des opérations élémentaires du processus de fabrication, peuvent être modifiés et ne sont pas forcément ceux qui ont été planifiés ; de ce fait, l'évolution dans le temps d'une opération en cours de réalisation n'est pas toujours déterministe (les caractéristiques des composants : acidité, concentration, etc. ou encore les conditions de production : température, taux d'humidité, etc.). Ces produits peuvent être complètement perdus si aucune action corrective n'est possible. Des décisions d'interruptions des opérations en cours ou d'engagements d'autres opérations, peuvent avoir lieu dans de telles conditions. On est donc face à un problème d'ordonnancement dynamique. Un outil réactif d'aide à la décision en temps réel peut constituer alors la meilleure solution pour résoudre ce type de problème.

Le tableau II.1 relève quelques points de différences entre les données d'un ordonnancement classique [Gargouri, 03] et celles d'un ordonnancement en industries agroalimentaires.

Tableau II.1. Différences entre ordonnancement classique et ordonnancement en industries agroalimentaires [Gargouri, 03]

<table>
<tr><th>Critères de comparaison</th><th>Ordonnancement classique</th><th>Ordonnancement Agroalimentaire</th></tr>
<tr><td>Matières premières</td><td>- non périssables
- discrétisées
- approvisionnement maîtrisé</td><td>- périssables
- non discrétisées
- approvisionnement irrégulier</td></tr>
<tr><td>Produits semi-finis</td><td>- non périssables
- discrétisés</td><td>- durée de vie courte
- non discrétisés</td></tr>
<tr><td>Produits finis</td><td>- non périssables
- prix de vente fixe et invariable</td><td>- périssables
- prix de vente variable
- coût de stockage élevé</td></tr>
<tr><td>Durées opératoires</td><td>- fixes
- temps d'attente connus</td><td>- variables
- temps d'attente variable</td></tr>
<tr><td rowspan="2">Critères</td><td></td><td>- minimisation des produits périmés
- minimisation du discount de distribution</td></tr>
<tr><td colspan="2">- minimisation du makespan
- minimisation du retard
- respect des dates d'échéances clients
- équilibre des charges de ressources
- minimisation des changements d'outils</td></tr>
<tr><td>Contraintes</td><td colspan="2">- contraintes de disponibilité des matières premières
- contraintes de disponibilité du personnel
- contraintes de gamme (de précédence)
- contraintes disjonctives et cumulatives</td></tr>
<tr><td>Demande</td><td>- demande prévisible</td><td>- demande variable</td></tr>
</table>

II.2.3. Sur l'optimisation dans le cas d'ordonnancements en industries agroalimentaires

Les problèmes d'ordonnancement dans le secteur industriel, sont parmi les problèmes d'optimisation les plus étudiés. Améliorer le rendement des ressources et minimiser les coûts de production sont devenus les objectifs principaux des industriels. C'est dans ce contexte qu'entrent nos travaux de recherche. Ils concernent la résolution de problèmes multi-objectifs d'ordonnancement en industries agroalimentaires. Au niveau de ce type d'industries, assurer la production en quantité et surtout de qualité irréprochable, dans les délais impartis et tenant compte des différentes saisons tout en minimisant les coûts, représente un challenge de tous les jours.

Sachant que les problèmes de production dans les industries agroalimentaires sont complexes et nécessitent la prise en compte de plusieurs facteurs essentiellement liés à la nature des produits manipulés et fabriqués, nous nous orientons pour leur résolution vers le choix des méthodes approchées.

Les algorithmes génétiques sont considérés comme une méthode approchée, pour la résolution de problèmes d'ordonnancement. Gargouri et Tangour se sont intéressés à la résolution des problèmes de production dans les industries agroalimentaires utilisant cette méthode [Gargouri, 03], [Tangour, 07].

II.3. Les algorithmes génétiques

II.3.1. Introduction

L'idée consistant à appliquer les principes de l'évolution biologique des espèces aux systèmes artificiels, introduite à la fin des années 50 [Fraser, 57], a donné naissance aux algorithmes évolutionnaires. Proposée par Holland en 1975 [Holland 75], ils constituent l'une des approches évolutionnaires qui a connu le plus d'essor ces dernières décennies [Davis, 91], [Goldberg, 94], [Chang, 07], [Ruiz, 07].

Les algorithmes génétiques, techniques de recherche basées sur la théorie de l'évolution naturelle des espèces énoncée par Darwin, se sont avérés très efficaces dans la résolution des problèmes complexes d'optimisation [Mesghouni, 98], [Harbaoui, 09].

II.3.2. Présentation des algorithmes génétiques

Les algorithmes génétiques manipulent un ensemble de points dans l'espace de recherche, appelé population d'individus. Chaque individu ou chromosome représente une solution possible du problème posé. Il est constitué d'éléments, appelés gènes, dont les valeurs sont appelées allèles. Les algorithmes génétiques font évoluer cette population d'individus par générations successives, en utilisant des opérateurs inspirés de la théorie de l'évolution qui sont la sélection, le croisement et la mutation.

Cinq éléments de base sont nécessaires pour l'utilisation des algorithmes génétiques [Durand, 94] :

- un principe de codage des éléments de la population, qui consiste à associer à chacun des points de l'espace d'état une structure de données, la qualité de ce codage des données conditionnant le succès des algorithmes génétiques; bien que le codage bi-

naire ait été très utilisé à l'origine, les codages réels sont désormais largement exploités, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles,

- un mécanisme de génération de la population initiale qui doit être capable de produire une population d'individus non homogène servant de base pour les générations futures; le choix de la population initiale est important, car il influence la rapidité de la convergence vers l'optimum global; dans le cas où l'on ne dispose que de peu d'informations sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche,
- une fonction à optimiser, appelée fitness ou fonction d'évaluation de l'individu,
- des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état; l'opérateur de croisement recompose les gènes d'individus existant dans la population alors que l'opérateur de mutation garantit l'exploration de l'espace d'état,
- des paramètres de dimensionnement, représentés par la taille de la population, le nombre total de générations, ou le critère d'arrêt, ainsi que les probabilités d'application des opérateurs de croisement et de mutation.

II.3.3. Codage des algorithmes génétiques

Lorsqu'on aborde les algorithmes génétiques, la première tâche consiste à trouver une modélisation adéquate de l'individu qui facilite la description du problème et respecte ses contraintes. Cette modélisation, appelée codage, permet de représenter les solutions sous forme de chromosomes.

Le codage se base sur deux notions importantes :

- *le génotype* : représente l'ensemble des valeurs des gènes d'un chromosome,
- *le phénotype* : représente la solution du problème traduisant les données contenues dans le génotype.

Pour un passage immédiat entre le phénotype et le génotype, le codage est dit direct, sinon il est dit indirect et une procédure de passage est indispensable à définir.

Dans un codage binaire, le chromosome représente simplement une suite de 0 et de 1. D'après Goldberg [Goldberg 89], ce codage permet d'avoir le nombre maximal de solutions possibles pour une exploration exhaustive de l'espace de recherche.

Aussi, le codage binaire :

- est indépendant des opérations de croisement et de mutation.
- peut présenter des difficultés d'exploitation [Wood, 97] [Caux, 95] du fait qu'il ne permet pas une représentation facile et directe des problèmes.

Même si le codage binaire peut résoudre certains problèmes, il s'est avéré plus pratique d'utiliser un codage réel des chromosomes pour des problèmes d'ordonnancement [Harrath 03]. Celui-ci permet, en effet, de présenter d'une manière directe, plus simple et mieux intelligible les solutions des problèmes.

Les figures II.2 et II.3 présentent respectivement un codage binaire et un codage réel.

1	1	0	1	0	0	1	1

Figure II.2. Exemple de codage binaire

A	T	C	G	T	A

Figure II.3. Exemple d'un codage réel

II.3.4. Opérateurs des algorithmes génétiques

II.3.4.1. *Opérateur de sélection*

Pour faire évoluer et améliorer le patrimoine génétique d'une génération, les individus d'une population se reproduisent. La reproduction doit alors se faire entre les individus les mieux adaptés.

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer partiellement les mauvais. Une opération de sélection est donc nécessaire pour pouvoir choisir les chromosomes qui garantissent l'amélioration de la qualité des solutions.

L'opération de sélection se base essentiellement sur l'évaluation des solutions qui consiste à donner une valeur à un chromosome en fonction de sa qualité. Après avoir évalué une population, ses individus passent par un processus permettant la sélection des parents de la population suivante. Parmi les techniques de sélection, on cite les sélections par la roulette, par le tournoi et par le classement.

- La sélection par la roulette (**RWS** : **R**oulette **W**heel **S**election)

Cette méthode exploite la métaphore d'une roulette de casino pour laquelle chaque individu de la population occupe une section de la roue proportionnelle à sa valeur d'évaluation. Ainsi, la position d'arrêt de la bille indique l'individu sélectionné.

La probabilité de sélection d'un individu x_i, notée $p(x_i)$, définie par la relation :

$$p(x_i) = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)} \tag{II.3}$$

N étant la taille de la population et $f(x_i)$ l'évaluation ou fitness de l'individu x_i.

Le tableau II.2 présente un exemple de cinq individus avec leurs valeurs d'évaluation respectives et leurs probabilités de sélection. La figure II.4 illustre la roue associée à cet exemple.

Tableau II.2. Exemple de sélection à la roulette

Individu	Fitness	Probabilité de sélection
x_1	10	0.5
x_2	5	0.25
x_3	2	0.1
x_4	2	0.1
x_5	1	0.05

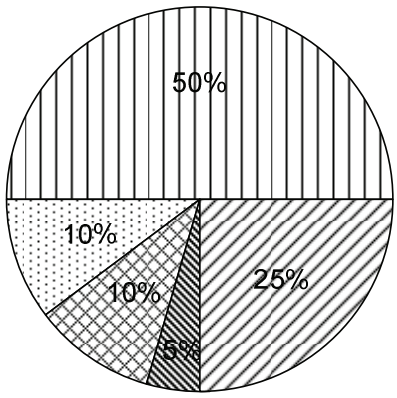

Figure II.4. La roulette de sélection

- La sélection par tournoi

La sélection par tournoi consiste à choisir aléatoirement deux ou plusieurs chromosomes au hasard et permettre à celui qui a une meilleure évaluation, d'être sélectionné. L'avantage de cette sélection par rapport à la sélection par la roulette est d'éviter qu'un individu très fort soit

sélectionné plusieurs fois. Par contre, en utilisant cette méthode, le meilleur individu peut ne pas être sélectionné et ainsi le champ d'exploration est réduit.

Toutefois, la notion d'élitisme peut être introduite. Si l'individu le plus fort n'a pas été sélectionné, il est copié dans la génération suivante à la place d'un autre choisi aléatoirement.

- La sélection par classement

La sélection par classement consiste à trier les individus de la population en fonction de leurs valeurs d'évaluation. Ainsi seuls les individus les plus forts sont conservés. L'inconvénient de cette méthode est la convergence prématurée de l'algorithme. En effet, il est parfois nécessaire de garder des individus jugés plus ou moins bons pour conserver la diversité au niveau de la population. Aussi certains individus faibles, contenant de bonnes configurations, peuvent ne pas être sélectionnés, ainsi le champ d'exploration est réduit.

II.3.4.2. *Opérateur de croisement*

Le croisement permet l'enrichissement de la diversité dans la population en manipulant la structure des chromosomes. En effet, le croisement permet de créer de nouvelles séquences de gènes pour les chromosomes enfants à partir d'une base de configurations des séquences héritées des chromosomes parents. Cette opération se produit selon une probabilité P_c fixée par l'utilisateur selon le problème à optimiser.

Dans la littérature, il existe plusieurs opérateurs de croisement qui dépendent essentiellement du type du codage et de la nature du problème à traiter [Mesghouni, 99].

Pour le codage binaire, le croisement à un point, le croisement multi-points et le croisement uniforme sont distingués. Pour le codage réel, nous pouvons citer à titre d'exemple, le croisement PMX (Partially Mapped Crossover), le croisement MPX (Maximal Preservative Crossover), …

- Croisement à un point

Une position k est choisie aléatoirement dans les chromosomes parents. Les chromosomes enfants sont le fruit d'un échange de deux parties des parents. La figure II.5 illustre un exemple de croisement à un point.

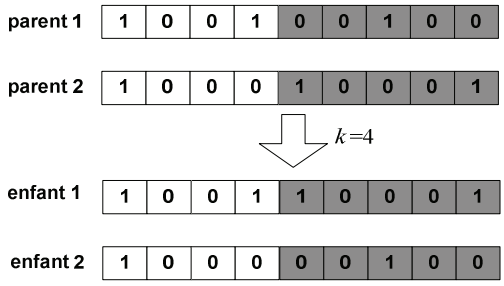

Figure II.5. Exemple du croisement à un point

- Croisement multi-points

Ce croisement est similaire au précédent sauf que plusieurs points de croisement y sont impliqués. La figure II.6 illustre un cas du croisement multi-points qui est le croisement bi-points. Il consiste à choisir deux points de croisement *k1 et k2* et à échanger les segments des deux parents.

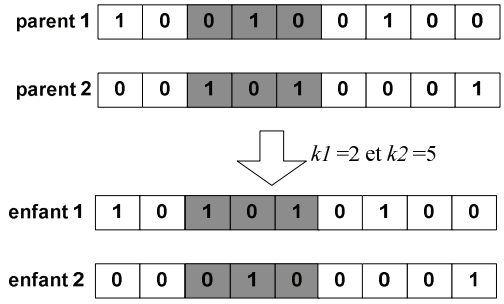

Figure II.6. Exemple du croisement bi-points

- Croisement uniforme

Parmi les approches les plus connues du croisement uniforme, on cite l'utilisation du vecteur masque, le masque étant une suite aléatoire de 0 et 1 de la taille du chromosome. Il sert à déterminer de quel parent sont repris les gènes. La figure II.7 présente un exemple du croisement uniforme utilisant le masque : en présence du 0, l'enfant hérite du parent 1 et en présence de 1, l'enfant hérite du parent 2.

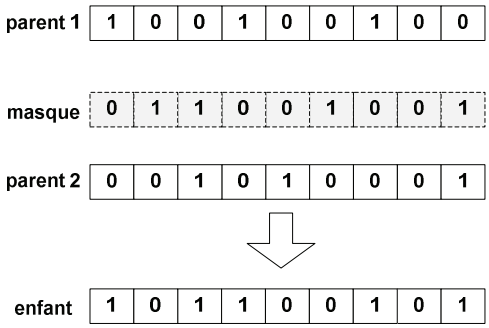

Figure II.7. Exemple d'un croisement uniforme

II.3.4.3. ***Opérateur de mutation***

La mutation est un changement aléatoire occasionnel avec une faible probabilité P_m (fixée par l'utilisateur) de la valeur d'un ou plusieurs gènes d'un chromosome. En général, la mutation permet de garder une diversité dans l'évolution des individus et d'éviter les optimums locaux.

- Mutation uni-point

Cette mutation consiste en la transformation aléatoire d'une seule valeur d'un chromosome. La figure II.8 présente un exemple d'une mutation uni-point.

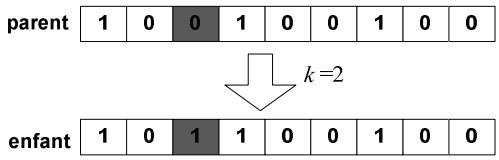

Figure II.8. Exemple d'une mutation uni-point

- Mutation multi-points

Elle consiste en la transformation aléatoire de deux ou plusieurs valeurs d'un chromosome. La figure II.9 donne un exemple d'une mutation bi-points.

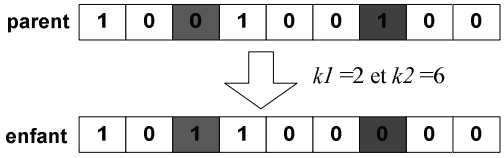

Figure II.9. Exemple d'une mutation bi-points

II.3.5. Paramètres de dimensionnement

Les paramètres de dimensionnement conditionnent la convergence des algorithmes génétiques. Leurs valeurs dépendent de la problématique étudiée comme le nombre de variables où la taille du problème. Ces paramètres sont au nombre de quatre.

- ✓ La taille de la population : le choix d'une petite population conduit à une convergence rapide et un optimum local. Par contre, une grande population engendre un temps de calcul excessif.
- ✓ La probabilité de croisement : plus la probabilité de croisement est grande plus la population subit de changements.

- ✓ La probabilité de mutation : une mutation avec une grande probabilité risque de perturber la convergence et de conduire à une solution sous-optimale. Par contre, un faible taux de mutation assure une bonne exploration de l'espace de recherche.
- ✓ Le nombre de générations qui peut être retenu a priori comme un critère d'arrêt.

II.4. Nouvelles approches proposées pour la résolution de problèmes d'ordonnancement en industries agroalimentaires basées sur les algorithmes génétiques

II.4.1. Formulation du problème d'ordonnancement relatif à un Benchmark relatif aux industries agroalimentaires

En industries agroalimentaires, les produits manipulés sont en général de durées de vie et de dates de mise en fabrication assez courtes. Une opération est, dans ce cas, caractérisée aussi bien par sa date de début au plus tôt et sa date de fin au plus tard mais aussi par les dates de validités v_{ijk} des composants c_{ijk} nécessaires pour accomplir cette opération [Karray, 10].

Notations

O_{ij} : $j^{ème}$ opération du produit *i*

t_{ij} : date effective de début de fabrication de l'opération O_{ij}

r_{ij} : date de début au plus tôt de l'opération O_{ij}

γ_{ij} : date de fin effective de l'opération O_{ij}

d_{ij} : date échue de l'opération O_{ij}

p_{ij} : durée opératoire de l'opération O_{ij}

P_i : produit fini de l'opération O_{ij}

c_{ijk} : $k^{ème}$ composant de l'ensemble des composants de l'opération O_{ij}

υ_{ijk} : date de validité limite du composant c_{ijk}

C_{P_i} : date de fin de fabrication du produit P_i

P_{ijk}^{rev} : prix de revient du composant c_{ijk}

$C_{\max}$: date de fin de l'ordonnancement

$d_{P_i}^{liv}$: date de livraison du produit P_i

Dv_{P_i} : durée de vie du produit P_i

Dr_{P_i} : durée de retour du produit P_i

$P_{P_i}^{ven}$: prix de vente unitaire du produit P_i

$C_{P_i}^{stk}$: coût du stockage, par unité de temps, d'une unité du produit P_i

II.4.1.1. ***Les contraintes***

L'objectif est de construire un ordonnancement qui satisfait les critères relatifs aux industries agroalimentaires tout en respectant les contraintes du problème :
- la contrainte de gamme qui représente l'ordre de passage de l'opération dans la gamme ;
- la contrainte de disponibilité de l'opération à ordonnancer selon sa date de début au plus tôt.

II.4.1.2. ***Choix des critères***

Dans ce problème, les critères considérés sont relatifs aux industries agroalimentaires, il s'agit de la périssabilité des produits et le discount de distribution et un troisième critère, plus classique, à savoir la date de fin de l'ordonnancement.

Ils s'expriment respectivement par les trois fonctions objectifs suivantes : *C1*, *C2* et *C3* :

- la durée maximale d'exécution de la dernière opération, notée $C_{\max}$,

$$C1 = C_{\max} = \max_{1 \le i \le n}(C_{P_i}) \tag{II.4}$$

- le coût des produits périmés, notée C_2,

$$C2 = \sum_i \sum_j \sum_k P_{ijk}^{rev} \left(\frac{\max(0, t_{ij} - v_{ijk})}{\left(t_{ij} - v_{ijk}\right)} \right) \tag{II.5}$$

- le coût du discount de distribution, notée C_3,

$$C3 = \sum_i \max\left(0, d_{P_i}^{liv} - C_{P_i}\right) \times \left(\frac{P_{P_i}^{ven}}{Dv_{P_i} - Dr_{P_i}} + C_{P_i}^{stk} \right) \tag{II.6}$$

L'objectif est alors de sélectionner parmi l'ensemble des ordonnancements réalisables celui qui présente le meilleur compromis entre les différents critères en réduisant et en filtrant l'espace de recherche initial.

II.4.2. Méthodes de résolution existantes

Le problème multi-objectifs ou multicritère, introduit dans le chapitre 1, est un problème d'optimisation vectoriel, dans lequel plusieurs composantes d'un vecteur fonction coût sont à optimiser. Parmi les méthodes de résolution, on cite l'approche Pareto et la transformation du problème multi-objectifs en un problème mono-objectif.

II.4.2.1. *Approche Pareto-Optimalité (APO)*

Un problème multicritère *P*, constitué de n variables, *m* contraintes d'inégalités, *p* contraintes d'égalités et *k* critères, peut être formulé de la manière suivante :

$$(P)\begin{cases} \min f(x) = [f_1(x), f_2(x), \ldots, f_k(x)] \\ g_i(x) \le 0 \quad i = 1 \cdots m \\ g_j(x) = 0 \quad i = 1 \cdots p \end{cases}$$

x étant un vecteur solution d'un univers *E* de dimension n, appelé espace de décision (ou espace des paramètres). L'espace défini par f noté *U*, est appelé espace des critères (ou espace des objectifs). La fonction d'évaluation $f : E \to U$, d'un problème multicritère, fait donc correspondre un vecteur *y* de dimension *k* tel que $y_k = f_k(x)$ à tout vecteur *x* de dimension *n* de l'espace *E*.

Il s'agit de trouver des solutions représentant un compromis possible entre les critères. Ce compromis est défini comme étant le concept de Pareto-optimalité introduit par l'économiste V. Pareto au XIX^{ème} siècle [Pareto, 64].

La formulation du concept est la suivante : dans un problème multicritère, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres. Cet équilibre a été appelé optimum de Pareto. Un point est dit Pareto optimal s'il n'est dominé par aucun autre point appartenant à l'espace des solutions. Ces points sont également appelés solutions non inférieures ou non dominées.

Un point $x \in E$ domine $y \in E$, si est seulement si :

$$\begin{cases} \forall i \in E, f_i(x) \le f_i(y) \\ et\ j\ tel\ que\ f_i(x) < f_j(y) \end{cases}$$

La figure II.10 illustre un exemple où l'on cherche à minimiser f_1 et f_2. Les points 1, 3 et 5 ne sont pas dominés. Par contre, le point 2 est dominé par le point 3, et le point 4 dominé par le point 5.

L'avantage de la méthode d'agrégation est la production d'une seule solution, ne nécessitant donc pas d'interaction avec le décideur.

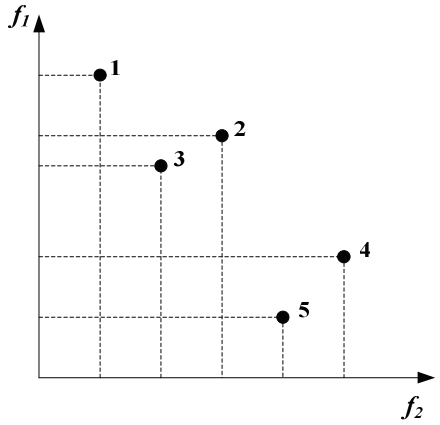

Figure II.10. Exemple de dominance

II.4.2.2. ***Méthode d'agrégation***

Dans la résolution d'un problème multi-objectifs, il existe plusieurs méthodes traditionnelles qui transforment ce problème en un problème mono-objectif.

Parmi ces méthodes, figure la méthode d'agrégation. Celle-ci consiste à transformer le problème *(P)* en un problème (P_λ) qui revient à combiner les différentes fonctions coûts f_i du problème, $i=1,2...$, en une seule fonction objectif F, généralement de façon linéaire [Hwang, 79] :

$$F(x) = \sum_{i=1}^{n} \lambda_i f_i(x) \tag{II.7}$$

λ_i étant un coefficient de pondération et/ou de mise à l'échelle. Le principe de fonctionnement de la méthode d'agrégation est de déterminer un vecteur poids. Il s'agit de trouver un hyperplan dans l'espace objectif avec une orientation fixée. Dans le cas du problème bi-critères, donné dans la figure II.11, l'hyperplan est une droite. La solution Pareto optimale est le point où l'hyper-plan possède une tangente commune avec l'espace réalisable.

