N° : 07/2015-M/MT

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene
Faculté de Mathématiques

Mémoire
présenté pour l'obtention du diplôme de Magistère
En : MATHÉMATIQUES
Spécialité : Recherche Opérationnelle
Par : BAAZIZ ADLANE

Thème

Optimisation Multi-objectif des Chaînes Logistiques par les Algorithmes Génétiques

Soutenu publiquement, le : 05-12-2015, devant le jury composé de :

M. CHAABANE Djamel	Professeur	à l'U.S.T.H.B	Président.
M. MOULAÏ Mustapha	Professeur	à l'U.S.T.H.B	Directeur de mémoire.
M. CHERGUI Med El-Amine	Professeur	à l'U.S.T.H.B	Examinateur.

Remerciements

Je tiens tout d'abord à remercier dieu, le tout puissant de ma donnée la patience, la santé et le courage pour finir ce travail.

Je tiens en tout premier lieu à remercier Monsieur MOULAÏ Mustapha, Professeur à l'Université des Sciences et de la Technologie Houari Boumediene (USTHB) qui a accepté d'être mon Directeur de thèse. Je le remercie pour son aide, ses orientations, sa disponibilité et sa patience, ses conseils judicieux m'ont permis de travailler dans les meilleures conditions.

J'adresse mes remerciements à Monsieur CHAABANE Djamel, Professeur à l'USTHB, pour l'honneur qu'il m'a fait en acceptant de présider le jury. Je tiens à exprimer mes remerciements à Monsieur CHERGUI Mohamed El-Amine, Professeur à l'USTHB, qui a accepté d'examiner ce travail et d'être membre de jury.

Je tiens également à remercier tous mes enseignants pour la qualité de l'enseignement qu'ils ont bien assuré.

Je dédie ce modeste travail à :

Ma très chère grand-mère
Mes très chers parents
Mon très cher petit frère Sofiane
Mes soeurs qui m'ont toujours soutenu
Tous mes amis sans citer les noms
Toute personne qui mérite l'appréciation et le respect de ma part
Merci à vous.

BAAZIZ Adlane.

Table des matières

Liste des tableaux

Table des figures

Introduction Générale

La plupart des problèmes d'optimisation réels sont décrits à l'aide de plusieurs objectifs souvent contradictoires devant être optimisés simultanément. Alors que, pour les problèmes n'incluant qu'un seul objectif, l'optimum cherché est clairement défini, celui-ci reste à formaliser pour les problèmes d'optimisation multi-objectif. En effet, pour un problème à plusieurs objectifs contradictoires, la solution optimale cherchée est un ensemble de points appelé solutions Pareto-Optimales correspondant aux meilleurs compromis possibles pour résoudre ce problème.
L'optimisation multi-objectif est une branche de l'optimisation combinatoire (OC) dont l'objectif est de résoudre les problèmes d'optimisation multi-objectif, qui sont une généralisation à n fonctions objectifs des problèmes d'optimisation classiques. c.à.d. calculer la surface de solutions qui offrent un bon compromis entre les différents objectifs. Les difficultés rencontrées lors de la résolution d'un problème multi-objectif, en plus de la présence de contraintes, nous avons aussi celles liées aux propriétés des fonctions à optimiser. Ces dernières, selon la méthode d'optimisation employée, peuvent constituer des obstacles quant à la progression vers le front optimal global. Tous ces obstacles posent des obstacles difficiles à franchir pour les méthodes de résolution exactes.
Pour faire face aux difficultés rencontrées par les méthodes de résolution exactes, beaucoup de méthodes métaheuristiques ont été développées. Ces méthodes, sont reconnues pour être très performantes, et rencontrent un succès considérable dans le monde de l'industrie. Ces dernières sont généralement inspirées des phénomènes naturelles et souvent très facile à implémenter, aussi avec une grande capacité d'adaptation aux différents types de problème d'optimisation quel que soit leurs formulations mathématiques linéaire ou factionnaire ou autres, stochastique ou non, mono-objectif ou multi-objectif.
L'optimisation des chaînes logistiques, et particulièrement la politique de distribution constitue une préoccupation majeure des entreprises dans un environnement fortement concurrentiel, elles cherchent à performer leurs systèmes de gestion et doivent être

réactives en proposant un rapport qualité/prix plus compétitif. Traditionnellement, les problématiques liées aux fonctions achats, production, distribution, planification et autres logistiques sont traitées de façons séparées par les décideurs, mais dans la pratique les objectifs sont généralement contradictoires. Pour surmonter ces contradictions entre les objectifs d'une même chaîne logistique, les décideurs sont tenus à fixer un mécanisme où les différentes fonctions objectifs (minimisation des coûts transport/production, les coûts d'achat et de maximiser les profits et le niveau de service à la clientèle, etc.) peuvent être intégrées ensemble. Des efforts considérables ont été dépensés dans le développement de modèles de décision pour les problèmes de la chaîne logistique.

Dans le mémoire de magister, nous nous sommes intéressés à un problème d'optimisation d'une chaîne logistique globale, où les décisions de sélection des fournisseurs, localisations des usines/centres de distribution(CDs) à installer, l'affectation des clients aux CDs, des CDs aux usines et des usines aux fournisseurs, sont intégrées dans un même modèle d'optimisation multi-objectif non-linéaire en variables mixtes. L'objectif est de concevoir une plate-forme optimale qui réponde au maximum aux besoins des demandes, pour cela nous avons considéré trois fonctions objectifs, le premier objectif sert à minimiser le coût total de la chaîne, le deuxième sert à maximiser le taux de satisfaction des demandes (en %) dans un délai de livraison τ fixé au préalable et le troisième objectif sert à maximiser l'équilibre d'utilisation des capacités des ressources usines et des CDs, ceci est traduit par la minimisation de l'erreur quadratique moyenne (MSE) de taux d'utilisation des capacités. C'est dans ce contexte que s'inscrit notre travail de magister présenté par un mémoire organisé en quatre chapitres :

Le premier chapitre présente quelques généralités sur l'optimisation combinatoire mono-objectif, la complexité et la résolution des problèmes d'OC. Dans un second temps nous présentons des généralités sur l'optimisation multi-objectif, formulation, complexité et résolution des problèmes d'optimisation multi-objectif.

Dans le deuxième chapitre nous commençons par un rappel sur les algorithmes génétiques, pour ensuite présenter les travaux réalisés pour l'adaptation des algorithmes génétiques pour l'optimisation multi-objectif.

Le troisième chapitre est consacré dans un premier temps à la présentations des chaînes logistiques, leurs définitions, leurs conceptions, leur gestion, et de la prise des décisions dans différents niveaux de la chaîne, nous présentons aussi les travaux en relation avec la gestion et l'optimisation des chaînes logistiques. Dans un second temps nous nous intéressons à un problème d'optimisation multi-objectif d'une chaîne logistique où nous

adoptons une modélisation existante dans la littérature, dont la résolution de cette dernière mène à un problème NP-difficile, pour la résolution de ce modèle nous proposons une adaptation d'un algorithme génétique dans lequel, nous avons considéré quatre procédures de sélection, les deux premières variantes sont basées sur l'agrégation des objectifs en un seul objectif à l'aide de la technique de pondération. Dans la première variante, le calcul des poids associés aux objectifs utilisé pour évaluer et sélectionner les nouveaux individus est effectué de manière aléatoire. Tandis que dans la seconde variante, le calcul se fait en utilisant le point idéal correspondant à la population courante. Par contre, la troisième variante est dotée de la technique de sélection Non-dominated Sorting Genetic Algorithm (NSGA), où les individus sont triés par ordre de dominance. La dernière variante est basée sur la sélection Weighted Average Ranking (WAR), où les individus sont scindés en k groupes (k est le nombre d'objectifs) et chaque groupe est trié par ordre de valeur de leur objectif.