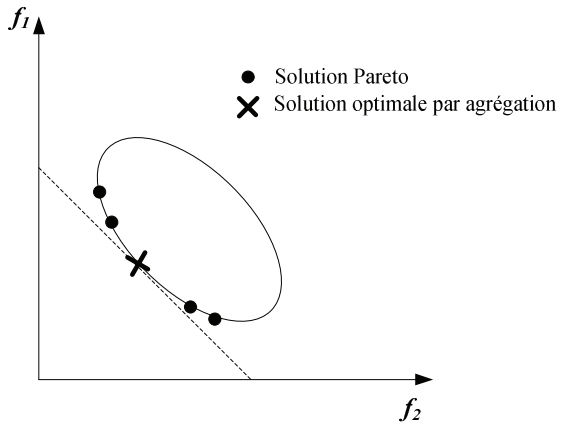

Figure II.11. Exemple d'optimisation par agrégation à deux objectifs

II.4.3. Méthodes de résolution proposées

Après avoir défini deux approches existantes pour la résolution des problèmes multi-objectifs, nous allons, par la suite, présenter trois approches basées sur les algorithmes génétiques pour la résolution des problèmes multi-objectifs.

II.4.3.1. *Première approche proposée: Algorithmes Génétiques Séquentiels (SGA_1)*

Notre première approche qui est l'algorithme génétique séquentiel SGA_1, repose sur l'idée d'utiliser la stratégie séquentielle [Karray, 11 (a)]. En effet, au lieu d'utiliser un seul algorithme multi-objectifs pour résoudre plusieurs critères, plusieurs algorithmes génétiques sont utilisés pour résoudre respectivement les différents critères. Le résultat trouvé suite à la mise en œuvre du premier algorithme, est injecté dans le deuxième et ainsi de suite jusqu'à obtention d'un résultat qui satisfait tous les critères.

Considérant que le problème d'optimisation concerne les trois critères *C1*, *C2* et *C3*, trois algorithmes génétiques sont donc considérés pour résoudre ce problème multi-objectifs.

Le codage choisi et les opérateurs génétiques pris en considération dans notre approche sont définis dans le paragraphe II.4.4.

La résolution par stratégie séquentielle, pour notre cas d'étude, se fait en trois étapes [Karray, 11 (a)] :

- *Etape 1*: Les algorithmes génétiques sont appliqués pour la résolution du critère *C1*. Dans cette étape, la population initiale est générée aléatoirement et les opérateurs de sélection, de croisement et de mutation sont mis au point et constituent ainsi le processus de reproduction pour créer de nouvelles populations.

Ce processus est répété jusqu'à obtention d'une population PF1 qui s'approche au mieux de la solution optimale.

- *Etape 2* : La population PF1, obtenue précédemment, est choisie comme population initiale de l'algorithme pour résoudre le problème relatif au critère *C2*. Sur les nouvelles populations, générées de façon itérative, on applique les opérateurs de croisement, de mutation et de sélection. Au final, une population PF2 proche de la solution optimale est obtenue.
- *Etape 3* : Cette étape est semblable à la précédente. En effet, la population PF2, obtenue précédemment, est choisie comme population initiale de l'algorithme pour résoudre le problème relatif au critère *C3*. Cette population servira de base pour les générations ultérieures, obtenues à partir du processus de reproduction.

  Ce processus est répété jusqu'à obtention d'une population PF3 qui s'approche au mieux de la solution optimale.

Nous obtenons ainsi une population satisfaisant les trois critères *C1*, *C2* et *C3*.

Cette méthode de résolution, que nous avons qualifiée de séquentielle ou de série, est schématisée dans la figure II.12.

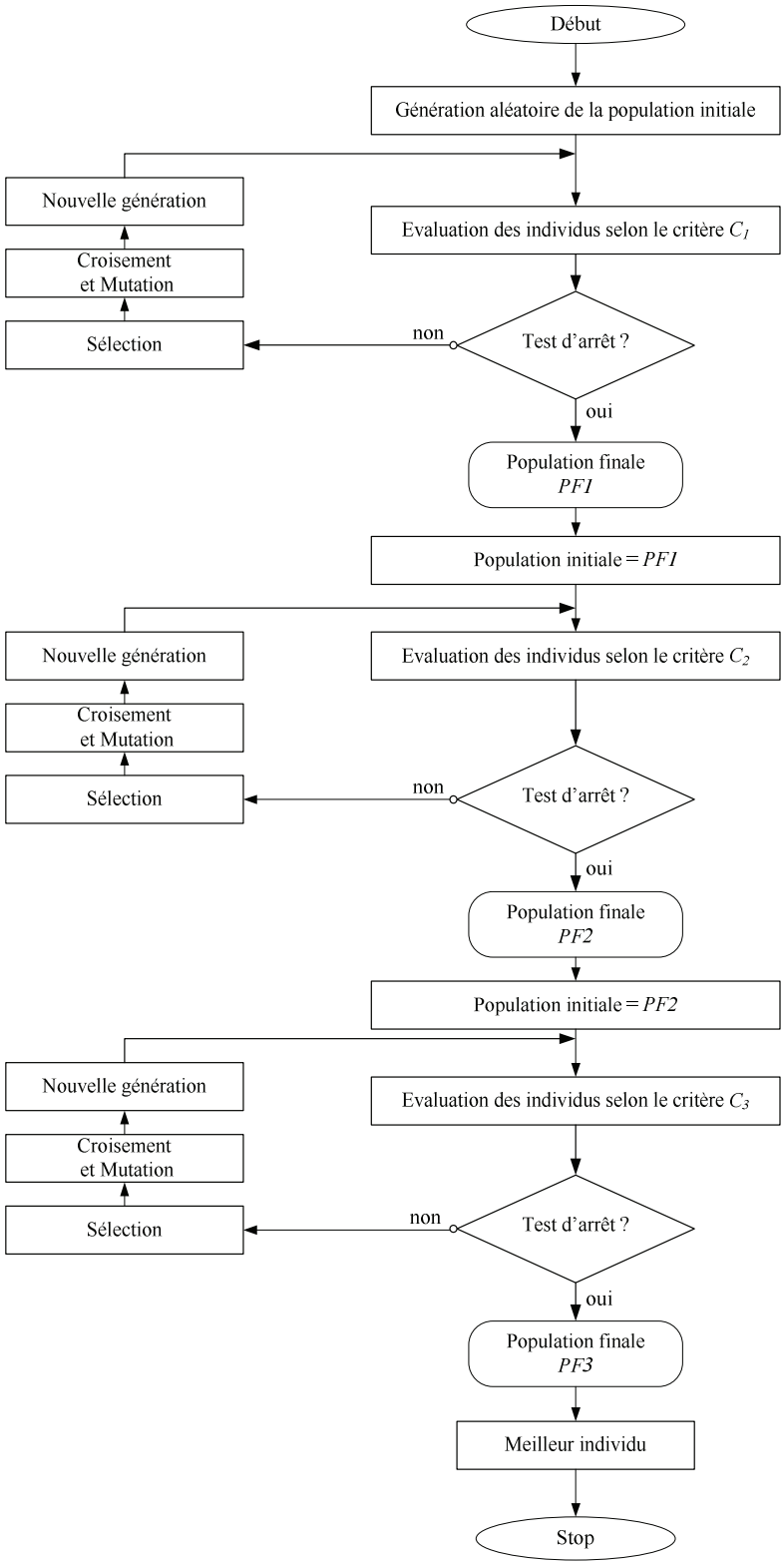

Figure II.12. Organigramme de l'algorithme SGA_1 proposé

II.4.3.2. *Deuxième approche proposée: Algorithmes Génétiques Parallèles (PGA_1)*

La deuxième approche qui est l'algorithme génétique parallèle PGA_1, se base sur une approche parallèle [Karray, 11(b)]]. Plusieurs travaux utilisant les algorithmes génétiques parallèles ont été développés dans la littérature [Cantú-Paz, 98] [Cantú-Paz, 00a] [Chipperfield, 96] [Dorigo, 93] [He, 07]. Ils ont montré leur efficacité et leur facilité d'implémentation [Alba, 02a] [Alba, 02b] [Cantú-Paz, 00b] [Goldberg, 00] [Gong, 05].

Il existe principalement deux modèles d'algorithmes génétiques parallèles: le modèle maître-esclave et le modèle en îlot. Dans le premier modèle, un processeur maître est dédié aux étapes de sélection, de croisement, de mutation et de remplacement et les processeurs esclaves s'occupent de calculer la fitness des individus. Dans le deuxième modèle, la population est divisée en plusieurs sous-populations réparties sur des processeurs. Chaque processeur utilise les algorithmes génétiques conventionnels.

Dans ce type de modèle, les sous-populations peuvent communiquer entre elles sous forme de migration d'individus. En effet, des individus sélectionnés parmi une sous-population peuvent être déplacés vers une autre. Le choix des individus candidats à la migration peut être paramétrable mais, très souvent, il s'agit de faire écouler une période de temps pendant laquelle aucune migration n'est permise pour ensuite choisir un individu (le meilleur, le pire ou l'aléatoire) et le substituer avec un autre (le meilleur, le pire ou l'aléatoire) d'une population voisine. La migration est régie par deux paramètres: l'intervalle de migration et le taux de migration.

L'approche PGA_1 développée se base sur le deuxième modèle mais la communication entre sous-populations n'est permise qu'à la dernière étape pour faire un croisement final des résultats trouvés dans chaque algorithme. La deuxième caractéristique de notre approche est que le nombre de sous-populations dépend du nombre de critères à résoudre. L'approche proposée repose sur l'idée que chaque processeur utilise les algorithmes génétiques pour résoudre un problème mono-objectif.

Sachant que trois critères *C1*, *C2* et *C3* sont pris en compte, trois sous-populations sont donc considérées.

A l'instar de l'approche SGA_1, le codage choisi et les opérateurs génétiques, pris en considération dans notre approche PGA_1, sont ainsi définis dans le paragraphe II.4.4.

Deux étapes sont nécessaires pour la mise en œuvre de l'approche proposée [Karray, 11(b)]].

- *Etape 1*: Elle est relative à la génération d'une population initiale qui est divisée en trois sous-populations réparties sur trois processeurs. Ainsi, trois algorithmes génétiques sont appliqués séparément et en parallèle pour optimiser respectivement le critère *C1*, le critère *C2* et le critère *C3*.

  Chaque algorithme va résoudre un problème d'optimisation mono-objectif. Le processus de reproduction de chaque algorithme est mis au point pour créer de nouvelles populations.

  Ce processus est répété jusqu'à l'obtention de trois populations distinctes PF1, PF2 et PF3. Chaque population s'approche au mieux de la solution optimale.
- *Etape 2* : Elle porte sur le choix aléatoire, pour chaque population finale trouvée précédemment et un certain nombre d'individus. Cette sélection se fait en utilisant l'opérateur de sélection par la roulette. L'opérateur de croisement bi-points est, ensuite, appliqué pour faire un croisement final de ces trois populations.

Finalement, une population satisfaisant les trois critères *C1*, *C2* et *C3* est obtenue.

Cette méthode de résolution proposée, que nous qualifions de parallèle, est schématisée dans la figure II.13.

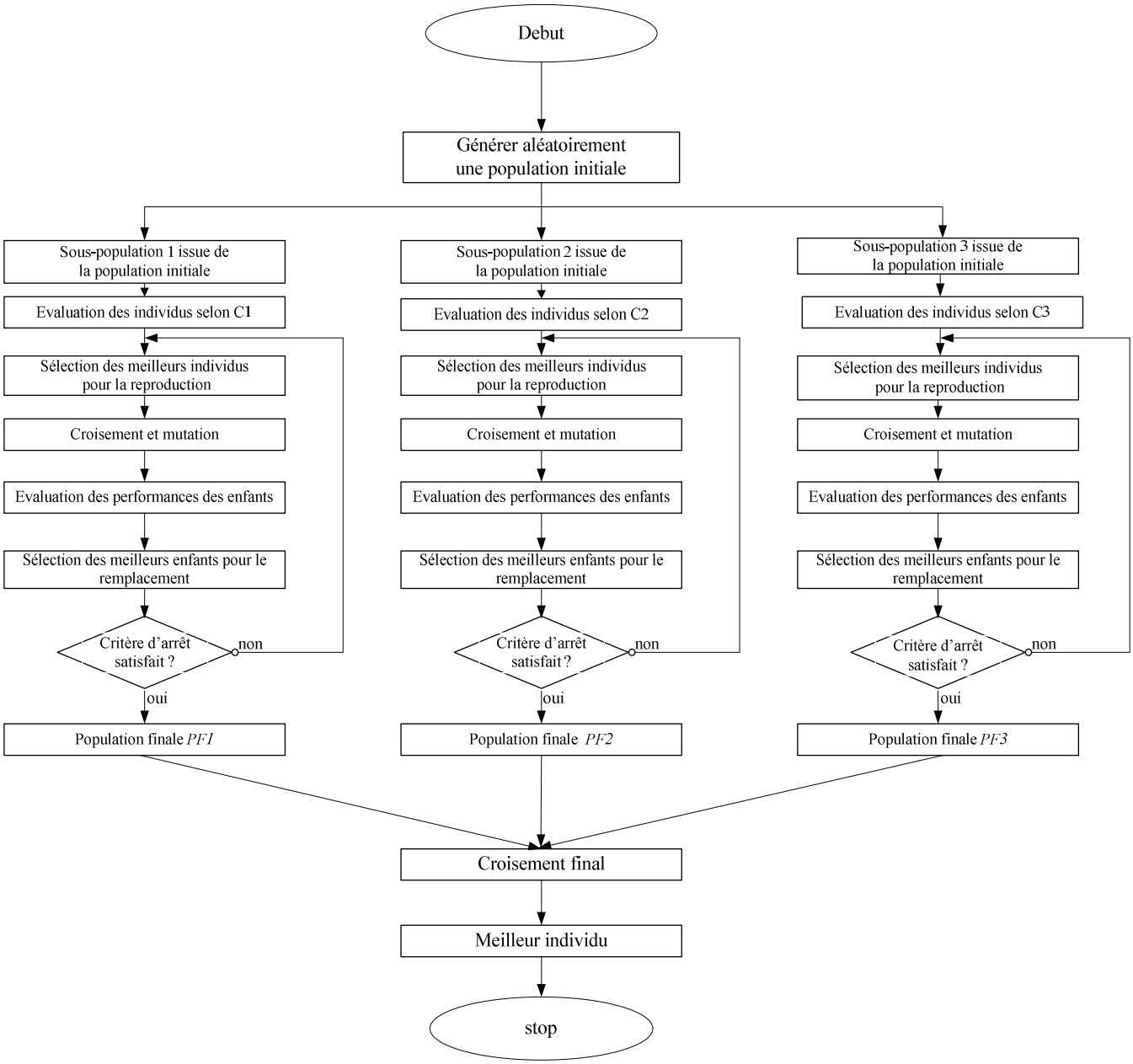

Figure II.13. Organigramme de l'algorithme PGA_1 proposé

II.4.3.3. *Troisième approche proposée relatifve aux algorithmes génétiques parallèles séquentiels (PSGA_1)*

La troisième approche développée est une approche hybride qui combine deux approches citées précédemment, à savoir SGA_1 et PGA_1.

L'idée de base de cette approche est d'utiliser d'abord l'approche PGA_1 pour résoudre deux critères. Le résultat ainsi trouvé est injecté comme population initiale des algorithmes génétiques pour résoudre un troisième critère.

L'approche PSGA_1 utilise le même codage et les mêmes opérateurs génétiques pris en considération par l'approche SGA_1 et PGA_1.

Cette approche de résolution s'effectue en trois étapes :

- *Etape 1*: Elle porte sur la génération d'une population initiale qui est divisée en deux sous-populations réparties sur deux processeurs. Deux algorithmes génétiques sont appliqués, séparément et en parallèle, pour résoudre respectivement le critère *C1* et le critère *C2*. Le processus de reproduction de chaque algorithme est mis au point pour créer de nouvelles populations.

  Ce processus est répété jusqu'à l'obtention de deux population distinctes PF1 et PF2, chacune s'approchant au mieux de la solution optimale.
- *Etape 2*: Elle concerne le choix, pour chaque population finale trouvée précédemment, d'un certain nombre d'individus. Cette sélection se fait en utilisant l'opérateur de sélection par la roulette. Ensuite, est appliqué l'opérateur de croisement bi-points pour la recombinaison de ces deux populations. Au final, une population PF s'approchant au mieux de la solution optimale, est obtenue.
- *Etape 3* : La population PF, obtenue précédemment, est injectée dans notre algorithme comme population initiale pour résoudre le problème relatif au critère *C3*. A partir de cette population, les nouvelles populations générées de façon itérative, on applique les opérateurs de croisement, de mutation et de sélection. Au final, une population satisfaisant les trois critères *C1*, *C2* et *C3* est obtenue.

Cette méthode de résolution PSGA_1 est schématisée dans la figure II.14.

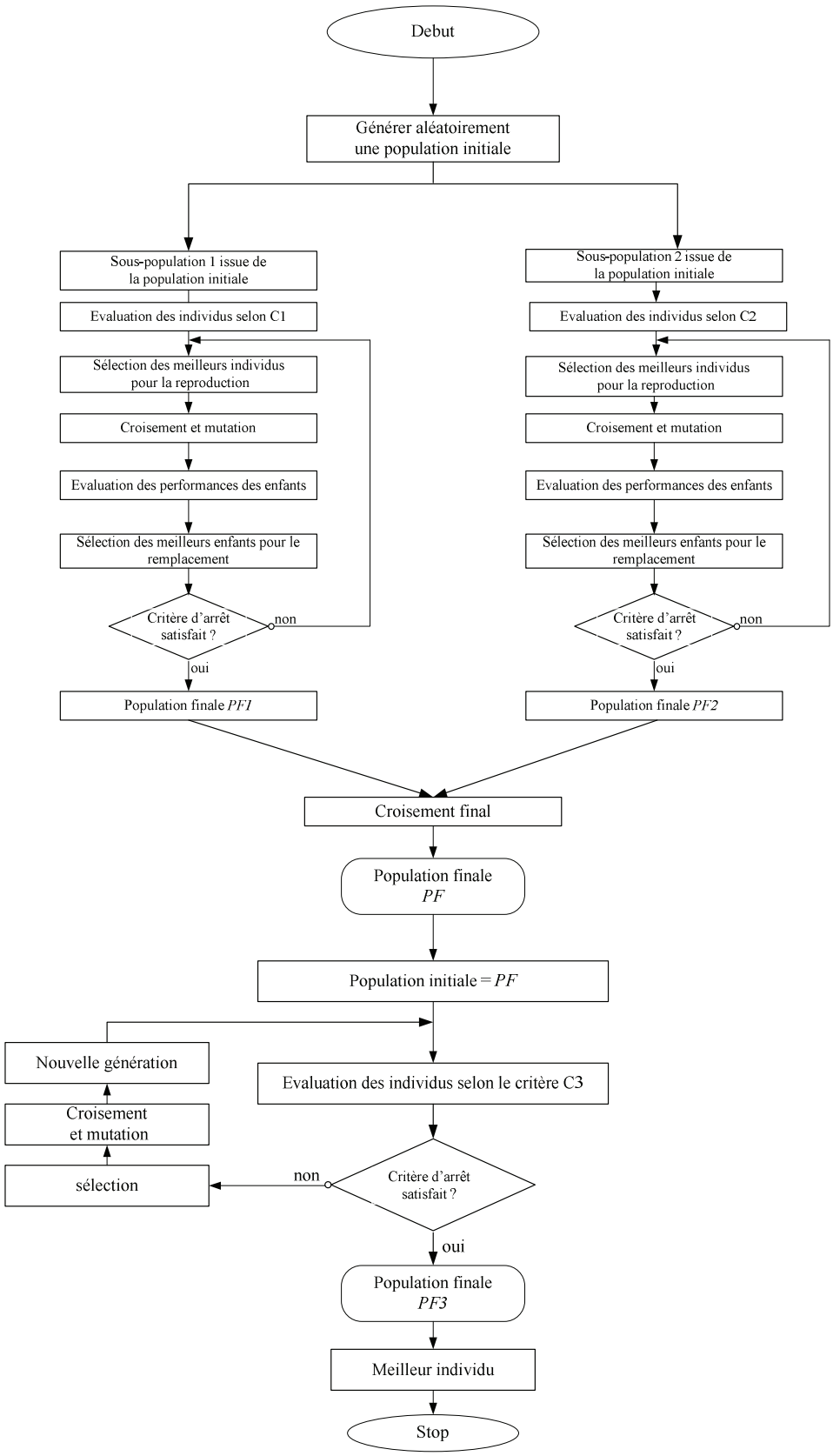

Figure II.14. Organigramme de l'algorithme PSGA_1 proposé

II.4.4. Spécificités des algorithmes génétiques proposés

Dans un algorithme génétique, la population évolue grâce à trois phases principales : la sélection favorisant les « bons » individus par rapport aux « mauvais », la mutation altérant l'individu sur lequel elle s'applique, et le croisement générant, pour chaque couple parents, un couple d'enfants qui héritent des caractéristiques de chacun de leurs parents.

Lorsqu'on aborde les algorithmes génétiques, la première tâche consiste à trouver une représentation adéquate de l'individu.

II.4.4.1. *Codage proposé*

La différence majeure entre les divers algorithmes génétiques appliqués à l'ordonnancement réside dans la représentation génétique des chromosomes. Deux types de représentations génétiques sont distingués : le codage direct et le codage indirect.

Dans un codage direct, le chromosome représente une solution complète du problème traité. Ainsi, chaque chromosome contient toutes les informations utiles à la création de l'ordonnancement comme les dates de début des tâches et les contraintes de précédence. Dans ce cas, les opérateurs de croisement et de mutation sont adaptés pour ce type de codage afin de générer des solutions admissibles. Le codage basé sur les tâches et celui basé sur les opérations sont considérés comme des codages directs.

Dans le codage indirect, les chromosomes ne représentent plus directement une solution complète du problème mais plutôt des listes de priorité ou des heuristiques.

Dans ce cas, un ordonnanceur est utilisé pour transformer tout chromosome en solution effective. Parmi les types de codages indirects, on cite le codage basé sur les règles de priorité et le codage basé sur les listes de préférence.

Le codage pris en compte dans le problème d'ordonnancement en industries agroalimentaires, CLOA, est un codage direct basé sur les opérations.

Le codage CLOA proposé définit l'ordre, la date de début au plus tôt, la durée et la date de fin effective des opérations. Ce codage est inspiré du codage CLO (Codage Liste des Opérations) [Kacem, 03] et du codage CPM (Codage Parallèle Machines) [Mesghouni, 99]. Il consiste à proposer des listes ordonnées d'une gamme de produits en ligne de fabrication.

La figure II.15 présente le codage proposé.

$O_{i1}, r_{i1}, p_{i1}, \gamma_{i1}$	$O_{i4}, r_{i4}, p_{i4}, \gamma_{i4}$	$O_{i3}, r_{i3}, p_{i3}, \gamma_{i3}$	• • •	$O_{ij}, r_{ij}, p_{ij}, \gamma_{ij}$

Figure II.15. Codage proposé

La date de fin effective de l'opération O_{ij} est obtenue par la relation suivante :

$$\gamma_{ij} = \max(r_{ij}, \gamma_{ik}) + p_{ij} \quad \text{(II.8)}$$

γ_{ik} représentant la date de fin de l'opération O_{ik} qui précède l'opération O_{ij}.

II.4.4.2. ***Génération de la population initiale***

Après avoir choisi le codage, la deuxième étape consiste à déterminer la population initiale formée de solutions admissibles du problème. Il existe dans la littérature, plusieurs mécanismes de générations de la population initiale [Caux, 95]. Une méthode classique consiste à générer aléatoirement les chromosomes constituant la population initiale. Cette méthode permet d'avoir une population variée et d'explorer ainsi diverses zones dans l'espace de recherche. Aussi, pour générer la population initiale, des heuristiques peuvent être utilisées. Cette méthode peut, cependant, faire converger l'algorithme vers des optimas locaux [Harrath, 03].

Pour avoir un ensemble d'individus assez diversifié, la génération de la population initiale, dans notre application, se fait en générant des individus aléatoirement. La figure II.16 présente l'algorithme de génération de la population initiale.

Algorithme : Génération Aléatoire de la population initiale

```
Variable locale n : taille de la population
Début
    Pour i=1 jusqu'à n
        Faire Générer un individu valide aléatoirement
    Fin Pour
Fin
```

Figure II.16. Algorithme de génération de la population initiale

II.4.4.3. ***Opérateurs génétiques proposés***

Les opérateurs génétiques permettent la diversification la population au cours des générations et l'exploration de l'espace d'état, représenté par l'espace des solutions. Ainsi, l'opérateur de croisement recompose les gènes d'individus existants dans la population, quant à l'opérateur de mutation, il a pour but de garantir l'exploration de l'espace d'état.

Ces différents opérateurs proposés pour la résolution du problème traité, sont présentés dans ce qui suit.

II.4.4.4.***Opérateur de sélection***

La méthode de sélection élaborée est basée sur le principe de la roulette, exposé au paragraphe précédent. Dans ce cas, les individus non dominés sont gardés afin de permettre l'obtention de différentes solutions optimales possibles constituant un support de décision et garantissant aussi la diversité de l'évolution des individus.

II.4.4.5.***Opérateur de croisement***

Les algorithmes génétiques utilisés pour résoudre les problèmes d'ordonnancement se basent sur le codage réel. Ceci nécessite des opérateurs de croisement spécifiques pour la production de chromosomes valides. Ainsi, selon le type de codage choisi, l'un des opérateurs de croisement, définis dans le paragraphe précédent, peut être adapté à notre problème. Pour notre cas d'étude, nous avons choisi d'adapter le croisement bi-points. La figure II.17 et la figure II.18 illustrent respectivement l'algorithme de l'opérateur de croisement et un exemple de cet algorithme.

Algorithme : Opération de croisement bi-points

Variable locale n : taille du chromosome
I_1 et I_2 : individus parent s
I_3 et I_4 : individus enfants
k_1 et k_2 : points de croisement
p_c : probabilité de croisement
p_1 et p_2 : probabilités de croisement des individus I_1 et I_2

Début
  Choisir aléatoirement deux individus parents I_1 et I_2 et deux points de croisement k_1 et k_2
  Si $p_1 \geq p_c$ **et** $p_2 \geq p_c$ **alors** parcourir toutes les opérations
    Tant que $j \leq n$ **faire**
      Copier les opérations de I_1, précédant le point de croisement k_1 et succédant le point de croisement k_1 dans I_3
      Copier les opérations de I_2, précédant le point de croisement k_1 et succédant le point de croisement k_1 dans I_4
      Copier, à la même position, les opérations non existantes de I_2 dans I_3
      Copier, à la même position, les opérations non existantes de I_2 dans I_4
    Fin Tant que
  Fin Si
  Terminer la construction de I_3 et respectivement I_4 avec les opérations manquantes
  Mettre à jour I_3 et I_4
  Calculer les fonctions objectif C_1, C_2 et C_3 des deux nouveaux individus
Fin

Figure II.17. Algorithme de croisement bi-points

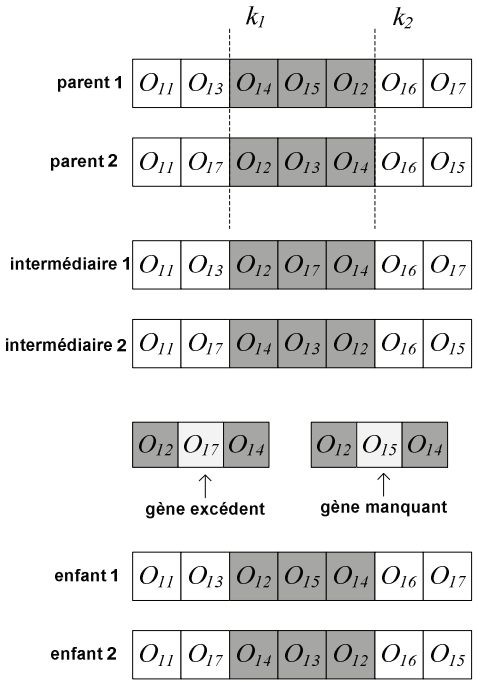

Figure II.18. Exemple d'un croisement bi-points

II.4.4.6. ***Opérateur de mutation***

A l'instar de l'opérateur de croisement, l'opérateur de mutation, défini dans le paragraphe précédent, est adapté à notre problème. Ainsi, l'opérateur de mutation choisi est la mutation bi-points. La figure II.19 et la figure II.20 présentent respectivement l'algorithme de l'opérateur de mutation et un exemple de cet algorithme.

Algorithme : Opération de mutation bi-points

Variable locale n : taille du chromosome
I_1: individu
I_2 : individu muté
k_1 et k_2 : points de mutation
p_m : probabilité de mutation
p : probabilité de mutation de l'individu

Début
  Choisir aléatoirement un individu I_1 et deux points de mutation k_1 et k_2 correspondant respectivement aux opérations O_i et O_j
  Si $p \geq p_m$ **alors**
    Permuter les opérations O_i et O_j
  Fin Si
  Mettre à jour I_2
  Calculer les fonctions objectif C_1, C_2 et C_3 du nouvel individu
Fin

Figure II.19. Algorithme de mutation bi-points

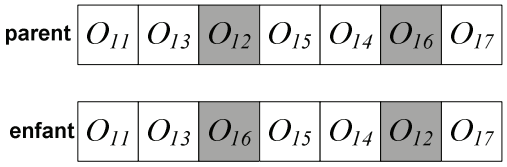

Figure II.20. Exemple d'une mutation bi-points

II.5. Résultats obtenus par simulation

II.5.1. Présentations des cas d'ateliers étudiés

Deux exemples, traitant de problèmes en industries agroalimentaires sont présentés dans cette partie. Ces problèmes consistent à ordonnancer respectivement 10 et 30 produits dans des ateliers à une machine pour montrer le comportement des approches développées à savoir SGA_1, PGA_1 et PSGA_1, face à deux problèmes de tailles et de complexités différentes.

Les données de ces problèmes d'ordonnancement en industries agroalimentaires sont consignées dans les tableaux II.3 et II.4. Elles sont relatives à la date de début au plus tôt des opérations (r_{ij}), la durée opératoire des opérations (p_{ij}), la date de validité limite des composants (υ_{ijk}), le prix de revient des composants (P_{ijk}^{rev}), la date de livraison du produit fini ($d_{P_i}^{liv}$), le durée de vie du produit fini (Dv_{P_i}), la durée de retour du produit fini (Dr_{P_i}), le prix de vente unitaire du produit ($P_{P_i}^{ven}$) et le coût du stockage, par unité de temps, d'une unité du produit fini ($P_{P_i}^{stk}$).

Tableau II.3. Données relatives au benchmark 1

	r_{1j}	p_{1j}	v_{1j1}	v_{1j2}	v_{1j3}	P_{1j1}^{rev}	P_{1j2}^{rev}	P_{1j3}^{rev}	$d_{P_1}^{liv}$	Dv_{P_i}	Dr_{P_i}	$P_{P_i}^{ven}$	$C_{P_i}^{stk}$
O_{11}	0	1	13	15	-	2	1	-	14	35	10	6	3
O_{12}	1	2	14	14	12	3	2	4	8	35	10	6	3
O_{13}	2	4	13	5	13	4	2	3	8	35	10	6	3
O_{14}	3	2	13	14	12	3	4	2	12	33	9	8	4
O_{15}	4	1	12	13	11	2	3	2	20	33	9	8	4
O_{16}	1	2	7	15	14	1	2	1	12	33	9	8	4
O_{17}	3	1	7	15	14	1	2	3	12	33	9	8	4
O_{18}	2	3	13	16	12	2	1	2	14	31	11	5	2
O_{19}	1	2	9	15	14	2	4	3	8	31	11	5	2
O_{110}	3	4	9	15	14	2	1	3	18	31	11	5	2

Tableau II.4. Données relatives au benchmark 2

	r_{1j}	p_{1j}	v_{1j1}	v_{1j2}	v_{1j3}	P^{rev}_{1j1}	P^{rev}_{1j2}	P^{rev}_{1j3}	$d^{liv}_{P_1}$	$^{Dv}P_i$	$^{Dr}P_i$	$P^{ven}_{P_i}$	$C^{stk}_{P_i}$
O_{11}	0	1	13	15	-	2	1	-	24	65	40	6	3
O_{12}	1	2	14	14	12	3	2	4	18	65	40	6	3
O_{13}	2	4	13	5	13	4	2	3	18	65	40	6	3
O_{14}	3	2	13	14	12	3	4	2	22	63	39	8	4
O_{15}	4	1	12	13	11	2	3	2	30	63	39	8	4
O_{16}	1	2	7	15	14	1	2	1	22	63	39	8	4
O_{17}	3	1	7	15	14	1	2	3	22	63	39	8	4
O_{18}	2	3	13	16	12	2	1	2	24	61	41	5	2
O_{19}	1	2	9	15	14	2	4	3	18	61	41	5	2
O_{110}	3	4	9	15	14	2	1	3	28	61	41	5	2
O_{111}	0	1	13	15	-	2	1	-	24	65	40	6	3
O_{112}	1	2	14	14	12	3	2	4	18	65	40	6	3
O_{113}	2	4	13	5	13	4	2	3	18	65	40	6	3
O_{114}	3	2	13	14	12	3	4	2	22	63	39	8	4
O_{115}	4	1	12	13	11	2	3	2	30	63	39	8	4
O_{116}	1	2	7	15	14	1	2	1	22	63	39	8	4
O_{117}	3	1	7	15	14	1	2	3	22	63	39	8	4
O_{118}	2	3	13	16	12	2	1	2	24	61	41	5	2
O_{119}	1	2	9	15	14	2	4	3	18	61	41	5	2
O_{120}	3	4	9	15	14	2	1	3	28	61	41	5	2
O_{121}	0	1	13	15	-	2	1	-	24	65	40	6	3
O_{122}	1	2	14	14	12	3	2	4	18	65	40	6	3
O_{123}	2	4	13	5	13	4	2	3	18	65	40	6	3
O_{124}	3	2	13	14	12	3	4	2	22	63	39	8	4
O_{125}	4	1	12	13	11	2	3	2	30	63	39	8	4
O_{126}	1	2	7	15	14	1	2	1	22	63	39	8	4
O_{127}	3	1	7	15	14	1	2	3	22	63	39	8	4
O_{128}	2	3	13	16	12	2	1	2	24	61	41	5	2
O_{129}	1	2	9	15	14	2	4	3	18	61	41	5	2
O_{130}	3	4	9	15	14	2	1	3	28	61	41	5	2

La date de validité limite des composants (υ_{ijk}), la date de livraison du produit fini ($d_{P_i}^{liv}$), la durée de vie du produit fini (Dv_{P_i}) et la durée de retour du produit fini (Dr_{P_i}) sont exprimées en jours.

Prenant en compte les données présentées, la minimisation de la durée maximale d'exécution de la dernière opération, le coût des produits périmés et le coût du discount de distribution sont calculés en utilisant les relations II.4, II.5 et II.6.

II.5.2. Résultats de mise en œuvre de l'approche SGA_1

L'algorithme SGA_1 proposé, utilisant le codage CLOA est appliqué en vue d'obtenir le meilleur ordre de passage des opérations pour minimiser les critères *C1*, *C2* et *C3*.

Le tableau II.5 présente les valeurs des paramètres utilisés par l'approche SGA_1, pour les deux benchmarks, afin de générer de bonnes solutions. Ces paramètres sont la taille de la population, le nombre d'itérations, les probabilités de croisement et de mutation.

Tableau II.5. Valeurs des paramètres relatifs à l'algorithme SGA_1

	Benchmark 1	Benchmark 2
Taille de la population	10	30
Nombre d'itérations	300	600
Probabilité de croisement	0.7	0.7
Probabilité de mutation	0.01	0.01

II.5.2.1. *Choix du séquencement des critères*

L'approche proposée repose sur l'idée d'utiliser les algorithmes génétiques d'une façon séquentielle. Trois algorithmes génétiques sont utilisés pour optimiser respectivement les trois critères. Le résultat trouvé dans le premier algorithme est injecté dans le deuxième et ainsi de suite jusqu'à l'obtention d'un résultat qui satisfait les trois critères. Ainsi, pour montrer l'efficacité de notre approche, nous appliquons d'abord la méthode pour optimiser respectivement les critères *C1*, *C2* et *C3* puis nous procédons à une permutation des critères pour l'étude de l'effet de l'ordre des critères sur la qualité du résultat final.

En conséquence, pour trois critères *C1*, *C2* et *C3*, nous allons avoir six algorithmes à savoir : SGA123, SGA132, SGA213, SGA231, SGA321 et SGA312, les chiffres indiquant dans

l'ordre les numéros des critères utilisés. Le tableau II.6 résume les résultats obtenus par application de l'algorithme SGA_1 proposé pour traiter le benchmark 1 et le benchmark 2.

Tableau II.6. Résultats de simulation obtenus par l'algorithme SGA_1

	Benchmark 1						**Benchmark 2**					
	SGA123	**SGA132**	**SGA213**	**SGA231**	**SGA321**	**SGA312**	**SGA123**	**SGA132**	**SGA213**	**SGA231**	**SGA321**	**SGA312**
C1	22	22	24	23	25	26	67	67	68	68	68	68
C2	8	8	8	8	13	14	59	59	59	58	62	64
C3	31	27	36	35	18	18	76	77	70	68	58	56

Le tableau II.6, montre l'influence du choix du premier critère sur le résultat final obtenu par l'algorithme proposée SGA_1. En effet, la valeur minimale de chaque critère est obtenue lorsque l'algorithme proposé commence par ce critère. Par exemple, la valeur minimale du critère *C1* est 22 pour le benchmark 1 et 67 pour le second benchmark ; ces valeurs sont obtenues par les algorithmes SGA123 et SGA132. De même, pour le critère *C2* et *C3*, la valeur minimale de chaque critère est obtenue respectivement par les algorithmes SGA213, SGA231 et les algorithmes SGA321 et SGA312.

II.5.3. Résultats de mise en œuvre de l'approche PGA_1

Le tableau II.7 présente les valeurs des paramètres utilisés par l'approche PGA_1, pour les deux benchmarks, afin de générer de bonnes solutions. Ces paramètres sont le nombre de sous populations, la taille de la population, le nombre d'itérations, les probabilités de croisement et de mutation.

Dans le cas de l'approche PGA_1 proposée, le nombre de sous-populations dépend du nombre de critères à résoudre. Le nombre de sous-populations choisi, pour notre cas d'étude, est de trois puisque trois critères à résoudre sont en présence.

Tableau II.7. Valeurs des paramètres relatifs à l'algorithme PGA_1

	Benchmark 1	Benchmark 2
Nombre de sous-populations	3	3
Taille de la population	10	30
Nombre d'itérations	100	200
Taux de croisement	0.7	0.7
Taux de mutation	0.01	0.01

Le tableau II.8 présente les résultats obtenus par application de l'algorithme PGA_1 proposé pour traiter le benchmark 1 et le benchmark 2.

Tableau II.8. Résultats de simulation obtenus par l'algorithme PGA_1

	Benchmark 1	Benchmark 2
C1	23	68
C2	13	64
C3	16	42

II.5.4. Résultats de mise en œuvre de l'approche PSGA_1

L'algorithme PSGA_1 proposé, utilisant le codage CLOA est appliqué en vue d'obtenir le meilleur ordre de passage des opérations pour minimiser les critères *C1*, *C2* et *C3*.

Le tableau II.9 résume les valeurs des paramètres utilisés par l'approche PSGA_1, pour les deux benchmarks, afin de générer de bonnes solutions. Ces paramètres sont le nombre de sous-populations, la taille de la population, le nombre d'itérations, les probabilités de croisement et de mutation.

Tableau II.9. Valeurs des paramètres relatifs à l'algorithme PSGA_1

	Benchmark 1	Benchmark 2
Nombre de sous-populations	2	2
Taille de la population	10	30
Nombre d'itérations	100	200
Taux de croisement	0.7	0.7
Taux de mutation	0.01	0.01

II.5.4.1. *Choix du séquencement des critères*

Dans le paragraphe II.5.2.1, nous avons montré l'influence du séquencement des critères dans le cas de l'approche SGA_1. Puisque l'approche PSGA_1 développée est une approche hybride des deux méthodes présentées précédemment à savoir PGA_1 et SGA_1, nous allons voir par la suite si le choix du séquencement des critères influe sur le résultat final.

Pour les trois critères *C1*, *C2* et *C3*, nous avons les trois algorithmes suivants : PSGA123, PSGA132 et PSGA321. Le tableau II.10 présente les résultats obtenus par application de l'algorithme PSGA proposé pour traiter le benchmark 1 et le benchmark 2.

Tableau II.10. Résultats de simulation obtenus par l'algorithme PSGA_1

	Benchmark 1			Benchmark 2		
	PSGA123	**PSGA132**	**PSGA321**	**PSGA123**	**PSGA132**	**PSGA321**
C1	26	24	24	69	69	69

C2	14	14	14	64	64	64
C3	12	12	13	43	40	40

Le tableau II.10 montre que le choix du séquencement des critères n'influe pas sur le résultat final. En effet, pour les deux benchmarks, on note que pour les trois algorithmes PSGA123, PSGA132 et PSGA321, les critères *C1* et *C2* n'ont pas changé de valeur et on note un léger changement du critère *C3*.

II.6. Etude comparative des performances des méthodes mises en œuvre

Dans cette partie, une étude comparative des performances de chaque méthode proposée par rapport aux méthodes existantes pour la résolution de problèmes multi-objectifs est présentée. La première méthode est basée sur l'approche Pareto-Optimalité (APO) et la deuxième basée sur la Pondération des Fonctions Objectifs (PFO). Nous allons ensuite, comparer les performances des méthodes proposées relativement à la qualité des résultats obtenus mais aussi sur le temps de compilation.

II.6.1. Comparaison des méthodes proposées avec les méthodes existantes

La méthode Pareto-optimalité et la méthode d'agrégation utilisent les mêmes valeurs des paramètres que ceux utilisés par l'algorithme SGA_1 proposé, tableau II.6.

On définit une fonction *F* comme la somme pondérée des trois fonctions objectifs *C1*, *C2* et *C3* avec des poids α_1, α_2 et α_3 définis en fonction de l'importance de ces trois critères :

$$F = \alpha_1 C_1 + \alpha_2 C_2 + \alpha_3 C_3 \qquad \text{(II.9)}$$

avec :

$$\alpha_i > 0 \text{ et } \sum_{i=1}^{3} \alpha_i = 1$$

Ces poids peuvent, par un choix adéquat de leurs valeurs, privilégier un critère par rapport à un autre. Dans notre cas, les valeurs des coefficients α_1, α_2 et α_3 sont choisis respectivement égaux à 0.4, 0.1 et 0.5. La minimisation du coût du discount de distribution est ainsi privilégiée par rapport à la minimisation du makespan et par rapport à la minimisation du coût des produits périmés.

II.6.1.1. ***Comparaison des performances de l'approche SGA_1 avec les méthodes existantes***

Pour montrer l'efficacité de l'approche SGA_1 développée, une comparaison avec la méthode Pareto-optimalité et la méthode d'agrégation est effectuée.

Le tableau II.6 donne le séquencement qui correspond au meilleur résultat pour les trois critères et le Tableau II.11 la valeur de la fonction *F* pour les six algorithmes définis précédemment.

Tableau II.11. Résultats relatifs à la fonction F pour chaque algorithme SGA_1

	Benchmark 1						**Benchmark 2**					
	SGA123	**SGA132**	**SGA213**	**SGA231**	**SGA321**	**SGA312**	**SGA123**	**SGA132**	**SGA213**	**SGA231**	**SGA321**	**SGA312**
F	25.1	23.1	28.4	27.5	20.3	20.8	70.7	71.2	68.1	67.1	62.4	61.6

Le tableau II.11 montre que l'algorithme SGA321 présente une meilleure valeur de la fonction *F* pour le premier benchmark alors que l'algorithme SGA312 présente une meilleure valeur de la fonction *F* pour le second benchmark. Ainsi, cet algorithme est retenu pour effectuer la comparaison avec l'approche Pareto-optimalité et l'approche basée sur la pondération des fonctions.

Le tableau II.12 résume les résultats obtenus par application des différentes méthodes utilisées pour traiter les deux benchmarks.

Tableau II.12. Résultats obtenus par les algorithmes APO, PFO et SGA_1

	Benchmark 1			Benchmark 2		
Méthode d'optimisation	APO	PFO	SGA_1	APO	PFO	SGA_1
F	25.5	24	20.3	82.4	85	61.6
CPU (s)	9.252	8.669	6.501	68,356	72,113	45,866

Le tableau II.12 montre que la méthode développée SGA_1 a donné un meilleur résultat pour le benchmark 1, puisque la valeur de la fonction *F* trouvée est inférieure à celles trouvées par la méthode Pareto-optimalité et la méthode d'agrégation. Ce résultat reste valable même lorsqu'on augmente le nombre d'opérations (benchmark 2).

Aussi, nous avons noté que du point de vue différence du temps de compilation, l'approche SGA_1 est plus rapide que les deux méthodes précédentes. En effet, la différence du temps de compilation, pour le benchmark 1, entre l'approche SGA_1 et l'approche Pareto-optimalité est approximativement de 3 secondes et entre l'approche SGA_1 et l'approche par pondération des fonctions objectifs est approximativement de 2 secondes.

Aussi, la différence du temps de compilation, pour le second benchmark, entre l'approche SGA_1 et la méthode Pareto-optimalité est approximativement de 22.5 secondes et entre l'approche SGA_1 et la méthode d'agrégation est approximativement de 26.3 secondes.

L'approche SGA_1 a permis, ainsi, de trouver un meilleur résultat que les deux autres méthodes et ceci en un temps de compilation plus rapide.

II.6.1.2. ***Etude comparative des performances de l'algorithme PGA_1 proposé***

Le tableau II.13 donne les résultats obtenus par application des différentes méthodes utilisées pour traiter les deux benchmarks étudiés.

Tableau II.13. Résultats obtenus par les algorithmes APO, PFO et PGA_1

	Benchmark 1			Benchmark 2		
Méthode d'optimisation	APO	PFO	PGA_1	APO	PFO	PGA_1
F	25.5	24	18.5	82.4	85	54.6
CPU (s)	9.252	8.669	3.547	68,356	72,113	31,387

A partir du tableau II.13, nous notons que la méthode développée PGA_1 a donné un meilleur résultat pour les deux benchmarks, puisque la valeur de la fonction F trouvée est inférieure à celles trouvées par la méthode Pareto-optimalité et la méthode d'agrégation.

Aussi, on note que du point de vue différence du temps de compilation, l'approche PGA_1 est plus rapide que les deux méthodes précédentes. En effet, le temps de compilation, pour le benchmark 1, entre l'approche PGA_1 et la méthode Pareto-optimalité est approximativement de 6 secondes et entre l'approche PGA_1 et la méthode d'agrégation est approximativement de 5 secondes.

La différence du temps de compilation, pour le second benchmark, entre l'approche PGA_1 et la méthode Pareto-optimalité est, par ailleurs, approximativement de 37 secondes et entre l'approche PGA_1 et la méthode d'agrégation est approximativement de 41 secondes.

Finalement, on constate que l'approche PGA_1 développée permet de trouver un meilleur résultat que les méthodes classiques et cela en un temps de compilation plus court.

II.6.1.3. ***Etude comparative des performances de l'algorithme PSGA_1 proposée***

A l'instar de l'approche SGA_1, nous cherchons à déduire pour l'approche PSGA_1, le séquencement qui donne le meilleur résultat pour les trois critères. Le tableau II.14 donne les valeurs de la fonction F pour les trois algorithmes définis précédemment.