Le quatrième chapitre est dédié aux expérimentations comparant les quatres variantes de l'algorithme génétique proposé avec discussion des résultats obtenus. Dans un second temps nous adaptons l'algorithme génétique NSGA-II pour la résolution du problème d'optimisation multi-objectif dans les chaînes logistiques et les résultats des expérimentations seront présentés.

Nous finalisons notre travail par une conclusion en proposons des perspectives avec des pistes de recherche à exploiter.

Chapitre 1

Optimisation Combinatoire

1.1 Introduction

Dans ce chapitre nous définissons l'optimisation combinatoire (OC) et les éléments de base relatifs aux problèmes d'optimisation combinatoire, à savoir la formulation et la complexité algorithmique, aussi nous citons quelques méthodes dédiées à la résolution de ces derniers qui sont organisées en deux types exactes et approchées. Ensuite nous introduisons les éléments de bases de l'optimisation multi-objectif (OM), la formulation d'un problème d'OM et les caractéristiques d'un problème d'OM, nous présentons aussi une classification des méthodes de résolution des problème d'OM en citant quelques techniques de résolution.

1.2 Optimisation Combinatoire

L'optimisation combinatoire (OC) occupe une place très importante dans la recherche opérationnelle, dans les mathématiques discrètes et dans l'informatique. Son importance se justifie d'une part par la grande difficulté des problèmes abordés et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'OC.

Un problème d'OC consiste à trouver la meilleure solution dans un ensemble discret dit ensemble des solutions réalisables. En général, cet ensemble est fini mais de cardinalité très grande et il est décrit d'une manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes qui doivent satisfaire les solutions réalisables.

Pour définir la notion de **meilleure solution**, une fonction, dite fonction objectif, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution (ou **solution**

optimale) est celle qui minimise ou maximise la fonction objectif. Il est claire qu'un problème peut avoir plusieurs solutions optimales.

Définition 1.2.1 *[3] Un problème d'optimisation combinatoire est défini à partir d'un ensemble fini S et d'une application $f : S \to \Re$. Il s'agit de déterminer $\hat{s} \in S$ tel que :* $f(\hat{s}) = \boldsymbol{Min}_{s \in S}[f(s)]$.

1.2.1 Quelques notions sur la complexité des problèmes d'optimisation combinatoire

Un problème d'OC peut être résolu par plusieurs algorithmes, c'est pour-quoi qu'il est très important dans le cadre de la recherche opérationnelle de pouvoir comparer l'efficacité des différents algorithmes pouvant être mis en oeuvre pour résoudre ce problème. Deux critères principaux sont généralement considérés quand le chercheur opérationnel étudie l'efficacité d'un algorithme : le premier est le temps de calcul, le second est l'espace mémoire nécessaire à l'exécution de l'algorithme. Si l'espace mémoire reste un critère important pour l'évaluation de l'efficacité d'un algorithme, en pratique, dans la grande majorité des problèmes, le temps de calcul devient le paramètre principal mesurant l'efficacité d'un algorithme et c'est à cause de lui que les problèmes d'optimisation combinatoire sont réputés si difficiles. En effet, que le problème soit à traiter en temps réel, ou la quantité de temps dont on dispose pour obtenir une réponse au problème posé est toujours limitée. C'est la **Théorie de Complexité** des algorithmes qui permit à l'optimisation combinatoire d'asseoir sur une base théorique solide, en donnant un sens précis aux termes d'algorithmes "efficaces" .

Définition 1.2.2 *[3] Un algorithme est dit* **polynomial** *si le nombre d'opérations élémentaires nécessaires pour résoudre un problème de taille n est une fonction polynomiale en n, l'algorithme est considéré comme* **efficace** *si, et seulement si, il est polynomial.*

Dans la définition précédente, le terme opérations élémentaires est assez vague. On doit le comprendre comme des additions, des multiplications, des comparaisons, etc... (tout ce que le processeur sait réaliser en un temps fixe).

Définition 1.2.3 *[3]*
*- On dit qu'un problème est dans **P** s'il existe un algorithme pour le résoudre en un temps polynomial.*
*- On dit qu'un problème est dans **NP** s'il existe un algorithme pour vérifier qu'une solution donnée convient en un temps polynomial.*
*- On dit qu'un problème est **NP-complet** si la résolution de ce problème en temps polynomial entraîne la résolution en temps polynomial de tout problème NP.*

Les problèmes **NP-complets** sont donc les plus compliqués des problèmes NP.

Tout problème peut être résolu avec un algorithme de complexité polynomiale est considéré **"facile"**. Tout problème peut être résolu avec un algorithme de complexité exponentielle, ou pire qu'exponentielle, est considéré **"difficile "**.

1.2.2 Résolution des Problèmes d'Optimisation Combinatoire

La majorité des problèmes d'OC sont des problèmes NP-difficiles et donc ne possèdent pas à ce jour un algorithme efficace pour les résoudre. Ceci a motivé les chercheurs à développer de nombreuses méthodes de résolution. Ces méthodes appartiennent à deux grandes familles : les méthodes exactes et les méthodes approchées. Nous présentons dans ce qui suit les principales méthodes exactes et approchées connues dans la littérature.

1.2.2.1 Méthodes Exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des combinaisons de l'espace de recherche. Parmi les méthodes exactes, nous trouvons les méthodes de séparation et évaluation (branch and bound) et la programmation dynamique.

Séparation et Évaluation (Branch and Bound)
C'est une technique qui effectue un parcours en profondeur de l'arbre de recherche afin de fournir une ou plusieurs solutions optimales à partir d'un ensemble de solutions potentielles. A chaque étape de la recherche correspondant à un noeud de l'arbre de recherche, l'algorithme utilise une fonction Bound pour calculer une borne de l'ensemble des solutions du sous-arbre prenant sa racine à ce noeud. Au début de la résolution, cette borne est initialisée à une valeur maximale (cas de minimisation). Si cette évaluation

est moins bonne que la meilleure solution trouvée jusqu'à ce niveau de recherche, tout le sous-arbre peut être stérilisé. Il importe de souligner que l'efficacité de l'algorithme Branch and Bound dépend étroitement du calcul de la borne utilisée.

Programmation Dynamique[9]
C'est une méthode ascendante : On commence d'habitude par les sous problèmes les plus petits et on remonte vers les sous-problèmes de plus en plus difficiles. Elle est utilisée pour les problèmes qui satisfont au principe d'optimalité de Bellman : "Dans une séquence optimale (de décisions ou de choix), chaque sous-séquence doit aussi être optimale". Un exemple de ce type de problème est le plus court chemin entre deux sommets d'un graphe. L'idée de base est d'éviter de calculer deux fois la même chose, généralement en utilisant une table de résultats déjà calculés, remplie au fur et à mesure qu'on résout les sous problèmes.

1.2.2.2 Méthodes Approchées

Le but d'une méthode approchée est de trouver une solution réalisable tenant compte des contraintes du problème mais sans garantie d'optimalité. Nous distinguons parmi les méthodes approchées les approches par voisinage et les approches par construction. Les approches par voisinage partent d'une ou plusieurs solutions initiales qu'elles cherchent à améliorer à partir d'un voisinage défini alors que les approches constructives partent d'une solution vide et génèrent de nouvelles solutions de façon incrémentale en ajoutant itérativement des composantes aux solutions en cours d'exécution.