Tableau II.14. Résultats relatifs à la fonction F pour chaque algorithme PSGA_1

	Benchmark 1			Benchmark 2		
	PSGA123	PSGA132	PSGA321	PSGA123	PSGA132	PSGA321
F	17.8	17	17.5	55.4	54	54

Le tableau II.14 montre que l'algorithme PSGA132 présente de meilleures valeurs de la fonction *F* pour les deux benchmarks.

Ces deux algorithmes vont être comparés avec la méthode Pareto-optimalité et la méthode d'agrégation pour montrer leur efficacité.

Le tableau II.15 résume les résultats obtenus par application des différentes méthodes utilisées pour traiter les deux benchmarks étudiés.

Tableau II.15. Résultats obtenus par les algorithmes APO, PFO et PSGA_1

	Benchmark 1			Benchmark 2		
Méthode d'optimisation	APO	PFO	PSGA_1	APO	PFO	PSGA_1
F	25.5	24	17	82.4	85	54
CPU (s)	9,252	8.669	4.008	68,356	72,113	33.405

Le tableau II.15 montre que la méthode développée PSGA_1 a donné un meilleur résultat pour le benchmark 1, puisque la valeur de la fonction *F* trouvée est inférieure à celles trouvées par la méthode Pareto-optimalité et la méthode d'agrégation. Ces résultats restent valables même lorsqu'on augmente le nombre d'opérations (benchmark 2).

On note aussi que du point de vue du temps de compilation, l'approche PSGA_1 est plus rapide que les deux méthodes précédentes. En effet, la différence du temps de compilation, pour le benchmark 1, entre l'approche PSGA_1 et la méthode Pareto-optimalité, est approximativement de 5,2 seconde et entre l'approche PSGA_1 et la méthode d'agrégation est approximativement de 4,6 seconde.

La différence du temps de compilation, pour le second benchmark, entre l'approche PSGA_1 et la méthode Pareto-optimalité est, par ailleurs, approximativement de 35 secondes et entre l'approche PSGA_1 la méthode d'agrégation est approximativement de 39 secondes.

Finalement, nous pouvons conclure que l'approche développé à savoir PSGA_1 a permis de trouver un meilleur résultat que les méthodes classiques et en un temps de compilation plus court.

II.6.2. Comparaison des performances des approches SGA_1, PGA_1 et PSGA_1

Dans cette partie, nous allons comparer les performances des trois méthodes proposées. La comparaison se base non seulement sur la qualité des résultats obtenus mais aussi sur la vitesse de compilation.

Le tableau II.16 présente le résultat trouvé pour les trois approches développées à savoir SGA_1, PGA_1 et PSGA_1.

Tableau II.16. Résultats obtenus par les différentes méthodes développées

	Benchmark 1			Benchmark 2		
Méthode d'optimisation	SGA_1	PGA_1	PSGA_1	SGA_1	PGA_1	PSGA_1
F	20.3	18.5	17	61.6	54.6	54
CPU (en seconde)	6.1547	3.5474	4.0085	45,8669	31,3875	33.4051

On constate, que l'approche PSGA_1 a donné un meilleur résultat que les deux autres approches puisque le minimum de la fonction F pour les deux benchmarks a été obtenu par l'approche PSGA_1.

Le tableau II.17 présente la différence du temps de compilation du CPU en secondes, trouvée pour les deux benchmarks utilisant les trois approches SGA_1, PGA_1 et PSGA_1 développées.

Tableau II.17. Différences du temps de compilation du CPU

	Benchmark 1			Benchmark 2		
	SGA_1	PGA_1	PSGA_1	SGA_1	PGA_1	PSGA_1
SGA_1	0	-2.607	-2.146	0	-14.479	-12.461
PGA_1	2.607	0	-0.461	14.479	0	2.017
PSGA_1	2.146	0.461	0	12.461	-2.017	0

Le tableau II.17 montre que l'algorithme PGA_1 est plus rapide que les autres méthodes. On constate aussi que la différence du temps de compilation entre PGA_1 et PSGA_1 n'est pas

grande. En effet, pour le benchmark 1, la différence est de moins 1 seconde et pour le benchmark 2, la différence est inférieure à 3 secondes.

L'algorithme SGA_1 est donc le moins rapide des trois approches développées.

Aussi, nous notons que le choix de séquencement des critères n'a pas d'effet sur l'approche PSGA_1 contrairement à l'approche SGA_1. Nous avons, ainsi, améliorer la première méthode développée, à savoir SGA_1, en ajoutant la stratégie parallèle.

Finalement, on constate que l'approche développée PSGA_1 permet de trouver un meilleur résultat que les deux autres approches développées et que la vitesse du CPU est assez satisfaisante, même si, l'approche PGA_1 est la plus rapide.

II.7. Conclusion

La résolution des problèmes d'ordonnancement dans le cas des industries agroalimentaires par des approches basées sur les algorithmes génétiques dans le cas multi-objectifs, constitue la contribution principale de ce chapitre. Après avoir introduit les approches développées à savoir SGA_1, PGA_1 et PSGA_1, une résolution de deux problèmes d'ordonnancement en industries agroalimentaires est proposée. Les différents critères, leur combinaison formant la fonction fitness ainsi que les contraintes relatives au problème de l'ordonnancement, des ateliers à une machine en industries agroalimentaires et le codage CLOA ainsi que les différents opérateurs à utiliser et leur fonctionnement proposés, sont ensuite présentés.

La résolution de ces deux problèmes, ainsi présentés, et la comparaison des approches basées sur les algorithmes génétiques par rapport à l'approche Pareto-optimalité et l'approche basée sur la pondération des fonctions a montré l'efficacité de nos approches développées du point de vue aussi bien qualité de la solution que temps de compilation.

Une étude comparative des performances des approches développées a abouti à la conclusion que l'approche PSGA_1 permet d'atteindre une meilleure solution que l'approche SGA_1 ou l'approche PGA_1.

CHAPITRE III

HYBRIDATION DE METAHEURISTIQUES POUR LA RESOLUTION DE PROBLEMES D'ORDONNANCEMENT

III.1. Introduction

Durant ces dernières années, l'intérêt portant sur les métaheuristiques n'a cessé de grandir. En effet, les métaheuristiques ont montré leur capacité dans de vastes domaines d'application en résolvant de nombreux problèmes d'optimisation [Talbi, 09].

La diversité de ces méthodes de résolution et les avantages qu'elles possèdent séparément, permettent d'envisager de les utiliser conjointement pour résoudre des problèmes complexes.

Il semble donc intéressant de faire coopérer ou d'hybrider différentes méthodes afin de combiner leurs qualités complémentaires.

Dans ce chapitre, nous nous intéressons aux méthodes hybrides constituées seulement de métaheuristiques. En première partie de ce chapitre, une classification des métaheuristiques selon Talbi [Talbi, 02] est présentée, suivie, de la définition des métaheuristiques à parcours que nous allons utiliser lors de l'hybridation. Ensuite, le schéma de coopération des deux approches développées est proposé. On termine ce chapitre par une étude comparative des approches hybrides mises en œuvre.

III.2. Généralités sur l'hybridation des métaheuristiques

Les métaheuristiques sont des méthodes approchées qui ont prouvé leur efficacité pour la résolution de problèmes NP-difficiles. Quelque soit la méthode choisie, elles présentent des avantages et des inconvénients. La coopération de méthodes peut permettre de trouver de bons compromis où les avantages et les défauts de chaque méthode se compensent.

Par la suite, sont présentées des métaheuristiques, suivies de la classification hiérarchique de leur coopération.

III.2.1. Présentation de métaheuristiques

Les métaheuristiques peuvent être définies comme des algorithmes d'optimisation de type stochastiques, progressant vers un optimum par un échantillonnage d'une fonction objectif dont le but est la résolution de problèmes d'optimisation difficiles.

Deux classes de métaheuristiques peuvent être distinguées : les métaheuristiques à base de solution unique (ou métaheuristiques à parcours) et les métaheuristiques à base de population. Les métaheuristiques, dont la recherche tabou et le recuit simulé, de la première classe manipulent et transforment une seule solution durant le processus de recherche, alors que dans les métaheuristiques à base de populations dont les algorithmes génétiques et les colonies de fourmis, un ensemble de solutions, appelé population, évolue en parallèle.

Lors de la conception d'une métaheuristique, deux critères contradictoires doivent être pris en compte : d'une part l'exploration de l'espace de recherche (diversification), et d'autre part l'exploitation des meilleures solutions trouvées (intensification). Les métaheuristiques à parcours sont plutôt axées sur l'exploitation de l'espace de recherche. Les régions prometteuses sont explorées localement dans l'espoir de trouver de meilleures solutions. Les métaheuris-

tiques à base de populations sont plutôt de type exploratoire et permettent une meilleure diversification de l'espace de recherche.

Pour être performante, une méthode de recherche doit être capable d'associer les deux critères définis précédemment. Or, une méthode de recherche est rarement aussi efficace pour exploiter que pour explorer l'espace de recherche. Une solution consiste à ajouter des mécanismes complémentaires dans une méthode de recherche donnée. Cependant, cette opération peut s'avérer délicate et peut nuire au fonctionnement de la méthode [Duvier, 00]. De ce fait, il est intéressant de faire coopérer une ou plusieurs méthodes ayant une capacité d'exploration à une méthode ou plusieurs méthodes ayant une capacité d'exploitation. C'est ainsi que les métaheuristiques ont prouvé leur efficacité pour la résolution de problèmes d'optimisation multi-objectifs [Ehrgott, 08], [Talbi, 09].

III.2.2. Classification hiérarchique

D'après Talbi [Talbi, 02], la classification hiérarchique des métaheuristiques coopératives, consignée dans la figure III.1, traite du niveau haut et du mode de coopération.

D'après la taxinomie proposée par Talbi, deux niveaux de coopération peuvent être distingués. Dans le cas d'*une hybridation bas niveau*, une fonction interne d'une métaheuristique est remplacée par une autre métaheuristique, alors que dans le cas d'*une hybridation haut niveau*, les différentes métaheuristiques sont autonomes et aucune méthode n'est modifiée.

Les deux modes de coopération sont *le mode relais* et *le mode teamwork*.

Dans une coopération en *mode relais*, les métaheuristiques coopératives opèrent les unes après les autres dans un ordre pré-établi. Chaque méthode reçoit en entrée le résultat trouvé par la méthode précédente. Le *mode teamwork* représente un modèle d'optimisation coopératif où les différentes méthodes évoluent en parallèle.

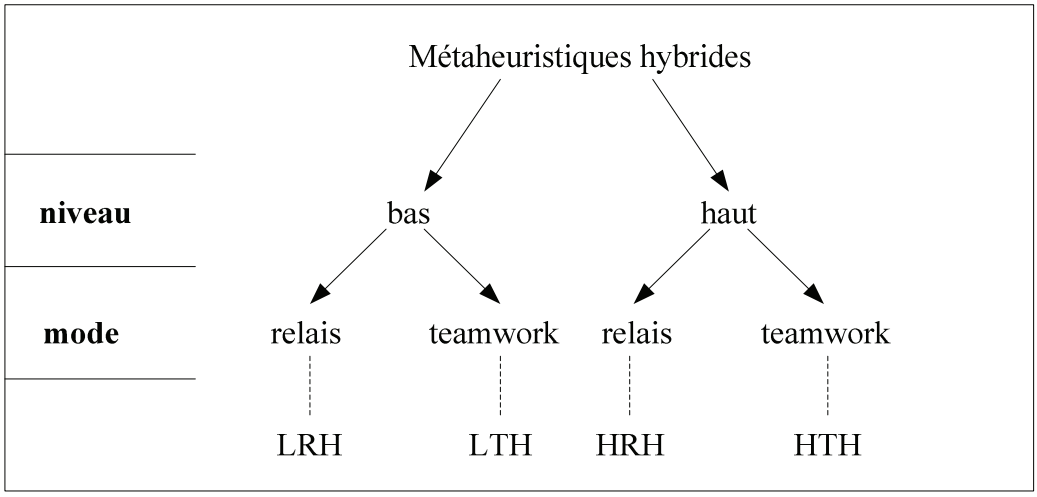

Figure III.1. Classification hiérarchiques des métaheuristiques coopératives [Talbi, 02]

A partir de cette classification, quatre classes de métaheuristiques coopératives sont obtenues.

- ✓ Coopération relais de bas niveau (Low-level Relay Hybrid, LRH) : Dans cette classe, on trouve les méthodes à base de solution unique où une autre méthode d'optimisation est insérée. Dans ce type de coopération, les deux (ou éventuellement plus) méthodes utilisées perdent leur intégrité pour produire un nouvel algorithme. Un exemple de ce type de coopération consiste à combiner le recuit simulé avec la méthode par descente, proposé dans [Martin, 92], pour résoudre le problème du voyageur de commerce.
- ✓ Coopération co-évolutionnaire de bas niveau (Low-level Teamwork Hybrid, LTH) : Cette classe regroupe les métaheuristiques à base de populations où un (ou plusieurs) opérateur(s) est (sont) remplacé(s) par une (ou plusieurs) méthode(s) d'optimisation. On cite les algorithmes mimétiques, qui sont des algorithmes évolutionnaires dans lesquels un opérateur est remplacé par une métaheuristique à base de solution unique. Cette coopération permet de combiner le pouvoir d'exploration des algorithmes à base de populations et celui d'intensification des méthodes à base de solution unique. Généralement, l'opérateur de mutation est remplacé, par une recherche par descente [Suh, 87] ou par une recherche tabou [Fleurent, 94] ou par un recuit simulé [Brown, 89]. Dans d'autre cas, l'opérateur de recombinaison est remplacé par une méthode d'optimisation telle que l'algorithme glouton [Goldberg, 87].
- ✓ Coopération relais de haut niveau (High-level Relay Hybrid, HRH) : Dans cette classe de coopération, les méthodes d'optimisation sont utilisées de manière séquentielle gardant ainsi leur intégrité. Le (ou les) résultat(s) final(aux) de la première méthode devient (ou deviennent) la (ou les) solution(s) initiale(s) de la méthode suivante. Cet en-

chaînement de méthodes peut notamment permettre de procéder à une phase de diversification pour couvrir une grande partie de l'espace de recherche avant d'arriver à une phase d'intensification pour améliorer les solutions obtenues. Dans [Talbi, 94], un algorithme génétique est suivi d'une Recherche Tabou afin d'améliorer les solutions obtenues.

- ✓ Coopération co-évolutionnaire de haut niveau (High-level Teamwork Hybrid, HTH) : Cette classe de coopération correspond à l'exécution en parallèle de méthodes d'optimisation pouvant communiquer entre elles durant leur exécution. Un exemple de ce type de coopération est l'exécution parallèle d'un algorithme génétique. Dans ce modèle, la population initiale est divisée en sous-populations qui sont réparties entre différentes îles sur lesquelles un algorithme génétique est lancé. Ainsi, chaque île correspond à une recherche de solutions dans une partie de l'espace de recherche. De manière périodique, un certain nombre d'individus migre entre les îles afin que les sous-populations des différentes îles contribuent toutes à produire les meilleures solutions possibles. En effet, de l'information extérieure peut permettre, à un algorithme génétique qui n'évoluait plus, de pouvoir être relancé afin de tendre vers les meilleures solutions possibles. Parmi les travaux basés sur cette coopération, on cite les algorithmes génétiques parallèles [Tanese, 87], la recherche tabou parallèle [De Falco, 94] et le recuit simulé parallèle [De Falco, 95].

III.2.3. Métaheuristiques coopératives et optimisation multi-objectifs

En optimisation multi-objectifs, quelques ajustements sur la taxinomie présentée ci-dessus sont souvent nécessaires. Ils sont dûs au fait que le but de la recherche est de fournir un ensemble de solutions non-dominées, et non une solution unique.

Dans les études sur les métaheuristiques hybrides, dédiées à la résolution de problèmes d'optimisation multi-objectifs, Erghott et Gandibleux identifient trois catégories de méthodes d'hybridation entre un algorithme à base de populations et un algorithme de recherche locale [Ehrgott, 08] : une hybridation pour rendre une méthode plus agressive, une hybridation pour diriger une méthode, et une hybridation pour exploiter des forces complémentaires. Les deux premières catégories peuvent être vues comme appartenant à la classe LTH. Enfin, la troisième catégorie rassemble les métaheuristiques coopératives de la classe HRH, où plusieurs méthodes (généralement deux) sont exécutées en séquence.

III.3. La Recherche Tabou (RT)

III.3.1. Introduction

La recherche avec Tabou (ou recherche tabou) est une métaheuristique originalement développée par Glover [Glover, 89] et [Glover, 90]. Cette méthode a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation, comme les problèmes de routage de véhicules [El Fallahi, 08], les problèmes d'affectation quadratique [James, 09] ou les problèmes de coloration de graphes [Porumbel, 09]. Aussi la recherche tabou a montré son efficacité pour la résolution de problèmes d'ordonnancement de type flow shop [Ekşioğlu, 08], job shop [Zhang, 07] ou open job [Liaw, 99] mais aussi ceux à une machine [Weng, 02], [Choobineh, 06], [Venditti, 10].

III.3.2. Principe de fonctionnement

Le principe de la recherche Tabou, dont l'organigramme est illustré dans la figure III.2, est similaire à celui des méthodes de recherche locales itératives. Il est basé sur la création et l'évaluation d'un voisinage de solutions [Kammarti, 05], [Karray, 08]. Sa principale particularité réside dans la mise en œuvre d'un mécanisme inspiré de la mémoire humaine. En effet, cette méthode combine une procédure de recherche locale avec des règles et des mécanismes lui permettant de surmonter l'obstacle des extremums locaux et d'éviter en même temps d'être piégé dans un cycle.

La recherche tabou fonctionne avec une seule configuration à la fois qui est actualisée à chaque itération. A chaque itération, le passage d'une configuration *s* à une autre *t* se fait en deux étapes :

- l'ensemble des voisins *V(s)* de *s* est construit à partir de mouvement élémentaire de *s*,
- la fonction objectif *f* du problème est évaluée en chacune des configurations de *V(s)*, la configuration *t* qui succède à *s* est la configuration de l'ensemble *V(s)* ou *f* prend la valeur minimale.

Pour éviter l'apparition d'un cycle et le retour à une configuration déjà retenue, une liste de mouvements interdits ou liste tabou est mise à jour. Cette liste contient les m derniers mouvements effectués. Toutefois, une fonction d'aspiration pourra autoriser le retour vers une solution déjà visitée et ainsi permettre l'exploration d'autres zones et obtenir de meilleures solutions.

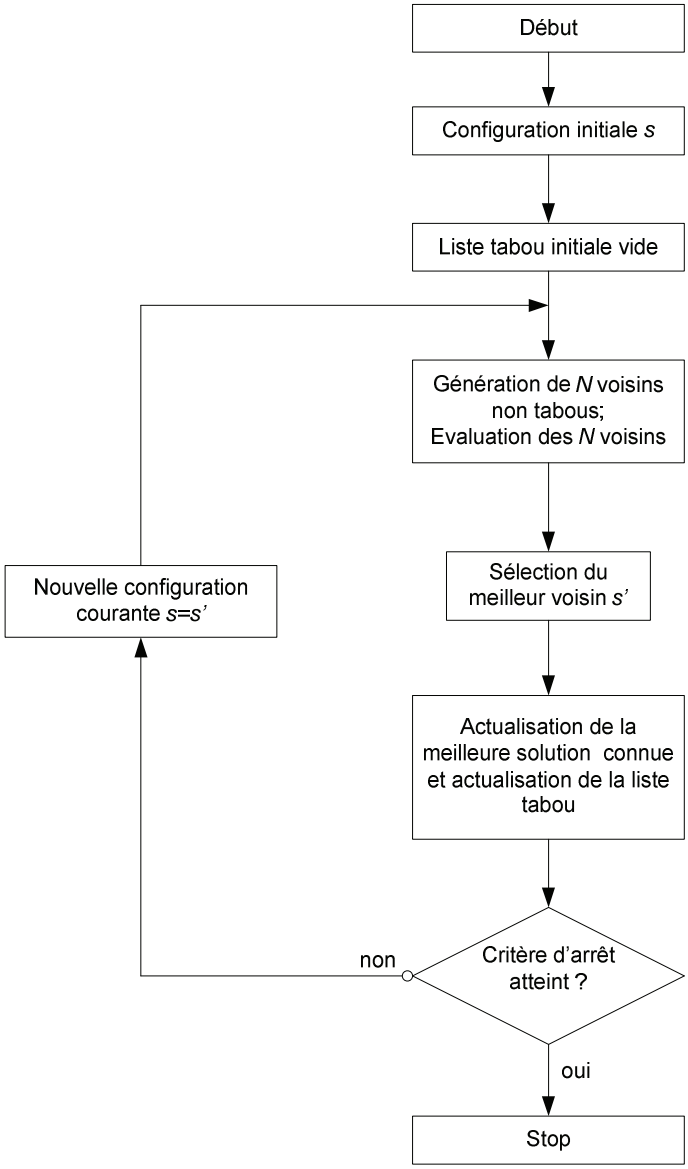

Figure III.2. Organigramme de la recherche tabou

III.3.3. Solution initiale

L'efficacité des méthodes basées sur le principe de la recherche locale dépend de la qualité de la solution initiale choisie [Jain, 99]. Pour notre application, l'algorithme de génération de la solution initiale aléatoire, présenté dans la figure III.3, est initialisé d'abord avec une séquence canonique *{0, 1, 2, ..., n}*, pour avoir une solution initiale aléatoire de taille *n*. Ensuite, des mutations sont effectuées sur cette solution.

Algorithme : Génération Aléatoire de la solution initiale

Variable locale S : vecteur de taille n ;

m : nombre de mutations, tel que $m \in \left[0 \ldots n/2\right]$

Début

Pour *i=0* à *n-1* **faire**

$S[i] \leftarrow i$

Fin Pour

Générer Aléatoirement un nombre de mutations m ;

Pour *k=0* à m **faire**

Générer Aléatoirement deux positions *P1* et *P2* / $P_1, P_2 \in [0 \ldots n-1]$ et $P_1 \neq P_2$;

Permuter $S[P_1]$ et $S[P_2]$

Fin Pour

Fin

Figure III.3. Algorithme de génération de la solution initiale

III.3.4. Voisinage

L'ensemble *N(s)* est généralement défini comme l'ensemble des solutions voisines de la configuration courante *s*, que nous considérons d'un point de vue pratique comme l'ensemble des modifications que nous pouvons apporter à *s*. Nous appelons mouvement, une modification apportée à une solution.

L'ensemble *N(s)* est l'ensemble de solutions valables susceptibles d'être obtenues en appliquant à *s* un mouvement *m* appartenant à l'ensemble *M* des mouvements possibles. Dans la littérature et dans le cas de la recherche tabou, l'application d'un mouvement *m* à une solution *s,* notée $s \oplus m$, conduit à l'expression suivante du voisinage $N(s) = \{s' / s' = s \oplus m, m \in M\}$, [Kammarti, 06].

Parmi les mouvements possibles susceptibles d'engendrer des solutions voisines admissibles, nous citons :

- l'inversion de deux éléments successifs dans la solution initiale, figure III.4,
- la permutation de deux éléments quelconques distincts, figure III.5,
- le déplacement d'un élément de sa place d'origine à une nouvelle place, figure III.6.

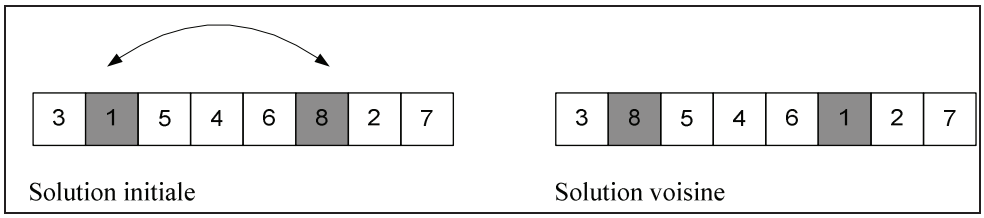

Figure III.4. Inversion de deux éléments successifs

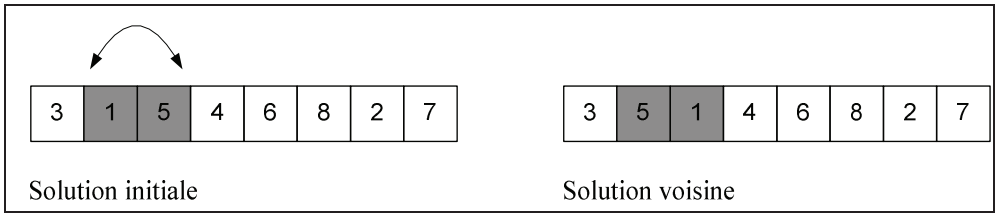

Figure III.5. Permutation de deux éléments

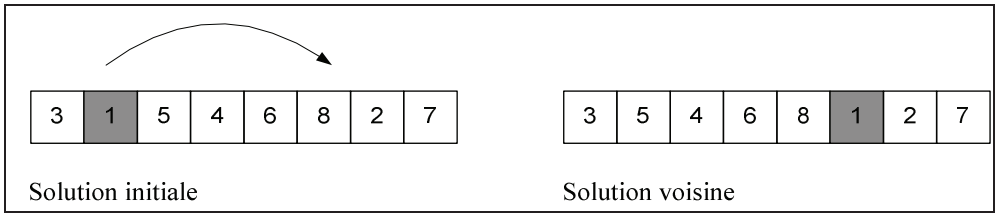

Figure III.6. Déplacement d'un élément

Le premier type de mouvement permet d'avoir un voisinage de taille $(n-1)$, le deuxième un voisinage de taille $\left(\frac{n(n-1)}{2}\right)$ et le troisième un voisinage de taille $(n(n-2)+1)$ [Dréo, 03].

III.3.5. Mémoire tabou

La recherche tabou est une méthode itérative qui explore un espace de solutions. La mémoire tabou ou la liste tabou représente un élément fondamental de cette métaheuristique, puisqu'elle permet d'éviter les blocages dans des minima locaux, en interdisant de repasser sur des configurations de l'espace de recherche, précédemment visitées.

La gestion de la mémoire suit la stratégie FIFO (First In First Out) qui signifie que l'ordre de sortie des éléments de la liste correspond à celui de leur insertion.

La taille de la liste est statique et dépend des performances de la recherche tabou. Si elle est trop petite, elle engendre beaucoup de cycles et si elle est trop grande, elle engendre un phénomène de dispersion de la recherche.

Pour construire la liste tabou, deux techniques sont utilisées : soit on sauvegarde uniquement les mouvements qu'on s'interdit de reproduire pendant un certain nombre d'itérations, soit on sauvegarde la solution qu'on s'interdit de visiter pendant un certain nombre d'itérations.

III.3.6. Critère d'aspiration

Au cours de la recherche, il se peut qu'un mouvement interdit donne ou dirige la recherche vers une solution meilleure que toutes celles déjà visitées. Pour pallier à ce problème et pouvoir utiliser ce mouvement, nous modifions son statut tabou et nous disons que ce mouvement est aspiré. Il est, cependant, possible d'élaborer d'autres critères d'aspiration plus complexes. L'inconvénient de revenir fréquemment à l'aspiration est que, dans certains cas, elle peut éliminer la protection offerte par la liste Tabou vis-à-vis des cycles.

III.4. Le Recuit Simulé (RS)

III.4.1. Introduction

Le recuit simulé [Kirkpatrick, 83] trouve ses origines dans la thermodynamique. Cette méthode est issue d'une analogie entre le phénomène physique de refroidissement lent d'un corps en fusion, qui le conduit à un état solide, de basse énergie. Depuis son apparition, la méthode du recuit simulé a prouvé son efficacité dans plusieurs domaines comme les problèmes de traitement des images [Nakib, 06] ou les problèmes d'ordonnancement [Low, 05], [Zhang, 10], [Andresen, 08]. Aussi, le recuit simulé a montré son efficacité pour la résolution de problèmes d'ordonnancement à une machine [Köktener, 00], [Drira, 08], [Jin, 09].

III.4.2. Du recuit réel au recuit simulé

Pour modifier l'état d'un matériau, le physicien dispose d'un paramètre de commande : la température [Dréo, 03]. Le recuit constitue ainsi une stratégie de contrôle de la température en vue de trouver un état optimum. La croissance d'un monocristal est pris comme exemple pour expliquer cette stratégie. La technique de recuit consiste à chauffer préalablement le matériau pour lui conférer une énergie élevée. Puis, le matériau est refroidi lentement, en marquant des paliers de température de durée suffisante ; si la descente en température est trop rapide, il apparaît des défauts qui peuvent être éliminés par réchauffement local. Cette stratégie de baisse contrôlée de la température conduit à un état solide cristallisé stable, correspondant à un minimum absolu de l'énergie. La technique opposée est celle de la trempe, qui consiste à abaisser très rapidement la température du matériau ; dans ce cas, une structure amorphe rela-

tive à un état métastable correspondant à un minimum local de l'énergie est obtenue. Ainsi, avec la technique du recuit, le refroidissement du matériau a provoqué une transformation désordre-ordre, tandis que la technique de la trempe a abouti à figer un état désordonné [Dréo, 03].

Cette technique a donné naissance à la méthode du recuit simulé. Le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coût plus faible tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût.

Cette méthode consiste à introduire un paramètre de contrôle qui joue le rôle de la température. La température du système à optimiser doit avoir le même effet que la température du système physique, c'est-à-dire conduire vers l'état optimal si elle est abaissée de façon lente et bien contrôlée (technique de recuit) et vers un minimum local si elle est abaissée brutalement (technique de trempe).

III.4.3. Algorithme de recuit simulé

L'algorithme se base sur trois résultats de la physique statique.

1. Pour une température donnée T, l'équilibre thermodynamique d'un système est atteint.

   La probabilité, pour ce système, de posséder une énergie donnée E, est proportionnelle au facteur de *Boltzmann* : $\exp(-E/K_B T)$, où K_B désignant la constante de Boltzmann.

2. Le deuxième résultat s'appuie sur l'algorithme de *Metropolis* pour simuler l'évolution d'un système physique vers son équilibre thermodynamique à une température T donnée. Le principe de cet algorithme est le suivant : partant d'une configuration donnée, le système subit une modification élémentaire ; si cette transformation a pour effet de diminuer la fonction objectif f (ou énergie) du système, elle est acceptée, si elle provoque au contraire une augmentation Δf de la fonction objectif, elle peut être acceptée tout de même, avec la probabilité $\exp(-\Delta f / T)$.
3. La notion de l'équilibre thermique dépend de la chaine de Markov qui est l'ensemble des configurations explorées à température constante. L'équilibre thermique est caractérisé par la convergence de la distribution des énergies calculées sur les différentes configurations de la chaîne de Markov vers une loi normale. En pratique, ce critère d'équilibre thermique, trop sévère, est remplacé par la notion de quasi-équilibre. Ce

dernier est atteint lorsque la longueur de la chaîne de Markov, à savoir le nombre de configurations explorées, est suffisamment grande.

Kirkpatrick a fait une analogie entre l'optimisation et le phénomène physique de refroidissement, en faisant une correspondance entre arrangements des atomes et paramètres de conception, énergie et fonction objectif à minimiser, minimum de l'énergie et minimum global, chaîne de Markov et nombre de configurations explorées à température constante.

Toutefois, le concept de température d'un système physique n'a pas d'équivalent direct avec le problème à optimiser. Ainsi, le paramètre température T est un paramètre de contrôle, indiquant le contexte dans lequel se trouve le système à savoir le stade de la recherche. Le critère de Metropolis détermine si une nouvelle configuration générée présente une variation de la fonction objectif acceptable et permet aussi de sortir des minima locaux quand la température est élevée.

La figure III.7 présente les étapes de l'algorithme du recuit simulé. Celui-ci débute par une température initiale élevée et une configuration x initiale prise au hasard. A chaque itération, x varie; si cette variation diminue la fonction f, elle est acceptée sinon la règle de Metropolis est appliquée. Si l'équilibre thermodynamique est atteint, c'est-à-dire lorsque les variations élémentaires de x ne font plus varier l'énergie du système, à savoir la fonction f, la température est diminuée lentement et les modifications de x sont reprises jusqu'à atteindre un seuil préalablement choisi.

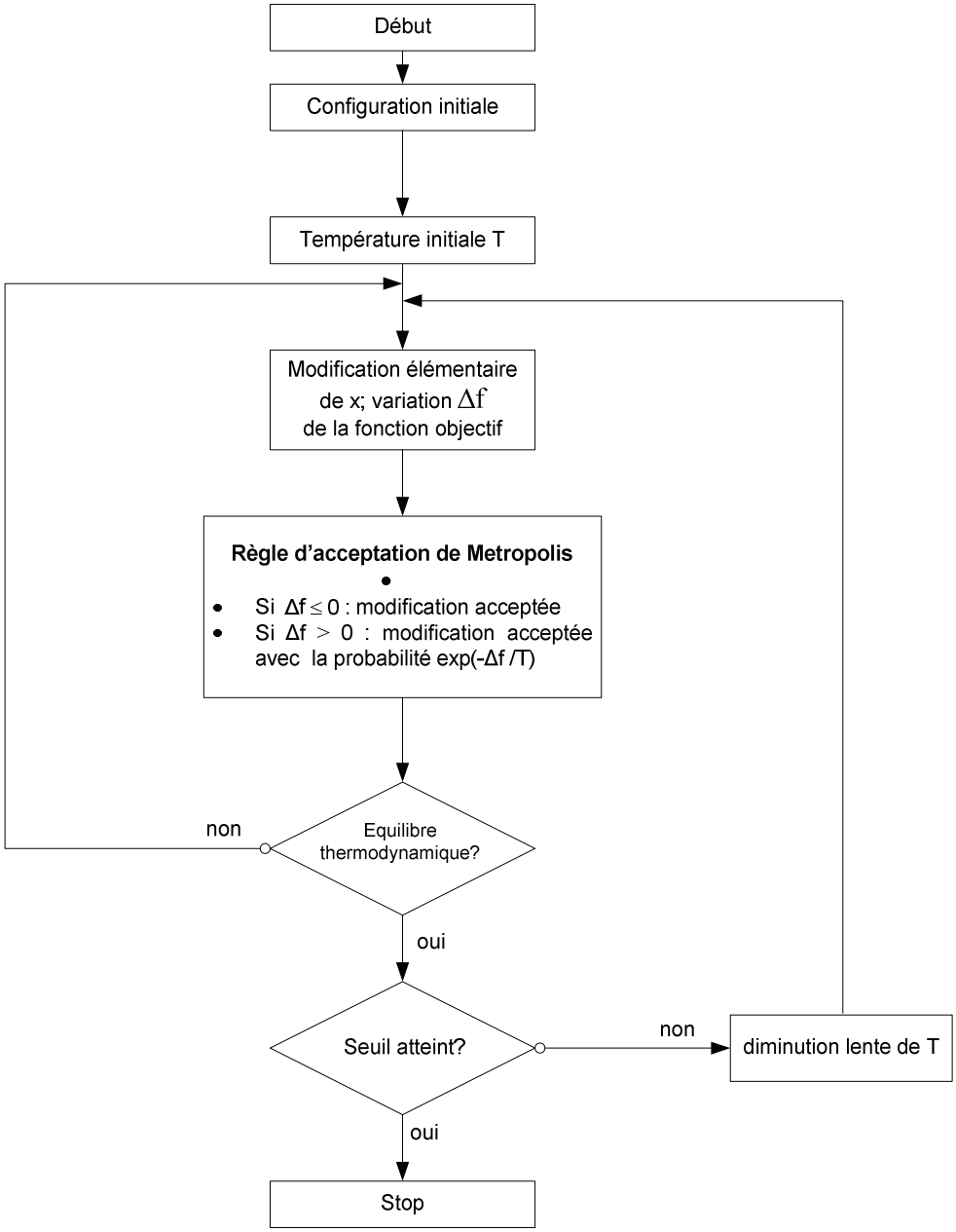

Figure III.7. Organigramme de l'algorithme du recuit simulé

III.4.4. Paramètres du recuit simulé

La convergence du recuit simulé dépend du contrôle de la « température » du système pour atteindre, le plus vite possible, une solution. Le programme de recuit doit préciser les valeurs des paramètres, de contrôle de la température, suivants :

- la température initiale,
- la longueur de chaîne de Markov, ou le critère de changement de palier de température
- la loi de décroissance de la température,

- le critère d'arrêt du programme ou seuil.

La température *T* a un rôle très important au cours du processus de recuit simulé. En effet, une forte décroissance de température risque de piéger l'algorithme dans un minimum local, alors qu'une faible décroissance au début du processus entraîne une convergence très lente de l'algorithme. La loi, selon laquelle la température décroît, est importante pour l'efficacité de l'algorithme puisqu'elle doit laisser le temps au système de tester le maximum de configurations pour être sûr d'obtenir le minimum global. Aussi, la température initiale doit aussi être suffisamment élevée pour que la descente en température soit aussi lente que possible.

✓ Loi de décroissance de la température

Le changement de température de T_k vers T_{k+1} est effectué au moment où l'équilibre thermique (ou l'état de quasi-équilibre) est détecté. La recherche de cet équilibre s'effectue en générant une succession de chaînes de Markov. La variation de température se fait donc par paliers suivant la fonction de décroissance utilisée. Le tableau III.1 présente les fonctions les plus couramment rencontrées dans la littérature : les fonctions linéaires, discrètes ou exponentielles [Saldanha, 92].

Tableau III.1. Lois de décroissance de la température.

Types	**Fonctions**	**Paramètres**
Linéaires	$T_{K+1} = \alpha T_K$	$0 \prec \alpha \prec 1$
Discrètes	$T_{K+1} = T_K - \Delta T$	$\Delta T \succ 0$
Exponentielles	$T_{K+1} = T_K \exp\left(\frac{-\lambda T_K}{\sigma_K}\right)$ avec $0 \leq \frac{T_{K+1}}{T_K} \leq 1$	σ_K est l'écart type des fonctions objectifs des configurations acceptées à la température T_K ; λ est un paramètre de réglage fixé par l'utilisateur, tel que $\lambda \succ 0$.

III.5. Les approches hybrides proposées

Les deux approches hybrides proposées sont basées sur une coopération relais de haut niveau (HRH). Cette coopération permet de procéder à une phase de diversification pour couvrir une grande partie de l'espace de recherche avant d'arriver à une phase d'intensification pour amé-

liorer les solutions obtenues. Pour la phase de diversification, les AGs sont choisis pour leur pouvoir d'exploration. Pour la phase d'intensification, la RT est choisie pour la première approche hybride et le RS pour la seconde puisque ces deux métaheuristiques sont des méthodes à solution unique.

Par la suite, une présentation des méthodes hybrides ainsi que leurs spécificités réciproques sont données.

III.5.1. Hybridation Séquentielle des Algorithmes Génétiques et de la Recherche Tabou (SH_GA/TS)

La finalité de l'approche proposée est de concevoir une collaboration entre deux métaheuristiques de types différents à savoir les AGs et la RT qui ont prouvé leur efficacité pour la résolution de problèmes d'ordonnancement. Nous envisageons de les faire coopérer pour améliorer leurs résultats. Sont, ensuite, présentés le schéma de coopération ainsi que les spécificités de l'algorithme hybride correspondant proposé.

III.5.1.1. *Schéma de coopération*

L'idée générale du schéma de coopération consiste à utiliser la RT pour exploiter le résultat précédemment trouvé par les AGs par l'exploration de l'espace de recherche [Karray, 11(c)]. Dans le cas d'une hybridation HRH, deux problèmes se posent, le premier est de savoir déterminer préalablement le moment adéquat pour passer de la phase 1, relative à la diversification, à la phase 2 d'intensification de la métaheuristique coopérative et le deuxième consiste en le choix des individus que la RT va explorer. Pour résoudre ces deux problèmes, nous avons opté pour un passage de la phase 1 à la phase 2 se faisant si, après un certain nombre de générations, la fonction fitness reste stable. Les individus pris en compte par la RT sont les meilleurs trouvés.

La résolution séquentielle, schématisée dans la figure III.8, est, ainsi, envisagé en deux étapes [Karray, 11(c)].

Etape 1 : Application des algorithmes génétiques pour la résolution du problème multi-objectifs. Dans cette étape, la population initiale est générée aléatoirement. Les opérateurs de sélection, de croisement et de mutation constituent le processus de reproduction, pour créer de nouvelles populations, qui est répété jusqu'à l'obtention d'une population finale qui s'approche au mieux de la solution optimale.

Etape 2 : Implémentation de la RT après les AGs : le meilleur individu de la population finale, obtenue dans l'étape 1, est choisi comme configuration initiale de la RT. Un ensemble de voisinages est généré à partir de la configuration courante, afin de générer de nouvelles solutions. La liste tabou est mise à jour et le critère d'aspiration vérifié. Ce processus est répété jusqu'à la validation du critère d'arrêt.

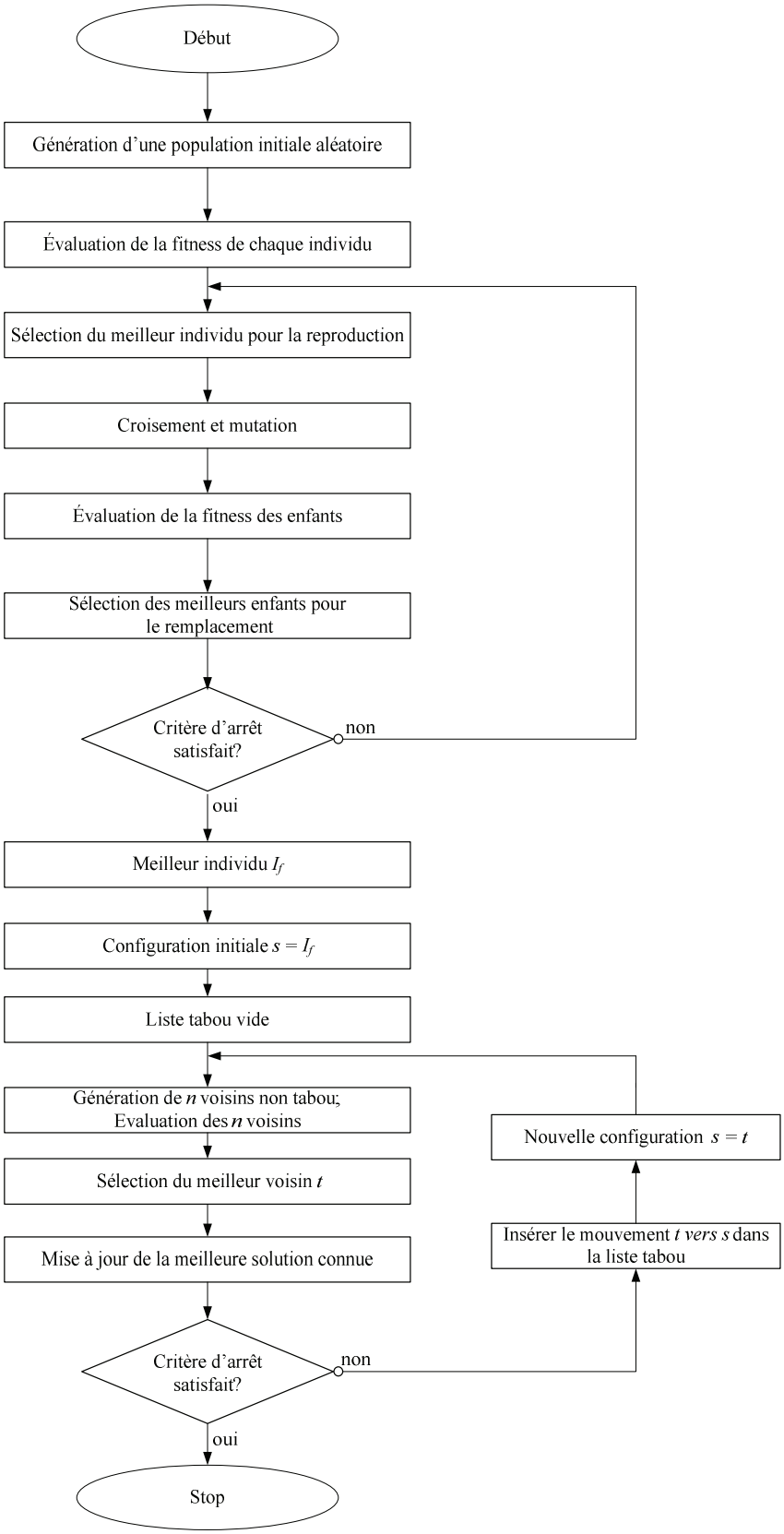

Figure III.8. Organigramme de l'algorithme SH_GA/TS

III.5.1.2. ***Spécificités de l'algorithme hybride SH_GA/TS***

Les spécificités de l'approche hybride sont :

- ✓ *le codage proposé* est le codage CLOA qui définit l'ordre, la date de début au plus tôt, la durée et la date de fin effective des opérations,
- ✓ *la génération de la population initiale* est la même que celle prise en compte dans le chapitre II, la population initiale étant, en effet, générée aléatoirement suivant l'algorithme de la figure II.15,
- ✓ *l'opérateur de sélection* choisi est la sélection par tournoi qui consiste à choisir aléatoirement deux ou plusieurs chromosomes au hasard et permettre à celui qui a une meilleure évaluation d'être sélectionné ; l'avantage de cette sélection par rapport à la sélection par la roulette étant d'éviter qu'un individu très fort soit sélectionné plusieurs fois,
- ✓ *l'opérateur de croisement* est le même que celui pris en compte dans le chapitre II, à savoir le croisement bi-points, l'algorithme de cet opérateur de croisement étant celui de la figure II.16,
- ✓ *l'opérateur de mutation* est le même que celui pris en compte dans le chapitre II, à savoir la mutation bi-points, l'algorithme de cet opérateur de mutation étant celui de la figure II.18.
- ✓ *la génération des voisins* utilise la méthode de permutation de deux éléments quelconques distincts et permet de générer $\left(\frac{n(n-1)}{2}\right)$ voisins ; la figure III.9 montre un exemple de génération de voisins.

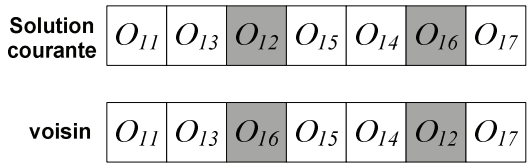

Figure III.9. Exemple de génération de voisins

- ✓ *la liste tabou* suit la stratégie FIFO (First In First Out) qui signifie que l'ordre de sortie des éléments de la liste correspond à celui de leur insertion. La taille de la liste est statique et la solution qu'on s'interdit de visiter pendant un certain nombre d'itérations est sauvegardée,

- ✓ *le critère d'aspiration* : après un certain nombre d'itérations, une solution déjà visitée n'est plus tabou,
- ✓ *le critère d'arrêt* : après un certain nombre d'itérations et si la valeur de la fonction objectif ne change plus, l'algorithme s'arrête.

III.5.2. Hybridation Séquentielle des Algorithmes Génétiques et du Recuit Simulé (SH_GA/SA)

A l'instar de la RT, le RS est une métaheuristique à parcours. Il est paraît intéressant d'utiliser cette métaheuristique dans la phase d'intensification dans le cadre d'une hybridation HTH avec les AGs et de comparer les résultats obtenus par cette hybridation avec ceux relatifs à l'hybridation précédente, à savoir SH_GA/TS. Par la suite, est présenté le schéma de coopération ainsi que les spécificités de l'algorithme hybride proposé.

III.5.2.1. *Schéma de coopération*

L'algorithme hybride proposé, consigné dans la figure III.10, commence par une phase d'exploration de l'espace de recherche grâce aux AGs. Après un certain nombre de générations et lorsque la fonction fitness est stable, le meilleur individu trouvé est choisi comme solution initiale de la phase d'exploitation. Ainsi, la coopération séquentielle se fait en deux étapes.

Etape 1 : Application des algorithmes génétiques pour la résolution du problème multi-objectifs. Dans cette étape, la population initiale est générée aléatoirement. Les opérateurs de sélection, de croisement et de mutation constituent le processus de reproduction pour créer de nouvelles populations, répété jusqu'à l'obtention d'une population finale qui s'approche au mieux de la solution optimale.