Les Méthodes Constructives (Heuristique)
Une Heuristique est un algorithme spécifique qui permet d'identifier au moins une solution réalisable à un problème d'optimisation donné, mais sans garantir que cette solution soit optimale. Elle permet d'aborder des problèmes ne pouvant pas être résolus par des techniques de résolution exactes, en particulier les problèmes difficiles d'optimisation combinatoire. Ces approches commencent à partir d'une solution vide qu'elles construisent au fur et à mesure de la recherche.

Les Méthodes Amélioratrices (Métaheuristiques)
Une Métaheuristique est une stratégie générale, applicable à un grand nombre de problèmes, l'idée de ces approches est de structurer l'ensemble des solutions en terme de

voisinage, et d'explorer cet ensemble en partant d'une ou plusieurs solutions initiales et en choisissant à chaque itération une ou plusieurs solutions voisines d'une ou plusieurs solutions courantes. Le voisinage d'une solution S est l'ensemble des solutions qui peuvent être atteintes à partir de S en un seul "mouvement", un mouvement étant par exemple, le changement de valeur d'une variable.

Dans ce qui suit nous allons donner une présentation de quelques Métaheuristiques :

- **Recherche Tabou**

  La recherche tabou est une méthode heuristique d'optimisation combinatoire développée par Glover [30]. Cette méthode effectue des mouvements d'une solution s à une meilleure solution s' appartenant au voisinage de s noté $N(s)$. Si ce voisinage est trop grand, seulement une partie $V(s) \subset N(s)$ sera examinée.

  Pour éviter de rester bloqué dans le premier minimum local rencontré, la recherche tabou accepte des solutions voisines dont la qualité est moins bonne que la solution courante. Elle se dirige alors vers la solution voisine qui dégrade le moins possible la fonction objectif. Cette dégradation de f que l'on espère être temporaire, devrait permettre à l'algorithme d'atteindre des régions de l'espace contenant des solutions meilleures que celles rencontrées jusque-là. Cependant, l'algorithme tabou risque de boucler en revisitant des solutions antérieures pendant l'itération suivante. Pour éviter ce phénomène, l'algorithme utilise une structure de mémoire dans laquelle il stocke les mouvements interdits à chaque étape afin de ne pas obtenir une solution déjà rencontrée récemment. Ces mouvements qui sont interdits à une itération donnée sont dits tabous, d'où le nom attribué à la méthode par Glover [5]. Le caractère tabou d'un mouvement doit être temporaire afin de donner une plus grande flexibilité à l'algorithme en lui permettant de remettre en question les choix effectués, une fois que les risques de bouclage ont été écartés. La façon la plus simple de mettre en oeuvre ce système d'interdictions consiste à garder en mémoire les derniers mouvements effectués et à empêcher l'algorithme de faire les mouvements inverses à ces derniers mouvements. Ces mouvements sont mémorisés dans une liste T, appelée Liste tabou. La gestion de cette liste se fait de manière circulaire en remplaçant à chaque itération le mouvement le plus ancien de la liste par le mouvement courant. Cependant, il peut se trouver des cas où il est intéressant de révoquer le statut tabou d'un mouvement lorsque son application à la solution courante permet d'atteindre une solution meilleure. La mise en oeuvre d'un tel mécanisme est réalisée à l'aide d'une fonction auxiliaire A qui est appelée fonction

d'aspiration.

Dans la recherche tabou, le compromis entre les mécanismes d'intensification et de diversification peut être géré par la taille de la liste tabou. Pour favoriser l'intensification de la recherche il suffit de diminuer la taille de la liste tabou, et contrairement, pour favoriser la diversification il s'agit d'augmenter la taille de cette liste.

– **Recuit Simulé**

La méthode du recuit simulé repose sur une analogie avec la thermodynamique. En effet, la recherche par un système physique des états d'énergie les plus bas est l'analogue formel d'un processus d'optimisation combinatoire.

Le recuit est un processus physique de chauffage. Un système physique porté à une température assez élevée devient liquide dans ce cas le degré de liberté des atomes qui le composent augmente. Inversement lorsque l'on baisse la température le degré de liberté diminue jusqu'à obtenir un solide. Suivant la façon dont on diminue la température on obtient des configurations d'énergie différentes :

- Baisse brutale de la température, la configuration atteinte est le plus souvent un état méta-stable, dont l'énergie est supérieure à celle du minimum absolu. Le système est en quelque sorte piégé dans ce minimum local.
- Baisse progressive de la température de façon à éviter de piéger le système dans des vallées d'énergie élevée, pour l'envoyer vers les bassins les plus importants et les plus profonds, là où les baisses ultérieures de température le précipiteront vers les fonds.

Cette méthode d'optimisation a été proposée en 1982 par S. Kirkpatrick et al [6], à partir de la méthode de Metropolis [7] qui était utilisée pour modéliser les processus physiques. Kirkpatrick et al se sont inspiré de la mécanique statistique en utilisant en particulier la distribution de Bolzmann. Le recuit simulé permet de sortir d'un minimum local en acceptant avec une certaine probabilité une dégradation de la fonction. Ainsi, la probabilité p qu'un système physique passe d'un niveau d'énergie E_1 à un niveau E_2 est donnée par :

$$P = e^{-\frac{E_2 - E_1}{k_b.T}}$$

k_b est la constante de Bolzmann, T est la température du système.

Comme le montre cette formule la probabilité d'observer une augmentation de l'énergie est d'autant plus grande que la température est élevée, donc au niveau du recuit

simulé :

- Une diminution de la fonction sera toujours acceptée ;
- Une augmentation de la fonction sera acceptée avec une probabilité définie selon une formule du type précédent.

Dans cette méthode, les mécanismes d'intensification et de diversification sont contrôlés par la température. En effet, la température décroît au fil du temps de sorte que la recherche tend à s'intensifier vers la fin de l'algorithme.

1.3 Optimisation Multi-Objectif

Dans la plupart des cas pratiques, lorsqu'on est face à un problème d'optimisation, la prise en compte d'un critère unique est malheureusement insuffisante. En effet, on rencontre rarement ce cas de figure. La plupart du temps, on a besoin d'optimiser simultanément plusieurs critères souvent contradictoires. Dans ce type de problèmes, on aura recours à l'optimisation multi-objectif. Il ne s'agit plus dans ce cas de trouver une solution optimale mais une multitude (ensemble) de solutions du fait que certains critères sont conflictuels, cet ensemble de solutions représente un compromis acceptable entre les différents objectifs souvent contradictoires.

En effet, si l'on prend l'exemple du dimensionnement d'une poutre devant supporter une charge donnée reposant en son milieu, on va vouloir obtenir une poutre de section la plus petite possible, et subissant la plus petite déformation possible. Dans cet exemple, de manière intuitive, on s'aperçoit que répondre à l'objectif « poutre petite section » ne va pas du tout dans le sens de répondre à l'objectif « petite déformation » (voir Coello Coello, Christiansen, [11] pour la modélisation mathématique de ce problème).

Donc, quand on résoudra un problème d'optimisation multi-objectif (POM), on obtiendra une grande quantité de solutions. Ces solutions, comme on peut s'en douter, ne seront pas optimales, au sens où elles ne minimiseront pas tous les objectifs du problème.