Etape 2 : Implémentation du RS à la suite des AGs : le meilleur individu de la population finale, obtenue dans l'étape 1, est choisi comme configuration initiale du RS. Une modification élémentaire de la configuration courante est faite. Si cette variation diminue la fonction *f*, elle est acceptée sinon la règle de Metropolis est appliquée. Vérifier si l'équilibre thermique est atteint. Si oui, faire une diminution lente de *T* et reprendre les modifications élémentaires de la configuration courante. Ce processus est répété jusqu'à ce qu'un seuil prédéfini est atteint.

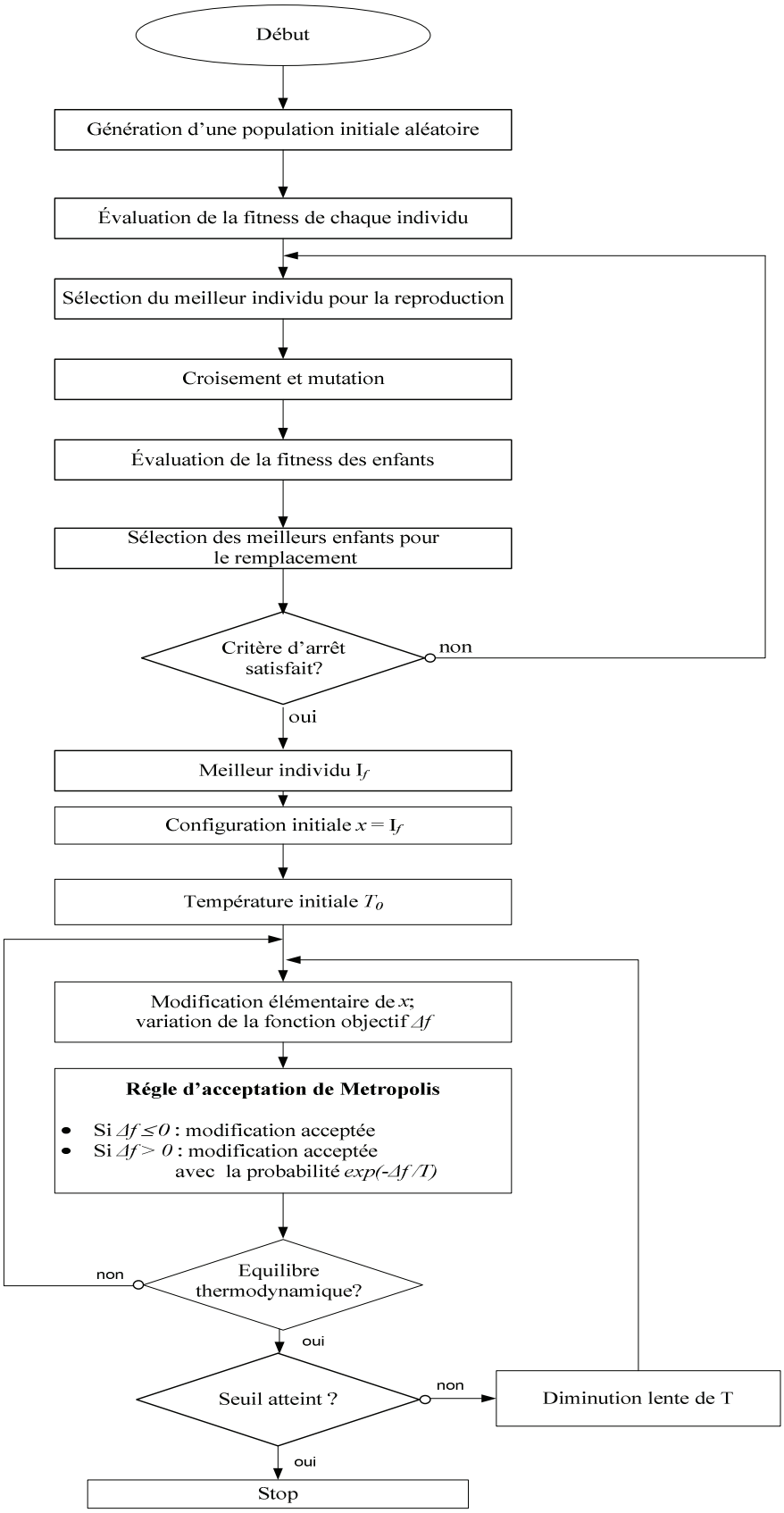

Figure III.10. Organigramme de l'algorithme SH_GA/SA

III.5.2.2. ***Spécificités de l'algorithme hybride SH_GA/SA***

Les spécificités de l'approche hybride sont :

- ✓ *le codage proposé* : le codage CLOA ; ce codage définit l'ordre, la date de début au plus tôt, la durée et la date de fin effective des opérations,
- ✓ *la génération de la population initiale* est la même que celle prise en compte dans le chapitre II, la population initiale étant générée aléatoirement suivant l'algorithme de la figure II.15,
- ✓ *l'opérateur de sélection* choisi est la sélection par tournoi qui consiste à choisir aléatoirement deux ou plusieurs chromosomes au hasard et à permettre à celui qui a la meilleure évaluation, d'être sélectionné. L'avantage de cette sélection par rapport à la sélection par la roulette est d'éviter qu'un individu très fort soit sélectionné plusieurs fois,
- ✓ *l'opérateur de croisement* est le même que celui pris en compte dans le chapitre II, à savoir le croisement bi-points, l'algorithme de cet opérateur de croisement étant celui de la figure II.16.
- ✓ *l'opérateur de mutation* est le même que celui pris en compte dans le chapitre II, à savoir la mutation bi-points, l'algorithme de cet opérateur de mutation étant celui de la figure II.18.
- ✓ *la température initiale* T_0 est calculée au préalable à partir de l'algorithme suivant [Dréo, 03] :
  - faire 100 perturbations au hasard ; évaluer la moyenne $< \Delta f >$ des variations Δf correspondantes,
  - choisir un taux initial τ d'acceptation des perturbations dégradantes, selon la qualité supposée de la configuration initiale, pour une température initiale "moyenne", la valeur de τ est de 0.5,
  - déduire T_0 de la relation : $\tau = \exp\left(\frac{-|< \Delta f >|}{T_0}\right)$,
- ✓ *le critère de changement de palier de température* est pris en compte après un certain nombre d'itérations,
- ✓ *la loi de décroissance de la température* est la loi de décroissance linéaire : $T_{K+1} = \alpha . T_K$, où α étant une constante, comprise entre 0 et 1, choisi préalablement,

- ✓ *le critère d'arrêt du programme ou seuil* peut être opéré, après un certain nombre de paliers de température successifs, sans aucun changement du résultat.

III.6. Résultats de simulation d'hybridation de métaheuristiques

III.6.1. Benchmarks étudiés

Pour illustrer l'efficacité et les performances des approches d'optimisation hybrides proposées, deux exemples d'ateliers à une machine et plusieurs opérations, caractérisés par des données concrètes, sont considérés. Les approches proposées sont ainsi appliquées pour optimiser les trois critères cités dans le chapitre II. Les données relatives aux exemples étudiés sont consignées dans les tableaux II.4 et II.5 du chapitre II.

III.6.2. Résultats relatifs à l'approche SH_GA/TS

L'algorithme SH_GA/TS proposé, utilisant le codage CLOA est appliqué en vue d'obtenir l'ordre de passage des opérations minimisant les critères *C1*, *C2* et *C3*.

Le tableau III.2 résume les valeurs des paramètres utilisés par l'approche SH_GA/TS, pour les deux benchmarks, afin de générer de bonnes solutions. Le tableau III.3 présente les résultats obtenus par application de l'algorithme SH_GA/TS proposé pour traiter le benchmark 1 et le benchmark 2.

Tableau III.2. Valeurs des paramètres relatifs à l'algorithme SH_GA/TS

	Benchmark 1	Benchmark 2
Taille de la population	10	30
Nombre d'itérations	100	200
Probabilité de croisement	0.7	0.7
Probabilité de mutation	0.01	0.01
Taille du voisinage	45	60
Taille de la liste tabou	300	500
Critère d'aspiration	20	30
Critère d'arrêt	200	300

La figure III.11 présente l'évolution des coûts pour le benchmark 1 et la meilleure configuration pour laquelle le minimum global est obtenu. De même, la figure III.12 donne l'évolution des coûts pour le benchmark 2.

Tableau III.3. Résultats de simulation obtenus par l'algorithme SH_GA/TS

	Benchmark 1	Benchmark 2
F	18	49
Convergence	130	273
CPU	3.841	31.017

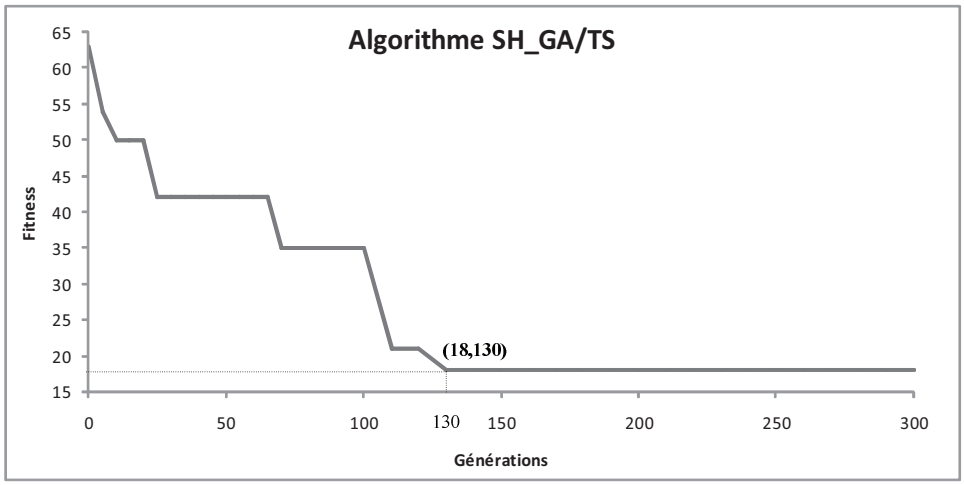

Figure III.11. Evolution du coût pour le benchmark 1 par application de l'algorithme SH_GA/TS

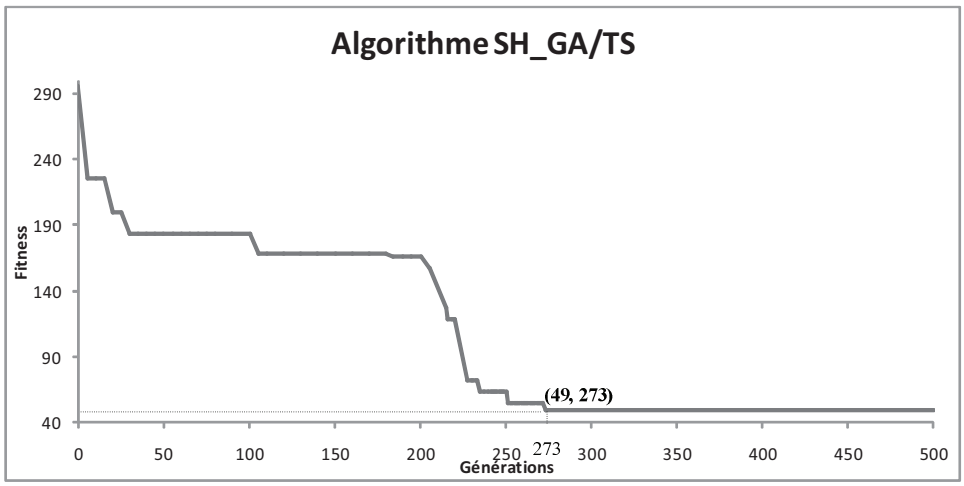

Figure III.12. Evolution du coût pour le benchmark 2 par application de l'algorithme SH_GA/TS

Les figures III.11 et III.12, montrent que la convergence de l'algorithme mis en œuvre, pour le benchmark 1, est obtenue à la 130ème itération pour un minimum global égal à 18. Pour le benchmark 2, la convergence est obtenue à la 273ème itération pour un minimum global égal à 49.

III.6.3. Résultats relatifs à l'approche SH_GA/SA

Dans le tableau III.4, sont consignées les valeurs des paramètres utilisés par l'approche SH_GA/SA, pour les benchmarks 1 et 2, afin de générer les meilleures solutions. Ces valeurs sont relatives respectivement aux paramètres des AGs et aux paramètres du RS. Le tableau III.5 présente les résultats obtenus par application de l'algorithme SH_GA/SA, proposé pour traiter le benchmark 1 et le benchmark 2.

Tableau III.4. Valeurs des paramètres relatifs à l'algorithme SH_GA/SA

	Benchmark 1	Benchmark 2
Taille de la population	10	30
Nombre d'itérations	100	200
Probabilité de croisement	0.7	0.7
Probabilité de mutation	0.01	0.01
Température initiale	75	96
Itérations pour changement de la température	10	20
Facteur de décroissance de la température, α	0.9	0.9
Seuil	0.25	0.19

Les figures III.13 et III.14 présentent les évolutions des coûts pour le benchmark 1 et le benchmark 2 respectivement. Elles présentent ainsi la meilleure configuration pour lequel le minimum global est obtenu pour chaque benchmark.

Tableau III.5. Résultats de simulation obtenus par l'algorithme SH_GA/SA

	Benchmark 1	Benchmark 2
F	17	49
Convergence	118	261
CPU	2.374	22.022

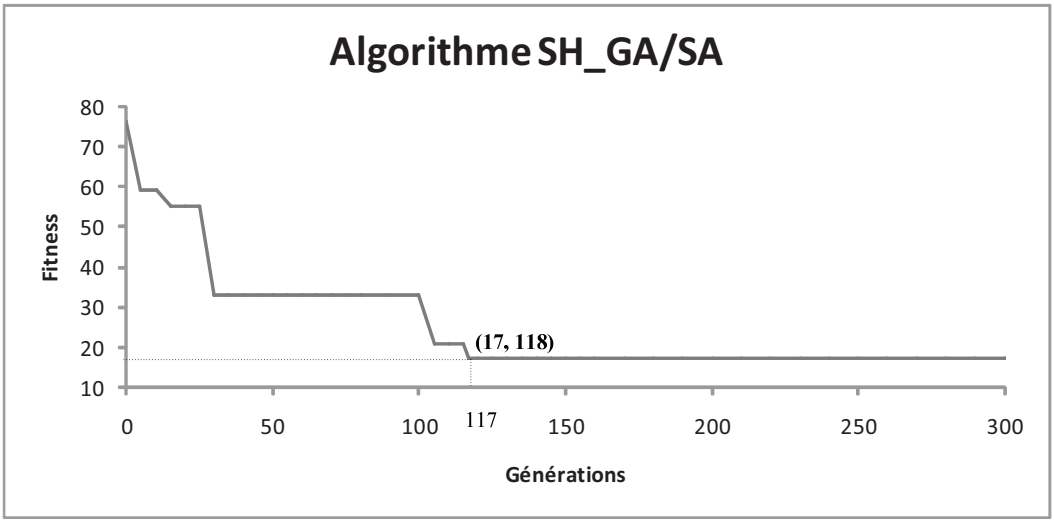

Figure III.13. Evolution du coût pour le benchmark 1 par application de l'algorithme SH_GA/SA

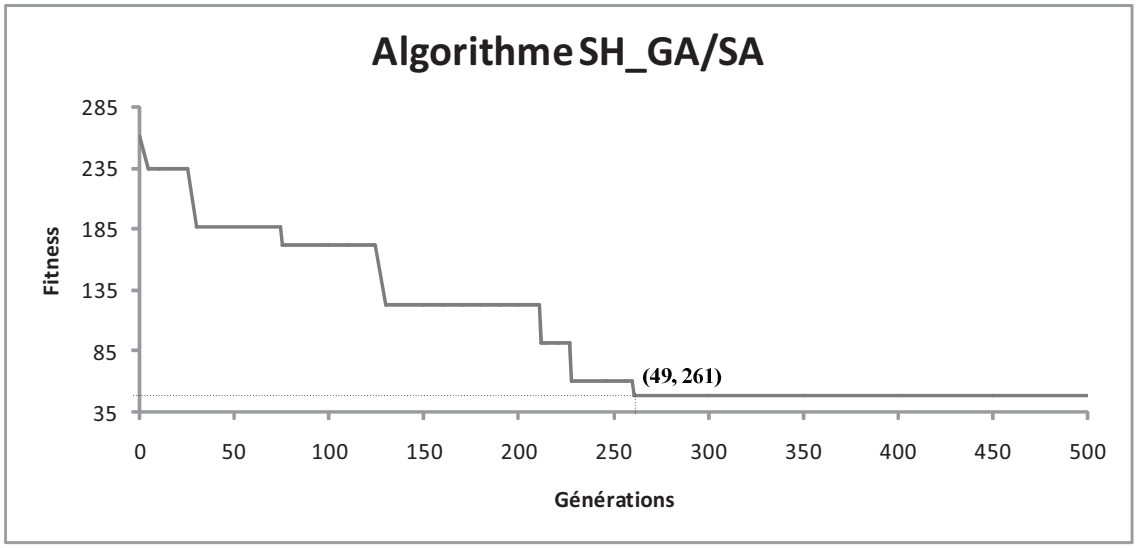

Figure III.14. Evolution du coût pour le benchmark 2 par application de l'algorithme SH_GA/SA

Les figures III.13 et III.14, montrent que la convergence de l'algorithme SH_GA/SA, pour le benchmark 1, est obtenue à la 118ème itération pour un minimum global égal à 17. Pour le benchmark 2, la convergence est obtenue à la 261ème itération pour un minimum global égal à 49.

III.7. Etude comparative des performances des méthodes hybrides mises en œuvre

Dans cette partie, une étude comparative des performances de chaque méthode hybride proposée par rapport à des méthodes existantes pour la résolution de problèmes multi-objectifs est présentée. La comparaison se base non seulement sur la qualité des résultats obtenus mais aussi sur la vitesse de convergence et sur la vitesse de compilation.

III.7.1. Comparaison des performances des méthodes hybrides proposées avec les méthodes existantes

Une comparaison des performances de l'approche SH_GA/TS par rapport aux AGs et à la RT est d'abord considérée. Par la suite, est proposée une comparaison de la 2ème approche hybride, à savoir SH_GA/SA, avec les AGs et le RS.

III.7.1.1. *Comparaison des performances de l'approche SH_GA/TS mise en œuvre avec les AGs et la RT*

III.7.1.1.1. Cas du benchmark 1

Dans les tableaux III.6 et III.7 sont consignés les paramètres utilisés par les AGs respectivement par la RT pour le benchmark 1, et les figures III.15 et III.16 les évolutions des coûts à travers les générations des AGs respectivement de la RT pour le benchmark 1.

Tableau III.6. Valeurs des paramètres relatifs aux AGs pour le benchmark 1

Paramètres	Valeurs
Taille de la population	10
Nombre d'itérations	300
Probabilité de croisement	0.7
Probabilité de mutation	0.01

Tableau III.7. Valeurs des paramètres relatifs à la RT pour le benchmark 1

Paramètres	Valeurs
Taille du voisinage	45
Taille de la liste tabou	300
Critère d'aspiration	50

Critère d'arrêt	500

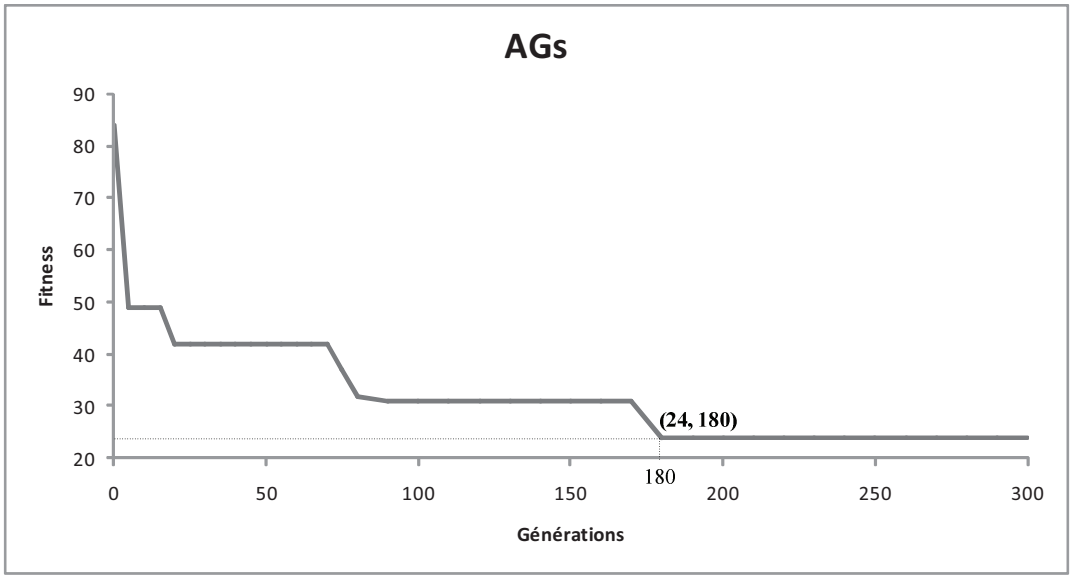

Figure III.15. Evolution du coût pour le benchmark 1 par application des AGS

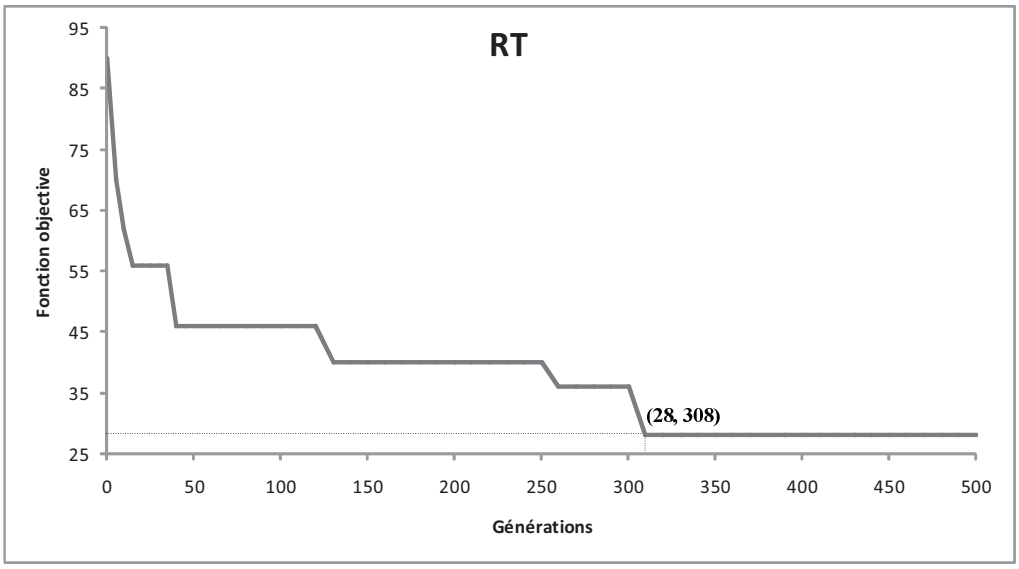

Figure III.16. Evolution du coût pour le benchmark 1 par application de l'algorithme de la RT

A partir de la figure III.15, on note que la convergence des algorithmes AGs, pour le benchmark 1, est obtenue à la 180ème itération pour un minimum global égal à 24 et à partir de la figure III.16, que la meilleure configuration, donnant l'optimum global égal à 28, est atteinte après 308 itérations.

Le tableau III.8 présente les résultats obtenus par application des différentes méthodes utilisées pour traiter le benchmark 1.

Tableau III.8. Résultats de simulation obtenus par les approches AGs, RT et SH_GA/TS pour le benchmark 1

Méthode d'optimisation	AGs	RT	SH_GA/TS
F	24	28	18
Convergence	180	308	130
CPU (s)	8.669	3.106	3.841

A partir du tableau III.8, on note que la méthode développée SH_GA/TS a donné un meilleur résultat pour le benchmark 1, puisque la valeur de la fonction F trouvée est inférieure à celles trouvées par les AGs et la RT appliqués seuls. Aussi, on note que la convergence de l'approche hybride est plus rapide que les deux autres méthodes puisque l'algorithme SH_GA/TS converge à la 130ème génération, donc 40 itérations avant l'AGs et 178 itérations avant la RT.