Un concept intéressant qui nous permettra de définir les solutions obtenues, est le compromis. En effet, les solutions que l'on obtient lorsque l'on a résolu le problème sont des solutions de compromis. Elles minimisent un certain nombre d'objectifs tout en dégradant les performances sur d'autres objectifs.

1.3.1 Problème d'Optimisation Multi-Objectif

Un problème d'OM est un problème dont on recherche l'action qui satisfait un ensemble de contraintes tout en optimisant un vecteur de fonctions multi-objectif c.à.d. optimiser plusieurs objectifs contradictoires ou conflictuels.
De ce qui précède on peut définir un problème d'optimisation objectif comme suit :

$$(POM) = \begin{cases} MinZ(x) = (z_1(x), z_2(x), ..., z_k(x)) \quad avec\ k \geq 2 \\ \qquad\qquad\qquad\qquad\qquad\qquad sous\ contraintes\ x \in S. \end{cases} \tag{1.1}$$

Avec :

- $x = (x_1, x_2, ..., x)$ représente l'action ou un vecteur de décisions, ou les x_i sont les variables du problème et n le nombre de ces variables.
- $Z(x) = (z_1(x), z_2(x), ..., z_k(x))$ est le vecteur de k fonctions objectifs z_i ou critères de décision, et k la taille de ce vecteur.
- S est l'ensemble non vide des solutions réalisables du problème.

Soit Z_S l'ensemble des objectifs réalisables défini par les images des solutions réalisables x dans S, on associe à S l'ensemble $Z(x)$ des objectifs correspondants à ces composantes $(x \in S)$, $z(x) = (z_1(x), z_2(x), ..., z_k(x))$ dans R^k et on construit donc, le polyèdre : $Z_S = \{z \in R^k | z = Cx; x \in S\}$.

Si l'ensemble S est un polyèdre convexe alors l'ensemble Z_S est un polyèdre convexe dans R^k dont les sommets sont des images des sommets de S (figure 2.1).
Sans perte de généralité nous supposerons par la suite que nous considérons des problèmes de minimisation.

1.3.2 Théorie d'Optimisation au sens de Pareto

1.3.2.1 Optimum de Pareto

Au XIX^{ème} siècle, Vilfredo Pareto [11], un mathématicien italien, a formulé le concept suivant : dans un problème d'OM, il existe un équilibre tel l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères. Cet équilibre est appelé optimum de Pareto. un point x est dit Pareto-optimal s'il n'est dominé par aucun autre point appartenant à S. Ces points sont également appelés solutions non inférieures ou non dominées.

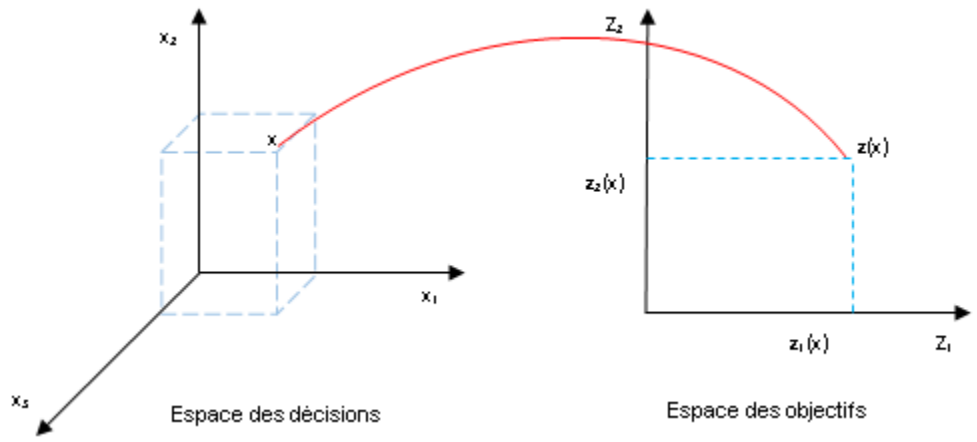


FIGURE 1.1 – Espace des objectifs Z_S

1.3.2.2 Notion de Dominance

Le principe de dominance est utilisé par la plus part des algorithmes d'optimisation multi-objectifs pour comparer deux solutions.

Définition 1.3.1 *(Principe de Dominance) Une solution $x^{(i)}$ domine une autre solution $x^{(j)}$ si les deux conditions suivantes sont vérifiées :*

1. $z_k(x^{(i)}) \leq z_k(x^{(j)}) \quad \forall k \in \{1, ..., K\}$;
2. $\exists k \in \{1, ..., K\} \quad telque \quad z_k(x^{(i)}) < z_k(x^{(j)})$.

Si $x^{(i)}$ domine $x^{(j)}$, cette relation est notée $x^{(i)} \prec x^{(j)}$.

D'autres relations qui peuvent être dérivées de la relation de dominance sont énoncées ci-dessous :

Définition 1.3.2 *(Dominance Stricte) : Une solution $x^{(i)}$ domine strictement une solution $x^{(j)}$ si et seulement si : $z_k(x^{(i)}) < z_k(x^{(j)}) \ \forall k \in \{1, ..., K\}$. Cette relation est notée $x^{(i)} \prec\prec x^{(j)}$.*

Définition 1.3.3 *(Dominance Faible) : Une solution $x^{(i)}$ domine faiblement une solution $x^{(j)}$ si et seulement si : $z_m(x^{(i)}) \leq z_k(x^{(j)}) \ \forall k \in \{1, ..., K\}$. Cette relation est notée $x^{(i)} \preceq x^{(j)}$.*

Définition 1.3.4 *(ϵ- Dominance) : Une solution $x^{(i)}$ ϵ- domine une solution $x^{(j)}$ si et seulement si :$z_k(x^{(i)}) \leq \epsilon . z_k(x^{(j)}) \ \forall k \in \{1, ..., K\}$. Cette relation est notée $x^{(i)} \preceq_\epsilon x^{(j)}$.*

Définition 1.3.5 *(ϵ- Dominance additive) : Une solution $x^{(i)}$ ϵ- domine additivement une solution $x^{(j)}$ si et seulement si :$z_k(x^{(i)}) \leq \epsilon + z_k(x^{(j)}) \quad \forall k \in \{1, ..., K\}$. Cette relation est notée $x^{(i)} \preceq_{\epsilon+} \ x^{(j)}$.*

1.3.2.3 Frontière de Pareto

Pour un ensemble de solutions fini, toutes les solutions peuvent être comparées deux à deux selon le principe de dominance, et nous pouvons déduire quelle solution domine l'autre. A la fin, nous obtenons un ensemble où aucune des solutions ne domine l'autre, cet ensemble est appelé ensemble des solutions non dominées ou bien ensemble des solutions Pareto-optimales.

Définition 1.3.6 *(Ensemble des solutions non-dominées) Soit P un ensemble de solutions, l'ensemble des solutions non dominées P' est l'ensemble des solutions non-dominées par aucun autre membre de l'ensemble P.*

Si l'ensemble P représente la totalité de l'espace de recherche S, l'ensemble des solutions non-dominées P' est appelé ensemble pareto-optimal dans l'espace des décisions ou front de pareto dans l'espace des objectifs.

Définition 1.3.7 *(Ensemble pareto-optimal) : l'ensemble pareto-optimal est l'ensemble des solutions non-dominées de l'espace de recherche faisable S.*

On dit qu'une solution $x \in S$ est efficace, s'il n'existe pas une autre solution $y \in S$ telle que le vecteur $Z(y)$ domine le vecteur $Z(x)$.

Résoudre un problème d'OM revient à trouver, soit l'ensemble des solutions efficaces dans l'espace des décisions, soit l'ensemble des solutions non dominées dans l'espace des critères.

1.3.2.4 Les Points Particuliers et la Matrice des Gains

En vue d'avoir certains points de références permettant de discuter de l'intérêt des solutions trouvées, des points particuliers ont été définis dans l'espace des critères. Ces points peuvent représenter des solutions réalisables ou non. Nous trouvons parmi ces points le point idéal, le point nadir et le point anti-idéal.

Point Idéal : Le point Idéal I est un point de l'espace des critères $\mathbb{R}^K$ de coordonnées : $I_k = Z_1^*, Z_2^*, \ldots, Z_k^*$ où $Z_j^* = \max_{x \in S} Z_j(x)$, $\forall j = 1, 2, \ldots, k$, pour un problème de maximisation.
En général, ce point ne correspond pas à une solution réalisable car si c'était le cas, cela sous-entendrait que les objectifs ne sont pas contradictoires et qu'une solution optimisant un objectif, optimise simultanément tout les autres objectifs, ce qui ramènerait le problème à trouver une seule solution pareto optimale.

Matrice des Gains : Soit $x^{*(l)}$ une solution optimale du critère Z_l non nécessairement unique. Notons par $Z_{kl} = Z_k(x^{*(l)})$. La matrice carrée G d'ordre K formée des éléments Z_{kl}, est appelée la matrice des gains.

$$G = \begin{pmatrix} Z_{1,1} & Z_{1,2} & \ldots & Z_{1,K} \\ Z_{2,1} & Z_{2,2} & \ldots & Z_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{K,1} & Z_{K,2} & \ldots & Z_{K,K} \end{pmatrix}$$

Les coordonnées du point idéal apparaissent sur la diagonale.
Notons que la matrice des gains n'est pas unique. En effet, s'il existe un critère Z_l ayant plus d'une solution optimale, l'évaluation Z_{kl} change suivant la solution optimale considérée.

Point Nadir : Le point nadir N est un point de l'espace des critères $\mathbb{R}^K$ ayant pour coordonnées :
$N_k = \min_{x \in eff} Z_k(x)$, $\forall k = 1, 2, \ldots, K$, où $eff \subset S$ est l'ensemble des solutions efficaces du problème de maximisation.

Remarque 1.3.1 *: Le point nadir correspond à la borne supérieure de la surface de Pareto, et non pas dans tout l'espace réalisable. La recherche des coordonnées du vecteur correspondant est un problème (pour le moins aussi difficile à résoudre) que celui de la recherche des solutions efficaces. Le point Nadir sert à restreindre l'espace de recherche, et il est utilisé dans certaines méthodes de résolutions.*

Une approximation du point nadir est donnée par : $\tilde{N}_k = \min_{l=\overline{1,K}} Z_{kl}$, $\forall k = 1, 2, \ldots, K$, qui représente le maximum de chaque colonne de la matrice des gains. Il est clair que si la matrice des gains G n'est pas unique, l'approximation du point nadir $\tilde{N}$ ne le sera pas.

Point anti-Idéal : Le point anti-idéal $\overline{I}$ est aussi un point de l'espace des critères ayant pour coordonnées :
$I_k = Z_1^*, Z_2^*, \ldots, Z_k^*, \forall k = 1, 2, \ldots, K$, où $Z_j^* = \min_{x \in S} Z_j(x), \forall j = 1, 2, \ldots, j$, pour un problème de maximisation.

1.3.3 Classification des Méthodes de Résolution des Problèmes Multi-Objectif

Dans la littérature, deux classifications principales des méthodes de résolution des problèmes d'OMs sont distinguées. La première classification adopte un point de vue décideur. La deuxième classe les méthodes suivant leur façon de traiter les fonctions objectifs.
La première classification divise ces méthodes en trois familles selon la manière dont sont combinés ces processus :

- **Les Méthodes a priori** : Dans ces méthodes, le décideur définit le compromis qu'il désire réaliser. Ainsi, il est supposé connaître à priori le poids de chaque objectif, ce qui revient à transformer le problème d'OM en un problème mono-objectif afin de pouvoir utiliser directement les méthodes de recherche.
- **Les Méthodes a posteriori** : Dans ces méthodes, le décideur choisit une solution parmi toutes les solutions de l'ensemble des solutions Pareto-optimales fourni par la méthode d'optimisation.ces dernières permettent au décideur d'exprimer ses préférences à l'issue du processus d'optimisation, parmi l'ensemble des solutions retenues (idéalement un ensemble de compromis optimaux au sens de Pareto), ce qui lui permet de tirer un profit maximum de l'étape d'optimisation.
- **Les Méthodes interactives** : Dans ces méthodes, il y a une coopération progressive entre la méthode d'optimisation et le décideur. Le décideur affine son choix de compromis au fur et à mesure du déroulement du processus de résolution à partir des connaissances acquises à chaque étape du processus.

On peut trouver des méthodes d'OM qui n'entrent pas exclusivement dans une famille. On peut utiliser, par exemple, une méthode à priori en lui fournissant des préférences choisies aléatoirement. Le résultat sera alors un ensemble pareto optimal à cardinalité élevée qui sera présenté au décideur pour qu'il choisisse une solution. Cette combinaison forme alors une méthode à posteriori. Cette classification permet donc seulement d'avoir une idée sur la démarche que l'on devra suivre pour obtenir la solution.

La deuxième classification divise les méthodes de résolution de problèmes d'OM en deux classes :

- La classe de méthodes transformant le problème initial en un problème mono-objectif, ainsi ces méthodes retournent une solution unique, elles utilisent généralement des méthodes d'agrégation.
- La classe de méthodes fournissant un ensemble de solution pareto optimal : cette classe est aussi divisée en deux sous-classes.

Il existe une autre classification qui tient en compte le principe de domination au sens de pareto, à savoir :

Les Méthodes Pareto : ces méthodes utilisent directement la notion de dominance au sens de Pareto dans la sélection des solutions générées. Cette idée a été initialement introduite par Goldberg [13] pour résoudre les problèmes proposés par Schaffer [14].

Les Méthodes non Pareto : ces méthodes optimisent séparément les objectifs.

1.3.4 Résolution d'un Problème Multi-Objectif

Pour la résolution des POM des méthodes exactes, ainsi que des heuristiques ont été proposées. Sans avoir la prétention de présenter ici la majorité des méthodes dédiées à l'optimisation multi-objectif, en voici quelques-unes afin d'avoir une vue plus globale.

1.3.4.1 Techniques Classiques pour l'Optimisation Multi-Objectif

Les méthodes classiques de résolution d'un problème d'OM ont le but de générer l'ensemble Pareto Optimal reposent généralement sur la transformation du problème multi-objectif initial (POM) en un problème d'optimisation mono-objectif (P). Les paramètres de la fonction employée ne sont pas fournis par le décideur mais par l'optimiseur. Plusieurs exécutions de l'optimiseur sont effectuées dans le but de trouver un ensemble de solutions qui approxime l'ensemble optimal de Pareto.

Plusieurs méthodes ont été proposées, nous citons pour exemple la méthode **Agrégation Pondérée** [15], la méthode $\epsilon-$**Contraintes** [16], et la méthode **Programmation par But** [17].

Ces méthodes vont être présentées avec plus de détails dans le paragraphe suivant.