Toutefois, la RT s'avère de point de vue temps de compilation plus rapide que les AGs et l'approche SH_GA/TS.

III.7.1.1.2. Cas du benchmark 2

Les tableaux III.9 et III.10 donnent les paramètres utilisés par les AGs respectivement par la RT pour le benchmark 2 et les figures III.17 et III.18 les évolutions des coûts à travers les générations des AGs respectivement de la RT pour le benchmark 2.

Tableau III.9. Valeurs des paramètres relatifs aux AGs pour le benchmark 2

Paramètres	Valeurs
Taille de la population	30
Nombre d'itérations	600
Probabilité de croisement	0.7

Probabilité de mutation	0.01

Tableau III.10. Valeurs des paramètres relatifs à la RT pour le benchmark 2

Paramètres	Valeurs
Taille du voisinage	60
Taille de la liste tabou	500
Critère d'aspiration	30
Critère d'arrêt	300

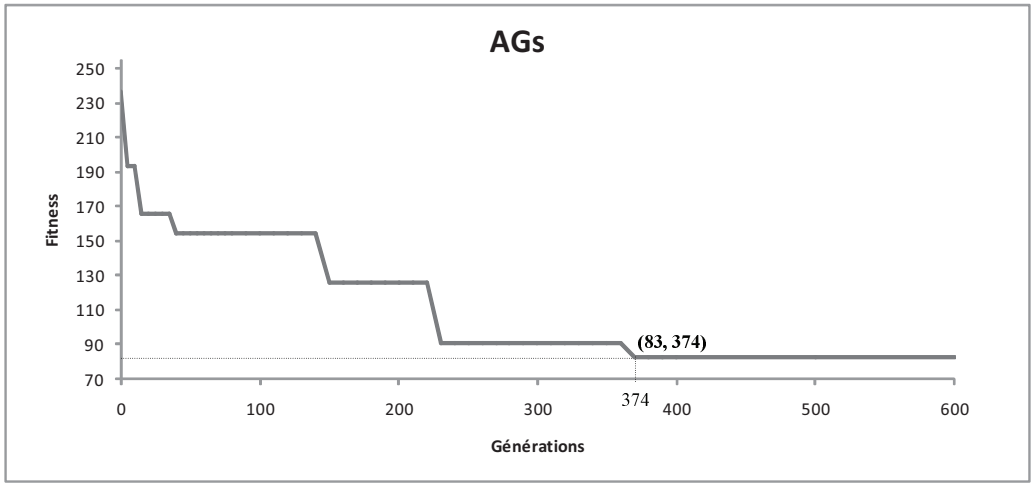

Figure III.17. Evolution du coût pour le benchmark 2 par application des AGS

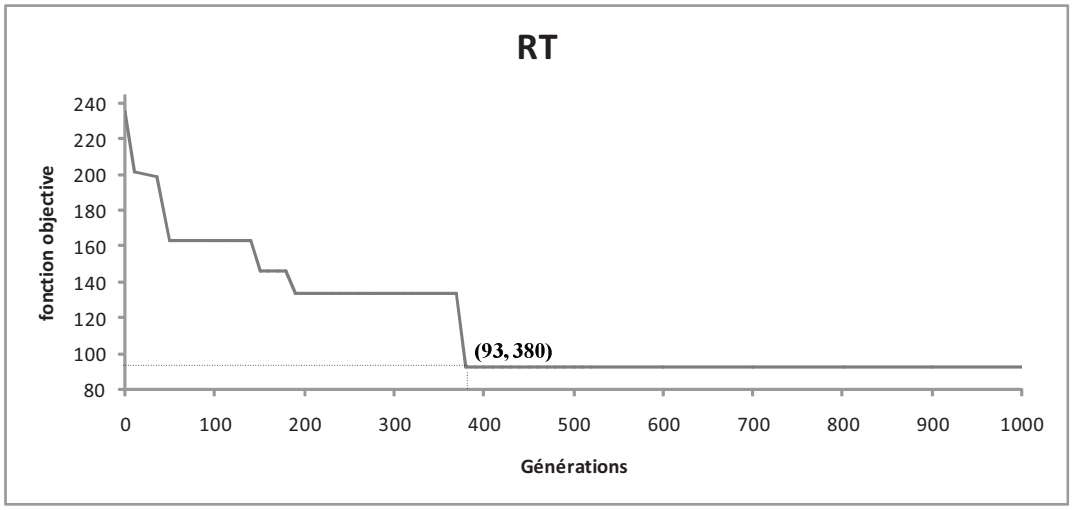

Figure III.18. Evolution du coût pour le benchmark 2 par application de l'algorithme de la RT

A partir de la figure III.17, on note que la convergence des AGs, pour le benchmark 2, est obtenue à la 374ème itération pour un minimum global égal à 83 et à partir de la figure III.18,

que la meilleure configuration donnant l'optimum global, égal à 93, est atteinte après 380 itérations.

Le tableau III.11 présente les résultats obtenus par application des différentes méthodes utilisées pour traiter le benchmark 2.

Tableau III.11. Résultats de simulation obtenus par les approches AGs, RT et SH_GA/TS pour le benchmark 2

Méthode d'optimisation	AGs	RT	SH_GA/TS
F	83	93	49
Convergence	374	380	273
CPU (s)	68,356	36,251	31,017

A partir du tableau III.11, on note que la méthode développée SH_GA/TS a donné un meilleur résultat pour le benchmark 2, puisque la valeur de la fonction F trouvée est inférieure à celles trouvées par les AGs et la RT appliqués seuls. Aussi, on note que la convergence de l'approche hybride est plus rapide que les deux autres méthodes puisque l'algorithme SH_GA/TS converge à la 273ème génération alors que les AGs convergent à la 374ème itération et la RT converge à la 380ème itération.

De point de vue temps de compilation, on note que l'approche hybride développée est plus rapide que les AGs et la RT. En effet, la différence du temps de compilation, pour le benchmark 2, entre l'approche SH_GA/TS et les AGs est approximativement de 37 secondes et entre l'approche SH_GA/TS et la RT approximativement de 5 secondes.

A partir des résultats de simulation des deux benchmarks, on constate l'avantage que l'hybridation de type HTH a apporté aux AGs de point de vue temps de compilation et vitesse de convergence.

Finalement, on note que la coopération co-évolutionnaire de haut niveau entre l'algorithme génétique et la recherche tabou a permis de trouver de meilleurs résultats que les deux méthodes autonomes.

III.7.1.2. *Comparaison des performances de l'approche SH_GA/SA mise en œuvre avec les AGs et la RS*

III.7.1.2.1. Cas du benchmark 1

Les tableaux III.6 et III.12 donnent les paramètres utilisés par les AGs et respectivement le RS pour le benchmark 1 et les figures III.15 et III.19 les évolutions des coûts à travers les générations des AGs respectivement du RS pour le benchmark 1.

Tableau III.12. Valeurs des paramètres relatifs au RS pour le benchmark 1

Paramètres	Valeurs
Température initiale	80
Itérations pour changement de la température	20
Facteur de décroissance de la température, α	0.9
Seuil	0.16

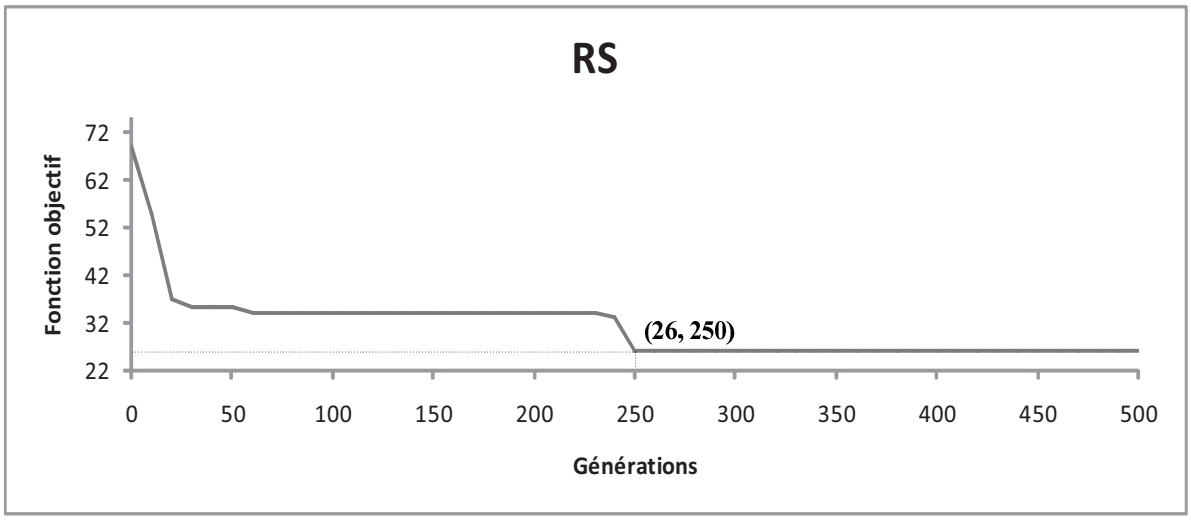

Figure III.19. Evolution du coût pour le benchmark 1 par application de l'algorithme du RS

D'après la figure III.15, on note que la convergence des algorithmes AGs, pour le benchmark 1, est obtenue à la 180ème itération pour un minimum global égal à 24. On note aussi, à partir de la figure III.19, que la meilleure configuration donnant l'optimum global égal à 26, est atteinte après 250 itérations.

Le tableau III.13 présente les résultats obtenus par application des différentes méthodes utilisées pour traiter le benchmark 1.

Tableau III.13. Résultats de simulation obtenus par les approches AGs, RS et SH_GA/SA pour le benchmark 1

Méthode d'optimisation	AGs	RS	SH_GA/SA
F	24	26	17
Convergence	180	250	118
CPU (s)	8.669	2.116	2.374

A partir du tableau III.13, on note que la méthode développée SH_GA/SA a donné un meilleur résultat pour le benchmark 1, puisque la valeur de la fonction F trouvée est inférieure à celles trouvées par AGs et RS appliqués seuls.

De point vue convergence de la fonction objectif pour les trois différentes approches, on note que l'algorithme SH_GA/SA converge plus rapidement que les AGs et le RS puisque ce dernier converge à la 118ème itération alors que les AGs convergent après 180 itérations et le RS converge après 250 itérations.

Aussi, on note que du point de vue temps de compilation, l'approche SH_GA/TS est plus rapide que les AGs. En effet, la différence du temps de compilation, pour le benchmark 1, entre l'approche SH_GA/TS et les AGs est approximativement de 6.3 seconde. Toutefois, on note que la vitesse de compilation de l'approche hybride et le RS est presque la même puisque la différence du temps de compliation est inférieure à 0.5 seconde.

III.7.1.2.2. Cas du benchmark 2

Les tableaux III.9 et III.14 présentent les paramètres utilisés par les AGs respectivement le RS pour le benchmark 2 et les figures III.17 et III.20 les évolutions des coûts à travers les générations des AGs respectivement du RS pour le benchmark 2.

Tableau III.14. Valeurs des paramètres relatifs au RS pour le benchmark 2

Paramètres	Valeurs
Température initiale	90
Itérations pour le changement de la température	30
Facteur de décroissance de la température, α	0.9
Seuil	0.06

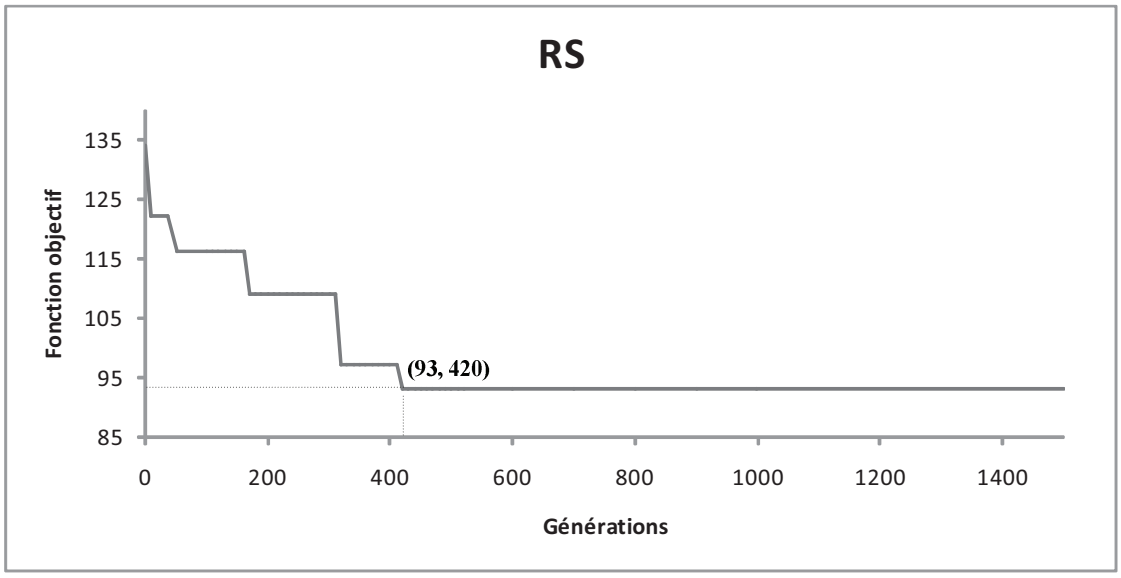

Figure III.20. Evolution du coût pour le benchmark 2 par application de l'algorithme du RS

A partir de la figure III.20, on note que la convergence de l'algorithme RS, pour le benchmark 2, est obtenue à la 420ème itération pour un minimum global égal à 93 et à partir de la figure III.17, que le meilleur individu donnant l'optimum global égal à 83, est atteint après 374 itérations.

Le tableau III.15 présente les résultats obtenus par application des différentes méthodes utilisées pour traiter le benchmark 2.

Tableau III.15. Résultats de simulation obtenus par les approches AGs, RS et SH_GA/SA pour le benchmark 2

Méthode d'optimisation	AGs	RS	SH_GA/SA
F	83	93	49
Convergence	374	420	261
CPU (s)	68,3566	28,210	22,022

A partir du tableau III.15, on note que la méthode développée SH_GA/SA a donné un meilleur résultat pour le benchmark 2, puisque la valeur de la fonction F trouvée est inférieure à celles trouvées par les AGs et la RS appliqués seuls.

Aussi, à partir du tableau III.15, on note que l'algorithme SH_GA/SA converge à partir de la 261ème itération alors que les AGs convergent après 374 itérations et le RS après 420 itéra-

tions. Ainsi, l'approche hybride SH_GA/SA converge plus rapidement que les deux autres approches.

De point de vue temps de compilation, on note que l'approche hybride développée est plus rapide que les AGs et la RS. En effet, la différence du temps de compilation, pour le benchmark 2, entre l'approche SH_GA/TS et les AGs est approximativement de 46 secondes et entre l'approche SH_GA/TS et la RT est approximativement de 6 secondes.

A partir des résultats de simulation des deux benchmarks, on constate l'avantage que l'hybridation de type HTH a apporté aux AGs de point de vue temps de compilation.

Finalement, on note que la coopération co-évolutionnaire de haut niveau, entre l'algorithme génétique et le recuit simulé, a permis de trouver de meilleurs résultats que les deux méthodes appliquées seules.

III.7.2. Comparaison des performances des approches SH_GA/TS et SH_GA/SA

Dans cette partie, nous allons comparer les performances des méthodes hybrides proposées. La comparaison se base non seulement sur la qualité des résultats obtenus mais aussi sur la vitesse de convergence et la vitesse de compilation.

Un bilan des résultats trouvés par les deux approches hybrides, pour les deux benchmarks, est donné dans le tableau III.16.

Tableau III.16. Résultats de simulation relatifs aux deux approches hybrides SH_GA/TS et SH_GA/SA

	Benchmark 1		Benchmark 2	
	SH_GA/TS	SH_GA/SA	SH_GA/TS	SH_GA/SA
F	18	17	49	49
Convergence	130	118	273	261
CPU	3.841	2.374	31.017	22.022

A partir du tableau III.16, on constate que, pour le benchmark 1, l'approche SH_GA/SA a conduit au meilleur résultat avec un temps de compilation plus rapide que l'approche SH_GA/TS. En effet, la différence du temps de compilation est de 1.5 seconde. De même, pour le benchmark 2, on constate aussi que l'approche SH_GA/SA a abouti au meilleur résultat avec un temps de compilation plus rapide que l'approche SH_GA/TS puisque la différence du temps de compilation entre les deux approches est approximativement de 9 secondes.

D'après le tableau III.16, on note que la convergence de SH_GA/SA a lieu à la 118ème itération pour le benchmark 1, alors que la convergence de SH_GA/TS a lieu à la 130ème itération. Aussi pour le benchmark 2, la convergence de SH_GA/SA a lieu à la 261ème itération pour le benchmark 1, alors que la convergence de SH_GA/TS a lieu à la 273ème itération.

L'approche SH_GA/SA a donc donné de meilleurs résultats avec une meilleure convergence et un meilleur temps de calcul pour les deux benchmarks par rapports à l'approche SH-GA/TS.

III.8. Conclusion

Ce chapitre a permis l'étude de l'hybridation des métaheuristiques dans le cadre de l'optimisation multi-objectifs. L'intérêt de ce type d'approche coopérative a été exposé par rapport aux métaheuristiques. Dans ce chapitre, deux approches hybrides de type haut ont été proposées entre différents types de métaheuristiques. Cette coopération consiste à combiner un ensemble de métaheuristiques.

La résolution d'un problème d'ordonnancement relatif à un atelier agroalimentaire avec les méthodes hybrides développées et la comparaison de ces résultats avec ceux des métaheuristiques ont montré l'efficacité des approches que nous avons développées de point de vue qualité de la solution, vitesse de convergence et vitesse de compilation.

Cependant, la différence de performances entre les approches SH_GA/TS et SH_GA/SA s'est avérée presque négligeable. Ceci peut s'expliquer par le fait que la RT et le RS sont deux métaheuristiques de parcours.

CONCLUSION GENERALE

La résolution de problèmes d'ordonnancement d'ateliers à une machine en industries agro-alimentaires constitue la principale contribution de nos travaux consignés dans ce mémoire. L'objectif de notre travail est de trouver un ordonnancement dynamique adapté à ce type d'industrie qui traite des produits fabriqués et manipulés qui se caractérisent par des courtes durées de vie et par la spécificité des critères à retenir. Il s'agit d'une part, de la minimisation des produits périmés à savoir les composants primaires, les produits semi-finis et les produits finis et de la minimisation du discount de distribution, tout en respectant la date de livraison ou au moins la réduction de la différence entre la date effective de fin de fabrication du produit fini et sa date de livraison et, d'autre part, la prise en compte de la date de fin d'ordonnancement, le C_{max}.

Dans ce mémoire, nous nous sommes, tout d'abord, intéressés à l'étude de l'ordonnancement dans sa globalité, avec ses différentes composantes (tâches, ressources, contraintes, critères), en rapport avec les différents types d'ateliers étudiés et avec les méthodes d'optimisation pour trouver de bonnes solutions aux problèmes correspondants en utilisant pour leur résolution, des méthodes exactes (programmation linéaire, branch & bound) et des méthodes approchées (recuit simulé, recherche tabou, algorithmes génétiques et algorithmes de colonies de fourmis).

Trois nouvelles approches, basées sur les algorithmes génétiques, ont été, ensuite, proposées pour la résolution de problèmes d'ordonnancement multi-objectifs en industries agroalimentaires : les algorithmes génétiques séquentiels (SGA_1), les algorithmes génétiques parallèles (PGA_1) et les algorithmes génétiques parallèles séquentiels (PSGA_1). Celles-ci ont été appliquées avec succès à la résolution deux problèmes d'ordonnancement en industries agroalimentaires pour différents critères.

La comparaison des méthodes de résolution de ces deux problèmes, ainsi formulés, et la comparaison des approches basées sur les algorithmes génétiques par rapport à l'approche Pareto-optimalité et l'approche basée sur la pondération des fonctions, a montré l'efficacité des ap-

proches développées du point de vu qualité de la solution et temps de compilation. Une étude comparative des performances de celle-ci a montré leurs avantages et leurs inconvénients.

L'attention a été, enfin, focalisée sur l'étude de l'hybridation des métaheuristiques. Nous avons tout d'abord présenté la taxinomie de coopération entre métaheuristiques introduites dans [Talbi, 02], et les métaheuristiques utilisées pour l'hybridation dont la recherche tabou et le recuit simulé.

Deux approches coopératives multiobjectif en mode relais, SH_GA/TS et SH_GA/SA, hybridant toutes les deux des métaheuristiques de haut niveau, ont été par la suite, proposées. Un algorithme évolutionnaire et un algorithme de recherche locale sont, dans ce cas exécutés séquentiellement.

Dans la formulation générale de ces deux méthodes hybrides, introduite, notre contribution a été la proposition d'un schéma de coopération pouvant être utilisable, d'une manière générale, dans les problèmes d'ordonnancement.

Au cours de nos expérimentations, l'hybridation proposée, testée sur un problème d'ordonnancement en industries agroalimentaires, a permis une amélioration par rapport aux algorithmes classiques multi-objectifs, sur trois plans : la vitesse de convergence, la vitesse de compilation et la qualité des résultats obtenus.

L'étude de l'efficacité de ces deux méthodes hybrides a montré un écart faible des performances entre les approches SH_GA/TS et SH_GA/SA pouvant s'expliquer par le fait que la RT et le RS sont deux métaheuristiques de parcours.

Toutes ces conclusions, confirment que la coopération des métaheuristiques a fait ses preuves, autant que les métaheuristiques, pour la résolution de problèmes d'ordonnancement.

Les résultats obtenus étant encourageants, il serait intéressant de pouvoir développer certains autres aspects envisageables en perspectives :

- généraliser et étendre les approches proposées à d'autres industries qui présentent les mêmes types de contraintes telles que les industries chimiques ou pharmaceutiques,
- appliquer les approches développées pour d'autres cas d'ateliers comme le job shop ou le flow shop,

- intégrer le cas de la maintenance permettant ainsi la résolution conjointe de la planification de la production et de la maintenance,
- généraliser nos travaux aux hybridations parallèle synchrone et asynchrone ; et c'est dans ce sens que nous envisageons de poursuivre nos travaux de recherche.

BIBLIOGRAPHIE

[Alba, 02a] E. Alba, A. J. Nebro, J. M. Troya, Heterogeneous computing and parallel genetic algorithms. Journal of Parallel Distributed Computing, vol. 65, pp. 1362–1385, 2002.

[Alba, 02b] E. Alba, M. Tomassini, Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computing, vol. 6 (5), pp: 443–462, 2002.

[Altay, 95] H. Altay, A genetic algorithm for multicriteria inventory classification. Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, pp. 6-9, Avril 1995.

[Andresen, 08] M. Andresen, H. Bräsel, M. Mörig, J. Tusch, F. Werner and P. Willenius, Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. Mathematical and Computer Modelling, vol. 48, pp. 1279–1293, 2008.

[Anghinolfi, 09] D. Anghinolfi, M. Paolucci, A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. European Journal of Operational Research, vol. 193, pp. 73–85, 2009.

[Armentano, 00] V. A. Armentano, R. Mazzini, A genetic algorithm for scheduling on a single machine set-up times and due dates. Production Planning & Control, vol. 11, pp. 713–720, 2002.

[Baptiste, 96] P. Baptiste, C. Le Pape, A constraint-Based Branch and Bound Algorithm for Preemptive Job-Shop Scheduling. 5th IEEE, International Symposium on Assembly and Task Planning, Besançon, 1996.

[Bel, 01] G. Bel, J. B. Cavaillé, Ordonnancement de la production. Editions Hermès, Paris, 2001.

[Bellman, 86] R. E. Bellman, The Bellman continum. Editions Robert S. Roth, Philadelphia (USA), 1986.

[Blazewicz, 96] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, J. Weglarz, Scheduling computer and manufacturing process. Springer, Berlin, 1996.

[Borne, 07] P. Borne, F. Tangour, Metaheuristics for the optimization in planning and scheduling. Conférence plénière, MCPL2007 IFAC conférence, Proc. T1, pp. 1-8, Sibiu (Romania), Septembre 2007.

[Borne, 10] P. Borne, M. Benrejeb, La nature source d'inspiration pour l'ingénieur. L'ingénieur, N° 260, pp. 12-15, janvier-février 2010.

[Boukef, 09] H. Boukef, M. Benrejeb, P. Borne, Genetic Algorithm and Particle Swarm Optimization comparison for solving a flow-shop multiobjective scheduling problem. International Review of Automatic Control (IREACO), Vol. 2, N°2, pp. 223-228, 2009.

[Brown, 89] D. Brown, C. Huntley, and A. Spillane, A parallel genetic heuristic for the quadratic assignment problem. In 3rd International Conference on Genetic Algorithms, pp. 406- 415, 1989.

[Brucker, 98] P. Brucker, Scheduling algorithms. Springer-Verlag, Berlin, 1998.

[Brucker, 94] P. Brucker, B. Jurisch, B. Sievers, A branch & bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics, vol. 49, pp. 107–127, 1994.

[Cantú-Paz, 98] E. Cantú-Paz, A survey of parallel genetic algorithms. Calculateurs Parallèles, Réseaux et Systems Repartis, vol. 10 (2), pp. 141–171, Hermès, Paris, 1998.

[Cantú-Paz, 00a] E. Cantú-Paz, D.E. Goldberg, Parallel genetic algorithms: theory and practice. Computing Methods Applications on Mechanics and Engineering, vol. 186, pp. 221–238, 2000.

[Cantú-Paz, 00b] E. Cantú-Paz, Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Dordrecht, 2000.

[Caux, 95] C. Caux, H. I. Pierreval, M. C. Portman, Les algorithmes génétiques et leurs applications aux problèmes d'ordonnancement. APII, vol. 29 (4), pp. 409-443, 1995

[Carlier, 88] J. Carlier, P. Chrétienne, Problèmes d'ordonnancement, Modélisation, Complexité, Algorithmes. Edition Masson, Paris, 1988.

[Carlier, 87] J. Carlier, Scheduling jobs with release dates and tails on identical machines to minimize the makespan. European Journal of Operational Research, vol. 29, pp. 298-306, 1987.

[Carlier, 84] J. Carlier, Problèmes d'ordonnancement à contraintes de ressources : algorithmes et complexité. Thèse de Doctorat, Université Paris VI, 1984.

[Chang, 06] P. C. Chang, J. C. Hsieh, C. H. Liu, A case-injected genetic algorithm for single machine scheduling problems with release time. Int. J. Prod. Econ., vol. 103 (2), pp. 551–564, 2006.

[Chang, 07] P. C. Chang, S. H. Chen et C. H. Liu, Sub-population genetic algorithm with mining gene structures for multi-objective flow-shop scheduling problems. Expert Systems with Applications, vol. 33, pp. 762–771, 2007.

[Chang, 08] P. C. Chang, S. S. Chen, C. Y. Fan, Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. Applied Soft Computing, vol. 8, pp.767–777, 2008.

[Charon, 96] I. Charon, A. Germa, O. Hudry, Méthodes d'optimisation combinatoire, Éditions Masson. Paris, 1996.

[Chipperfield, 96] A. Chipperfield, P. Fleming, Parallel genetic algorithms. Parallel and Distributed Computing Handbook, McGraw-Hill, New York, pp. 1118–1143, 1996.

[Choobineh, 06] F. F. Choobineh, E. Mohebbi and H. Khoo, A multiobjective tabu search for a single-machine scheduling problem with sequence-dependent setup times. European Journal of Operational Research, vol. 175, pp. 318–337, 2006.

[Chou, 09] F. D. Chou, An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem. Expert Systems with Applications, vol. 36, pp. 3857–3865, 2009.

[Chu, 96] C. Chu, J. Proth, L'ordonnancement et ses applications. Sciences de l'Ingénieur, Edition Masson, Paris, 1996.

[Collette, 02] Y. Collette, P. Siarry, Optimisation Multiobjectif. Éditions Eyrolles, Paris, 2002.

[Colorni & al, 91] A. Colorni, M. Dorigo, V. Maniezzo, Distributed Optimization by Ant Colonies. Proceedings of the First European Conference on Artificial Life, Paris, F. Varela and P. Bourgine (Eds.), Elsevier Publishing, pp. 134–142, 1991.

[Colorni & al, 92] A. Colorni, M. Dorigo, V. Maniezzo, An Investigation of some Properties of an Ant Algorithm. Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92), Brussels, R. Manner and B. Manderick (Eds.), Elsevier Publishing, pp. 509-520, 1992.

[Davis, 91] L. Davis, Handbook of genetic algorithm. New York: Van Nostrand Reinhold, 1991.

[Deb, 01] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, West Sussex, England, 2001.

[De Falco, 94] I. De Falco, R. del Balio, E. Tarantino and R. Vaccaro, Improving search by incorporating evolution principles in parallel tabu search. In International Conference on Machine Learning, pp. 823-828, Orlando, 1994.

[De Falco, 95] I. De Falco, R. del Balio and E. Tarantino, An analysis of parallel heuristics for task allocation in multicomputers. Computing, vol. 3(59), pp. 259-275, 1995.

[Dorigo, 93] M. Dorigo, V. Maaniezzo, Parallel Genetic Algorithms, Introduction and Overview of Current Research. Journal Stender (Ed.), IOS Press, 1993.

[Dorigo, 97] M. Dorigo, L. M. Gambardella, Ant Colony System, A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, vol. 1(1), pp.53–66, April 1997.

[Dréo, 03] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, Métaheuristiques pour l'optimisation difficile. Editions Eyrolles, Paris, France,2003.

[Drira, 08] A. Drira, S. Hajri-Gabouj, A. Ben Amor, Approche Pareto de recuit simulé pour la résolution d'un problème industriel d'ordonnancement à une machine. La cinquième Conférence Internationale d'Electrotechnique et d'Automatique JTEA, 02-04 Mai 2008, Hammamet.

[Dupont, 05] A. Dupont, Étude d'une métaheuristique hybride pour l'affectation de fréquences dans les réseaux tactiques évolutifs. Thèse de Doctorat, Université des Sciences et Techniques du Languedoc Montpellier II, 2005.

[Durand, 94] N. Durand, J. M. Alliot, J. Noailles, Algorithmes génétiques, un croisement pour les problèmes partiellement séparables. Journées Evolution Artificielle Francophones JEAF, Toulouse, 1994.

[Duvier, 00] D. Duvier, Etude de l'hybridation des métaheuristiques, application à un problème d'ordonnancement de type jobshop. Thèse de Doctorat à

l'Université du Littoral Côte d'Opale, Calais, 2000.

[El Fallahi, 08] A. El Fallahi, C. Prins, R. W. Calvo, A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Computers & Operations Research, vol. 35, pp. 1725 – 1741, 2008.

[Ekşioğlu, 08] B. Ekşioğlu, S. Duni Ekşioğlu, P. Jain, A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods. Computers & Industrial Engineering, vol. 54, pp. 1-11, 2008.

[Etiler, 04] O. Etiler, B. Toklu, M. Atak, J. Wilson, A genetic algorithm for flow shop scheduling problems. Journal of the Operations Research Society, vol. 55, pp.830–835, 2004.

[Fleurent, 94] C. Fleurent and A. Ferland, Genetic hybrids for the quadratic assignment problem. DIMACS Series in discrete Mathematics and Theoretical Computer Science, vol. 16, pp. 173-188, 1994.

[Franchini, 98] L. Franchini, E. Caillaud, G. Lacoste, P. Nguyên, Planning and scheduling skills: a case study an agro-food industry. IEEE International Conference on System Man and Cybernetic, San Diego USA, Octobre 1998.

[Fraser, 57] A. S. Fraser, Simulation of genetic systems by automatic digital computers. Australian Journal of Biological Sciences, tome 10, pp. 484-491, 1957.

[Fonseca, 95] C. M. Fonseca, J. Peter, An overview of evolutionary algorithm in multiobjective optimization. Evolutionary Computation, vol. 3(1), pp. 1-16, 1995.

[Garey, 79] M. Garey, D. Johnson, Computer and Intractability: a Guide to the Theory of NP- Completeness. Freeman W-H, San Francisco, 1979.

[Gargouri, 02] E. Gargouri, S. Hammadi, P. Borne, Analyze of reactive schedule in agro-food chain, JTEA, 2ème Conférence Internationale, 21-23 Mars, Sousse, vol. 2, pp. 312-319, 2002.

[Gargouri, 03] E. Gargouri, Ordonnancement coopératif en industries agroalimen-

taires. Thèse de Doctorat, Université des Sciences et Technologies de Lille 1, 2003.

[Glover, 89] F. Glover, Tabu search, part I. ORSA. Journal of Computing, vol.1, pp. 190-206, 1989.

[Glover, 90] F. Glover, Tabu search, part II. ORSA. Journal of Computing, vol.2, pp. 4-32, 1990.

[Glover, 95] F. Glover, Tabu Thresholding: Improved Search by Nonmonotonic Trajectories, ORSA Journal on Computing, vol. 7(4), pp. 426-442, 1995.

[Glover, 97] F. Glover, M. Laguna, Tabu search, Kluwer Academic Publishers, Massachusetts, USA, 1997.

[Goldberg, 87] D. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization. In 2nd International Conference on Genetic Algorithms, pp. 41-49, 1987, New Jersey.

[Goldberg, 94] G. E. Goldberg, Algorithmes génétiques. Editions Addison Wesley, France, 1994.

[Goldberg, 00] D. E. Goldberg, The design of innovation: lessons from genetic algorithms, lessons for the real world. Tech. Forecasting and Social Change, vol. 64 (1), pp. 7–12, 2000.

[Gong, 05] Y. Gong, M. Nakamura, S. Tamaki, Parallel genetic algorithms on line topology of heterogeneous computing resources. In: H.-G. Bauer (Ed.), Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1447–1454, New York, 2005.

[Gotha, 93] Gotha, Les problèmes d'ordonnancements, RAIRO-Recherche Opérationnelle, Vol. 27, pp. 77-150, 1993.

[Gupta, 93] M. C. Gupta, A. Gupta, A. Kumar, Minimizing flow time variance in a single machine system using genetic algorithms. European Journal of Operational Research, vol. 70, pp. 289–303, 1993.

[Harbaoui, 09] I. Harbaoui, R. Kammarti, M. Ksouri, P. Borne, A genetic algorithm

for the multi-pickup and delivery problem with time windows. Studies in Informatics and Control, Vol. 18, N°2, pp. 173- 180, mars 2009.

[Harrath, 03] Y. Harrath. Contribution à l'ordonnancement conjoint de la production et de la maintenance: application au cas d'un job shop. Thèse de Doctorat à L'UFR des Sciences et Techniques de l'Université de Franche-Comté, Besançon, 2003.

[Hassine, 08] M. Hassine, N. Liouane, K. Ben Othman, Elaboration d'une métaheuristique d'aide à l'ordonnancement. La 5ème Conférence Internationale d'Electrotechnique et d'Automatique, JTEA, 02-04 Mai 2008, Hammamet.

[He, 07] H. He, O. Sýkoraa, A. Salagean, E. Mäkinen, Parallelization of genetic algorithms for the 2-page crossing number problem. Journal of Parallel Distributed Computing, vol. 67, pp. 229 – 241, 2007.

[Holland, 75] J. H. Holland, Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press, 1975.

[Hwang, 79] C. Hwang, A. Masud, Multiple objective decision making - methods and applications. In Lectures Notes in Economics and Mathematical Systems, volume 164. Springer-Verlag, Berlin, 1979.

[Iyer, 04] S.K. Iyer, B. Saxena, Improved genetic algorithm for the permutation flowshop scheduling problem. Computers and Operations Research, vol. 31, pp. 593–606, 2004.

[Jain, 99] A. S. Jain, S. Meeran, Deterministic Job-shop scheduling past present and future, European Journal of Operational Research. vol. 113, n°2, pp. 390-434, 1999.

[James, 09] T. James, C. Rego, F. Glover, A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research, vol. 195, pp. 810–826, 2009.

[Jin, 09] F. Jin, S. Song, C. Wu, A simulated annealing algorithm for single machine scheduling problems with family setups. Computers & Op-

erations Research, vol. 36: 2133- 2138, 2009.

[Kacem, 02] I. Kacem, S. Hammadi, P. Borne, Pareto-optimality for flexible job-shop scheduling problem: hybridization of genetic algorithm with fuzzy logic. Mathematics and Computers in Simulation, Vol. 60, pp. 245-276, 2002.

[Kacem, 03] I. Kacem, Ordonnancement multicritère des job-Shop flexibles : formulation, bornes inférieures et approche évolutionniste coopérative. Thèse de Doctorat, Université des Sciences et Techniques de Lille1, 2003.

[Kammarti, 05] R. Kammarti, S. Hammadi, P. Borne, M. Ksouri, Special Tabu Search in an Hybrid Evolutionary Approach for the PDPTW. SCI IEEE International Conference, Orlando (USA), Juillet 2005.

[Kammarti, 06] R. Kammarti, Approches évolutionnistes pour la résolution du 1-PDPTW statique et dynamique. Thèse de Doctorat, Ecole Centrale de Lille, Université des sciences et technologies de Lille, 2006.

[Karray, 08] A. Karray, K. Laabidi, M. Ksouri, Métaheuristiques pour la commande des systèmes non linéaires. CIFA Conférence International Francophone d'Automatique, 3-5 Septembre, Bucarest, 2008.

[Karray, 10] A. Karray, M. Benrejeb, P. Borne, Sur l'hybridation des métaheuristiques Algorithmes Génétiques et Recherche Tabou pour la résolution de problèmes d'ordonnancement en industries agroalimentaires. ROADEF 11ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, 24-26 Février, Toulouse, France, 2010.

[Karray, 11(a)] A. Karray, M. Benrejeb, P. Borne, Algorithmes Génétiques Séquentiel pour la résolution de problèmes d'ordonnancement en industries agroalimentaires. Revue électronique Sciences et Technologies de l'Automatique e-STA, Vol. 8, N°1, 2011.

[Karray, 11(b)] A. Karray, M. Benrejeb, P. Borne, New Parallel Genetic Algorithms for the single-machine scheduling problems in agro-food industry.

International Conference on Communications, Computing and Control Application CCCA'11, march 3-5, Hammamet (Tunisia), 2011.

[Karray, 11(c)] A. Karray, M. Benrejeb, P. Borne, A hybrid method based on genetic algorithms and tabu search for the single-machine scheduling problem in agro-food industry. International Review of Automatic Control (IREACO), Vol. 4, N°2, march 2011.

[Kirkpatrick, 83] S. Kirkpatrick, C. D. Jr, M. P. Vecchi, Optimization by simulated annealing. Science, vol. 220, pp. 671-680, 1983.

[Koksalan , 03] M. Koksalan, A. B. Keha, Using genetic algorithms for single machine bicriteria scheduling problems. European Journal of Operational Research, vol. 145, pp. 543–556, 2003.

[Köktener, 00] E. Köktener, M. Koksalan, A simulated annealing approach to bicriteria scheduling problems on a single machine. Journal of Heuristics, vol. 6, N°3: 311-327, 2000.

[Lenstra, 77] J. K. Lenstra, A. H. G. Rinnooy Kan, P. Bruker, Complexity of machine scheduling problems. Annals of Discrete Mathematics, Vol. 1, pp. 343-362, 1977.

[Le Pape, 95] C. Le Pape, Three mechanisms for managing resource constraints in the library for constraint-based scheduling. INRIA/IEEE Conference on Emerging Technologies and Factory Automation, Paris, pp. 980-995, 1995.

[Lerman, 95] I. Lerman, F. Ngouenet, Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités. Rapport de Recherche de l'INRIA Rennes - Projet REPCO 2570, INRIA, 1995.

[Liaw, 99] C. F. Liaw, A tabu search algorithm for the open shop scheduling problem. Computers & Operations Research, vol. 26, pp. 109-126, 1999.

[Liaw, 00] C. F. Liaw, A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research, Vol. 124, pp. 28–42, 2000.

[Lopez, 99] P. Lopez, P. Esquirol, Ordonnancement, Edition Economica, Paris, France, 1999.

[Lopez, 01] P. Lopez, F. Roubellat, Ordonnancement de la production. Editions Hermès, Paris, France, 2001.

[Low, 05] C. Y. Low, Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Computers and Operations Research, vol. 32(8), pp. 2013–2025, 2005.

[Madureira, 01] A. Madureira, C. Ramos, S. D. C. Sliva, A GA based scheduling system for dynamic single machine problem. In Proceeding of the 4th IEEE International Symbosium on Assembly and Task Planning Soft Research Park, pp. 28–129, Fukuoka, 2001.

[Mayer, 99] M. K. Mayer, A Network Parallel Genetic Algorithm for the One Machine Sequencing Problem. Computers and Mathematics with Applications, vol. 37, pp. 71-78, 1999.

[Mesghouni, 98] K. Mesghouni, S. Hammadi, P. Borne, On modeling genetic algorithm for flexible job-shop scheduling problems. Studies in Informatics and Control, Vol. 7, N°1, pp. 37- 47, mars 1998.

[Mesghouni, 99] K. Mesghouni, Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production. Thèse de Doctorat, Université des Sciences et Technologies de Lille, 1999.

[Mesghouni, 04] K. Mesghouni, S. Hammadi, P. Borne, Evolutionary algorithms for job-shop scheduling. International Journal of Applied Mathematics and Computer Science, Vol. 14, N°1, pp. 91-103, 2004.

[Metropolis & al, 53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, Equation of State Calculations by Fast Computing Machines. J. Chem. Phys., vol. 21(6), pp. 1087–1092, 1953.

[Nakib, 06] A. Nakib, H. Oulhadj and P. Siarry, Segmentation optimale d'image par recuit simulé. Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'06), Février 6-8,

2006, Lille.

[Pareto, 64] V. Pareto. Cours d'économie politique, Lausanne : Rouge, 1896-7, reproduit dans Vilfredo Pareto, Œuvres complètes, Genève, Librairie Droz, 1964.

[Pinedo, 55] M. Pinedo, Scheduling: Theory, Algorithms and Systems. Prentice-Hall, Englewood Clis, New Jersey, 1955.

[Porumbel, 09] C. D. Porumbel, J. K. Hao and P. Kuntz, Position guided tabu search for graph coloring. In Selected Papers from the International Conference on Learning and Intelligent Optimization, vol. 5851 of LNCS, pp. 148-162, 2009.

[Rodammer, 88] F. A. Rodammer and K. Preston White, A recent survey of production Scheduling. IEEE Transaction on Systems, Man and Cybernetics, pp. 6-18, 1999.

[Ruiz, 07] R. Ruiz, J. C. García-Díaz and C. Maroto, Considering scheduling and preventive maintenance in the flow-shop sequencing problem. Computers and Operations Research, vol. 34, pp. 3314-3330, 2007.

[Sakarovitch, 84(a)] M. Sakarovitch, Graphes et Programmation Linéaire. Edition Hermann, Paris, 1984.

[Sakarovitch, 84(b)] M. Sakarovitch, Programmation Discrète. Edition Hermann, Paris, 1984.

[Saldanha, 92] R. R. Saldanha, Optimisation en électromagnétisme par application conjointe des méthodes de programmation non linéaire et de la méthode des éléments finis. Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1992.

[Taillard, 00] É. D. Taillard, An Introduction to Ant Systems. Kluwer, Computing Ttools for Modelling, Optimization and Simulation, Interfaces in Computer Science and Operations Research Edition, chapter 7, pp. 131– 144, 2000.

[Talbi, 94] E. G. Talbi, T. Muntean, and I. Samarandache, Hybridation des algo-

rithmes génétiques avec la recherche tabou. In Evolution Artificielle (EA'94), 1994.

[Talbi, 02] E. G. Talbi, A taxonomy of hybrid metaheuristics. Journal of Heuristics, vol 8(2), pp. 541-564, 2002.

[Talbi, 09] E. G. Talbi, A. Liefooghe, and L. Jourdan, Multi-objective metaheuristics: from design to implementation. In 23rd European Conference on Operational Research, Born, 2009.

[Tanese, 87] R. Tanese, Parallel genetic algorithms for a hypercube. In 2nd International Conference on Genetic Algorithms, pp. 177-183, 1987.

[Tangour, 07] F. Tangour, Ordonnancement Dynamique dans les Industries Agroalimentaires. Thèse de Doctorat à L'Ecole Centrale de Lille, Université de Lille 1, 2007.

[Tangour, 09] F. Tangour, P. Borne, Presentation of some Metaheuristics for the optimization of complex Systems. Studies in Informatics and Control, Vol. 18 (2), pp. 173-180, 2009.

[Treillon, 96] R. Treillon, C. Lecomte, Gestion industrielle des entreprises alimentaires : techniques et pratiques de la gestion des flux, Edition Lavoisier Tec et Doc, Paris, Décembre 1996.

[Venditti, 10] L. Venditti, D. Pacciarelli, C. Meloni, A tabu search algorithm for scheduling pharmaceutical packaging operations. Computers & Operations Research, vol.37: 1924–1938, 2010.

[Weng, 02] M. X. Weng, M.M. Sedani, Schedule one machine to minimize early/tardy penalty by tabu search. In Proceedings of the Annual IIE Research Conference, Orlando, FL, 2002.

[Widmer, 01] M. Widmer, Les Métaheuristiques : Des outils performants pour les problèmes industriels. 3ème Conférence Francophone de MOdélisation et SIMulation MOSIM'01, 25-27 avril, Troyes, 2001.

[Wood, 97] G. W. Wood, A hybrid Genetic Algorithm that adapts to binary and real coded operators. PhD Thesis, Department of Computer Science RMIT, Melbourne, 1997.

[Ying, 09] K. C. Ying, S. W. Lin, C. Y. Huang, Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic. Expert Systems with Applications, vol. 36, pp. 7087–7092, 2009.

[Zhang,10] R. Zhang, C. Wu, A hybrid immune simulated annealing algorithm for the job shop scheduling problem. Applied Soft Computing, vol. 10, Issue 1, pp. 79-89, 2010.

Contribution à l'ordonnancement d'ateliers agro-alimentaires utilisant des méthodes d'optimisation hybrides

Résumé

Nos travaux concernent la mise en œuvre de méthodologies pour la résolution de problèmes d'ordonnancement en industries agroalimentaires. Trois nouvelles approches basées sur les algorithmes génétiques, sont proposées pour la résolution de problèmes d'ordonnancement multi-objectifs : les algorithmes génétiques séquentiels (SGA), les algorithmes génétiques parallèles (PGA) et les algorithmes génétiques parallèles séquentiels (PSGA). Deux approches coopératives multi-objectifs en mode relais, SH_GA/TS et SH_GA/SA, hybridant toutes les deux des métaheuristiques de haut niveau, sont par la suite proposées. Un algorithme évolutionnaire et un algorithme de recherche locale sont, dans ce cas exécutés séquentiellement.

Mots-clés : ordonnancement, optimisation, métaheuristiques, algorithmes génétiques, recherche tabou, recuit simulé, hybridation, ateliers à une machine

Using hybrid optimization methods for the agro-food industry scheduling problem

Abstract

The purpose of our works is the implementation of methodologies for the resolution of the agro-food industry scheduling problem. Three new approaches based on genetic algorithms are proposed to solve multi-objectives scheduling problems: sequential genetic algorithms (SGA), parallel genetic algorithms (PGA) and parallel sequential genetic algorithms (PSGA). Two high-level hybrid algorithms, SH_GA/TS and SH_GA/SA, are also proposed. The purpose in this hybridization is to benefit the exploration of the solution space by a population of individuals with the exploitation of solutions through a smart search of the local search algorithm.

Key-words: scheduling, optimization, metaheuristics, genetic algorithm, tabu search, simulated annealing, hybridation, single-machine workshop