Technique d'Agrégation Pondérée

L'idée de cette méthode la plus populaire et la plus utilisée dans la pratique est très intuitive, elle consiste à transformer le problème multi-objectif POM en un problème mono-objectif P en combinant linéairement les différents objectifs. Ainsi, le nouveau problème obtenu consiste à optimiser :

$$(P)\begin{cases} F(x) = \sum_i \lambda_i . f_i(x) \\ \sum_i \lambda_i = 1 \\ \lambda_i \geq 0 \end{cases} \tag{1.2}$$

La résolution du problème (P) , en faisant varier les valeurs du vecteur poids λ, on peut retrouver l'ensemble de solutions supportées, si le domaine réalisable est convexe. Cependant, elle ne permet pas de trouver les solutions enfermées dans une concavité (les solutions non supportées), ceci est un résultat des deux théorèmes suivantes :

Théorème 1.3.1 *(Théorème de Geoffrion [10]) Une solution optimale du problème* (1.2) *est une solution Pareto-Optimale pour le problème multi-objectif* (1.1) *si tous les poids* λ_i *sont strictement positifs.*

Théorème 1.3.2 *Si le problème d'optimisation multi-objectif* (1.1) *est convexe et* x^* *est une de ses solutions pareto-optimales, il existe un vecteur de poids positifs* $(\lambda_1, ..., \lambda_k)$ *tel que* x^* *est la solution du problème* (1.2).

L'approche généralement utilisée consiste à répéter l'exécution de l'algorithme avec des vecteurs poids différents. Les résultats obtenus avec de telles méthodes dépendent fortement des paramètres choisis pour le vecteur de poids λ. Les poids λ_i doivent également être choisis en fonction des préférences associées aux objectifs, ce qui est une tâche délicate.

Technique ϵ-Contraintes

Cette méthode a été suggérée par Haimes [16] en 1971, elle est basée sur la transformation de $K-1$ des k objectifs du problème (1.1) en contraintes et l'objectif restant qui peut être choisi arbitrairement, est la fonction objectif du problème (P) qui en résulte :

$$(P)\begin{cases} min_{x\in S}\ f_k(x), \\ f_m(x) \geq \epsilon_m, \quad m = 1, ..., K, m \neq k. \end{cases} \tag{1.3}$$

Le choix de différentes valeurs pour $\epsilon = (\epsilon_1, ..., \epsilon_{k-1}, \epsilon_{k+1}, ..., \epsilon_K)$ reste l'étape la plus difficile de cette méthode et ceci nécessite une très bonne connaissance à priori des intervalles appropriés pour les valeurs pour tous les objectifs du problème est requise afin de pouvoir générer l'ensemble des solutions pareto optimales.

Technique Programmation par But (Goal Programming)
Dans cette méthode également appelée target vector optimisation [17], le décideur doit définir un but T_i qu'il désire atteindre pour chaque objectif f_i. Ces valeurs sont introduites dans la formulation du problème le transformant en un problème uni-objectif. La nouvelle fonction objectif est modifiée de façon à minimiser la norme pondérée qui minimise les déviations par rapport aux buts par exemple. Le problème est formulé comme suit :

$$min \sum_{i}^{k} \mid f_i(x) - T_i \mid \quad avec\ x\ \in S \tag{1.4}$$

Différentes méthodes métaheuristiques ont été dévelopées en utilisant cette technique, nous citons pour exemple les suivantes :

- L'algorithme Génétique [18] dont la sélection est basée sur la performance des individus calculée comme la somme pondérée des objectifs.
- La méthode recherche tabou a été utilisée pour résoudre un problème d'affectation de fréquences [19].
- Le recuit simulé a été appliqué au problème de voyageur de commerce multi-objectif [20], et au problème du sac à dos multi-objectif [21], au problème de design de réseaux de transport [22].

1.3.4.2 Métaheuristiques pour l'Optimisation Multi-Objectif

Les métaheuristiques ont été largement utilisées pour la résolution des Problèmes d'optimisation multi-objectif. Ceci est d'autant plus vrai pour les Algorithmes Évolutionnaires travaillant sur des populations de solutions étant bien adaptés pour générer des fronts de solutions. Ainsi Deb publiait dés 2001 un livre portant sur l'utilisation des Algorithmes Évolutionnaires pour l'optimisation multi-objectif [16].
De part leur nature difficile, il existe très peu de méthodes exactes pour la résolution

de POMs, c'est pourquoi nous présentons ici des méthodes stochastiques approchées qui ont été appliquées avec succès à une large variété de problèmes académiques comme des problèmes réels.

Recuit Simulé

Le premier algorithme multi-objectif basé sur le Recuit Simulé a été proposé par Serafini [20] en 1992. La méthode utilise le schéma standard du Recuit Simulé avec une seule solution courante. Le résultat de l'algorithme est un ensemble de solutions pareto-optimales contenant toutes les solutions non-dominées générées par l'algorithme. Serafini considère un certain nombre de règles d'acceptance définissant la probabilité d'accepter des nouvelles solutions voisines. Le recuit simulé mono-Objectif accepte avec une probabilité égale à 1 les nouvelles solutions meilleures ou égales à la solution courante. Si la nouvelle solution est inférieure à la solution courante elle est acceptée avec une probabilité inférieure à 1.
Dans le cas multi-objectif, trois situations sont possibles en comparant une nouvelle solution x' avec la solution courante x :

- x' domine x,
- x' est dominée par x,
- x et x' sont mutuellement non-dominées.

Dans le premier cas, il est évident d'accepter x' avec une probabilité égale à 1. Dans le deuxième cas la probabilité d'acceptance doit être inférieure à 1. La question se pose pour la troisième situation quelle doit être la valeur de la probabilité d'acceptance. Serafini propose des règles d'acceptance multi-objectif qui traitent différemment la troisième situation. Ces règles correspondent à certaines fonctions d'agrégation locales. Ulungu et al [21] ont proposé une méthode appelée MOSA. Ils utilisent aussi des règles d'acceptance multi-objectif. MOSA utilise un nombre de vecteurs poids prédéfinis, chaque vecteur est associé à un processus de recuit indépendant. Chaque processus commence d'une solution aléatoire ou d'une solution construite par une heuristique spécialisée. Ensuite, la solution réalise des mouvements qui sont acceptés avec une probabilité définie par une règle d'acceptance. Le résultat de l'algorithme est un ensemble de solutions Pareto optimales contenant toutes les solutions non-dominées générées par tous les processus de recuit. Ainsi, ces processus, qui travaillent indépendamment, coopèrent dans la génération d'un ensemble commun de solutions pareto optimales. Czyzak et Jaszkiewicz [23] ont proposé la méthode "Pareto Simulated Annealing". Cette méthode utilise des fonctions scalaires basées sur les probabilités pour l'acceptance des nouvelles solutions voisines.

Une population de solutions est utilisée, chacune d'elles explorant l'espace de recherche relativement aux règles du recuit simulé. Les solutions peuvent être traitées comme des agents travaillant indépendamment mais échangeant des informations sur leurs positions. A chaque solution est associée un vecteur poids séparé. La méthode met à jour les vecteurs poids des solutions en construction en utilisant une règle pour assurer la dispersion des solutions sur toutes les régions du front pareto. Suppapitnarm et Parks [24] ont proposé une méthode dans laquelle la probabilité d'acceptance d'une nouvelle solution dépend du fait qu'elle est ajoutée ou non à l'ensemble des solutions Pareto optimales potentielles. Si elle est ajoutée à cet ensemble, ce qui signifie qu'elle n'est dominée par aucune autre solution trouvée, elle est acceptée pour être la solution courante avec une probabilité égale à 1. Sinon, une règle d'acceptance multi-objectif est utilisée.

Recherche Tabou

Grandibleux et al [25] ont proposé une version multi-objectif de la recherche tabou. La méthode utilise des fonctions pondérées scalaires dont les poids sont changés périodiquement. La modification du vecteur poids dégrade les poids des objectifs qui ont été nettement améliorés. Deux listes tabous sont utilisées. La première est une liste tabou régulière qui empêche de revenir aux solutions déjà visitées. La deuxième contient les vecteurs poids. Hansen [26] a proposé une recherche tabou basée sur l'idée de la méthode "Pareto Simulated Annealing" [23]. La méthode utilise une population de solutions explorant chacune d'elles différentes régions de l'ensemble Pareto. Le vecteur poids est utilisé dans une fonction scalaire. De plus, à chaque solution est associée une liste tabou. La dispersion des solutions est assurée par la modification de leurs vecteurs poids. Pour chaque solution, la modification du vecteur poids vise à la déplacer des autres solutions proches dans l'espace des objectifs. Hansen a appliqué cette méthode au problème du Sac à Dos multi-objectif.

Algorithmes Évolutionnaires

Les Algorithmes Évolutionnaires (AEs) sont des algorithmes stochastiques d'optimisation inspirés du paradigme darwinien de l'évolution naturelle. Selon la théorie darwinienne, les individus les plus aptes survivent à la sélection naturelle et se reproduisent d'une génération à l'autre. En termes d'optimisation, l'évolution se traduit par un processus itératif de recherche de l'optimum dans l'espace de recherche. Il existe plusieurs familles historiques d'Algorithme Évolutionnaires qui se sont développées de façon indépendante :

La **Programmation Évolutionnaire** introduite par L.J. Fogel [31] 1966, **les Algorithmes Génétiques** proposés par J.Holland [36], et popularisés un peu plus de dix ans plus tard par son élève D.E. Goldberg [37], **les Stratégies d'Évolutions** inventées par I.Rechenberg [34] dans les années 70 à Berlin, et enfin la **Programmation Génétique** proposée par J.Koza [34, 35].
Ces méthodes sont largement utilisée pour la résolution des problèmes d'optimisation multi-objectif, dans le chapitre prochain nous nous intéressons particulièrement au principe de base de fonctionnement des Algorithmes Génétiques, nous allons donner avec détails les travaux développés.

1.4 Conclusion

Nous avons défini dans la première partie de ce chapitre les problèmes d'optimisation mono-objectif ainsi que quelques exemples de ces problèmes. Nous avons présenté par la suite différentes approches pour la résolution de problèmes d'optimisation combinatoire. La deuxième partie était dédiée à l'optimisation multi-objectif où nous avons défini les Problèmes d'optimisation multi-objectif (POMs), les techniques de résolution de ces derniers et nous avons résumé quelques travaux qui ont été développés dans cette branche de l'optimisation combinatoire.

Chapitre 2

Algorithmes Génétiques et Optimisation Multi-Objectif

2.1 Introduction

Devant la complexité des problématiques d'optimisation le recours à des méthodes approchées devient de plus en plus important, parmi ces méthodes les Algorithmes Génétiques (AGs) sont des Métaheuristiques stochastiques d'optimisation globale, s'inspirant de la théorie de l'évolution. La souplesse d'utilisation de ces algorithmes pour des fonctions objectifs non régulières, à valeurs vectorielles, et définies sur des espaces de recherche non standard permet leur utilisation pour des problèmes qui sont pour le moment hors d'atteinte des méthodes classiques.

2.2 Les Algorithmes Génétiques

2.2.1 Historique

Les Algorithmes Génétiques sont des Métaheuristiques d'Optimisation Stochastiques qui simulent le processus de l'évolution naturelle. Apparue dans les années 30, la théorie synthétique de l'évolution (ou néodarwinisme) [53] intègre à la théorie darwinienne [54], la théorie de l'hérédité mendélienne [55] et la génétique des populations [56]. Elle donne une explication des mécanismes de différenciation d'individus au sein d'une population ainsi que le principe de transmission de caractéristiques d'individus à leur descendance. Les caractéristiques d'un organisme sont en grande partie codées dans ses gènes, une séquence

d'ADN différente pour chaque individu. Les sources de cette diversité sont des mutations pouvant apparaître dans ces gènes, ainsi que des recombinaisons se produisant lors de la reproduction sexuée.

Dès 1960, de nombreux chercheurs essayèrent d'appliquer les connaissances sur l'évolution naturelle à des problèmes informatiques. Dans les travaux [61, 62, 63] les chercheurs tentèrent de simuler l'évolution naturelle par des programmes informatiques. Mais ces tentatives furent infructueuses, probablement à cause de l'utilisation seule de la mutation. En 1975, John Holland proposa une première solution [64] appelée Algorithme Génétique. Ses recherches sur les processus d'adaptation des systèmes naturels ont permis la découverte de cette nouvelle technique d'exploration. Les mécanismes de base de ces algorithmes sont très simples mais les raisons pour lesquelles cette méthode fonctionne si bien sont plus complexes et subtiles. Les fondements mathématiques des AGs ont été exposés par Goldberg [65]. La simplicité de mise en oeuvre, l'efficacité et la robustesse sont les caractéristiques les plus attrayantes de l'approche proposée par Holland. La robustesse de cette technique est due à une sélection intelligente des individus les plus proches de la solution du problème.

2.2.2 Principe Général des Algorithmes Génétiques

La figure 2.1 suivante illustre le principe général des Algorithmes Génétiques (AGs) :

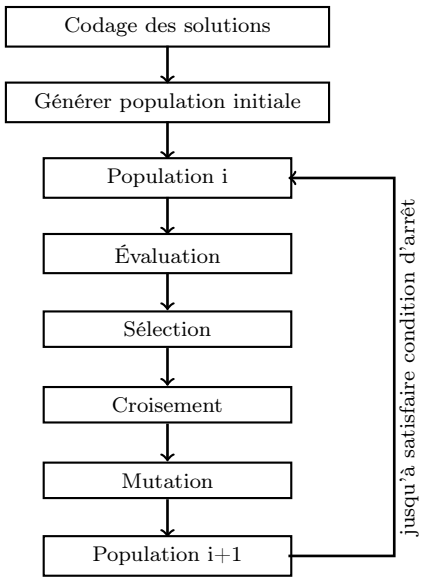


FIGURE 2.1 – Principe Général des AGs

2.2.2.1 Codage des solutions du problème

La première étape d'un AG est de définir et de coder convenablement les solutions du problème à traiter. En effet, le choix du codage dépend de la spécificité du problème et conditionne fortement l'efficacité de l'algorithme.
Historiquement, le codage utilisé par les AG était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'une solution du problème posé. Ce type de codage a pour intérêt de permettre de créer des opérateurs simples de Croisement et de Mutation. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus.
Cependant, le codage binaire n'est pas toujours bon pour des problèmes d'optimisation dans des espaces de recherche de très grande taille. Les AGs utilisant des vecteurs réels évitent ce problème en conservant les variables du problème dans le codage de l'élément de population sans passer par le codage binaire intermédiaire.
Des codages alphabétiques ou alphanumériques ont également été utilisés.

2.2.2.2 Génération de la population initiale

Le choix de la population initiale d'individus conditionne également fortement la rapidité de l'algorithme. Une connaissance de solutions de bonne qualité comme point d'initialisation permettrait à l'algorithme de converger plus rapidement vers l'optimum ou du moins s'y rapprocher. Les individus sont alors générés dans un sous-domaine particulier proche de ces solutions de départ.
Si par contre la position de l'optimum dans l'espace de recherche est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace de recherche, tout en veillant à ce que les individus produits respectent les contraintes.
Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités.
Les individus de la population sont générés, lorsque c'est possible, de telle façon à respecter les contraintes du problème.

2.2.2.3 Évaluation de la population

L'évaluation de chaque individu permet d'établir une hiérarchie au sein de la population. Elle détermine donc son aptitude à être sélectionné pour un croisement et à être gardé

pour la génération suivante. Il faudra définir pour l'ensemble des générations une fonction d'évaluation appelée généralement fitness. Cette fonction doit être capable d'interpréter les données contenues dans un chromosome et de décider si la solution résultante est optimale ou pas. Un élément de population dont les gènes ne forment pas une bonne solution se verra attribuer une mauvaise fitness et aura une probabilité forte d'être éliminé par le processus de sélection.
Il peut cependant être intéressant de conserver, tout en les pénalisant, les éléments de mauvaise fitness, car ils peuvent permettre de générer des éléments admissibles de bonne qualité. Gérer les contraintes en pénalisant la fonction fitness est difficile, un dosage s'impose pour ne pas favoriser la recherche de solutions admissibles au détriment de la recherche de l'optimum ou inversement. Disposant d'une population d'individus non homogène, la diversité de la population doit être entretenue au cours des générations afin de parcourir le plus largement possible l'espace d'état. C'est le rôle des opérateurs de croisement et de mutation.

2.2.2.4 Mécanisme de Sélection

Une fois la population initiale créée et évaluée, il faut sélectionner les individus qui participeront à la création d'une nouvelle génération. Dans la littérature, nous avons rencontré deux styles différents de travaux :

- Dans le premier, une population intermédiaire de $(N/2)$ individus est sélectionnée parmi les N individus de la population initiale. Ces individus, appelés parents, forment une population intermédiaire sur laquelle est appliquée les opérateurs de croisement et de mutation afin de donner naissance à $N/2$ nouveaux individus appelés enfants. L'ensemble des parents et des enfants forme alors une population de N individus considéré comme la génération suivante (figure 2.2).

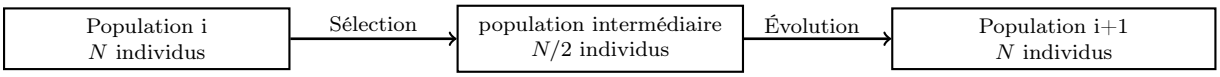


FIGURE 2.2 – Sélection puis Évolution

- La deuxième technique consiste à permuter entre les phases de sélection et d'évolution. En effet, les N individus de la population initiale donnent naissance à une population de N enfants qui sont concaténés directement dans la population. Une sélection est alors appliquée afin de revenir au nombre initial N individus (figure 2.3).

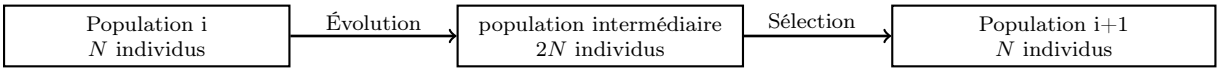


FIGURE 2.3 – Évolution puis Sélection

Dans un cas ou dans un autre, les individus peuvent être sélectionnés au moyen de plusieurs techniques que la littérature peut fournir :

- **Sélection aléatoire**

  La sélection la plus évidente et la plus simple à mettre en oeuvre est la sélection aléatoire des individus, indépendamment de leur évaluation. Si N est la taille de la population,chaque individu aura alors une probabilité de sélection uniforme égale à $(1/N)$. En général la convergence des AGs utilisant cette technique de sélection est assez lente.

- **Sélection par tournois**

  Cette technique de sélection s'effectue augmente les chances des individus de mauvaise qualité (mauvaise fitness), de participer à l'amélioration de la population. D'une manière générale, M individus sont pris au hasard parmi les N individus de la population. Ces M individus sont comparés entre eux et seul le meilleur individu est considéré comme vainqueur du tournoi et sera sélectionné pour l'application des opérateurs génétiques. Cette étape est répétée jusqu'à ce que la génération intermédiaire soit complétée. Il est tout à fait possible que certains individus participent à plusieurs tournois. S'ils gagnent plusieurs fois, ils auront donc le droit d'être copiés plusieurs fois dans la génération intermédiaire, ce qui favorisera la pérennité de leurs gènes.

  Le paramètre M est fixé a priori par l'utilisateur et joue un rôle important dans l'algorithme. En effet, si $M = N$, l'algorithme génétique est réduit à un algorithme de recherche locale travaillant sur une seul solution. L'inconvénient est de converger parfois, rapidement vers un optimum local. Si au contraire, M = 1, la sélection devient alors une sélection aléatoire.

- **Sélection par rang** La sélection par rang consiste à attribuer un rang dans la population à chaque individu en fonction de son évaluation. Pour un problème de maximisation par exemple, les individus sont classés selon un ordre croissant de la valeur de la fonction objectif. Le plus mauvais individu aura le numéro 1, et le meilleur individu aura le rang le plus élevé. Des probabilités de sélection $P(i)$ sont alors calculées pour chaque individu i en fonction de du rang selon la formule suivante :

$$\text{Probabilité de sélection}(Parent_i) = \frac{Rang_i}{\sum_{j \in Population} Rang_j}$$

Cette procédure est très simple et exagère le role du meilleur élément eu détriment des autres élément potentiellement exploitable. Le second, par exemple aura une probabilité d'être sélectionné plus faible que le premier, bien qu'il soit peut-être situé dans une région d'intérêt.

- **N/2-élitisme** Dans la sélection N/2-élitisme les individus sont tirés selon leur fonction de fitness. Seule la moitié supérieure de la population, correspondant aux meilleurs chromosomes, est sélectionnée.
- **Sélection par roulette** Dans un problème d'optimisation de maximisation, on associe à chaque individu i une probabilité de sélection, noté $prob_i$, proportionnelle à sa valeur F_i de la fonction objectif :

$$\text{Probabilité de sélection}(Parent_i) = \frac{F_i}{\sum_{j \in Population} F_j}$$

2.2.2.5 Opérateurs de Croisement

Le croisement a pour objectif d'enrichir la diversité de la population en introduisant de nouvelles solutions obtenues par combinaisons des solutions parents. Classiquement, un croisement est envisagé, avec une certaine probabilité communément appelée probabilité de croisement, sur deux parents et engendre deux enfants.
Initialement, le croisement associé au codage par chaînes de gènes est le croisement à découpage de chromosomes (slicing crossover). Pour effectuer ce type de croisement sur des chromosomes constitués de plusieurs gènes, on tire aléatoirement une position dans chacun des parents Parent 1 et Parent 2. On échange ensuite les deux sous-chaînes terminales de chacun des deux parents, ce qui produit deux enfants Enfant 1 et Enfant 2. Ce croisement est aussi appelé croisement en un point (figure 2.4). On peut étendre ce principe en découpant les chromosomes non pas en 2 sous-chaînes mais en 3, 4, etc (figure 2.5).

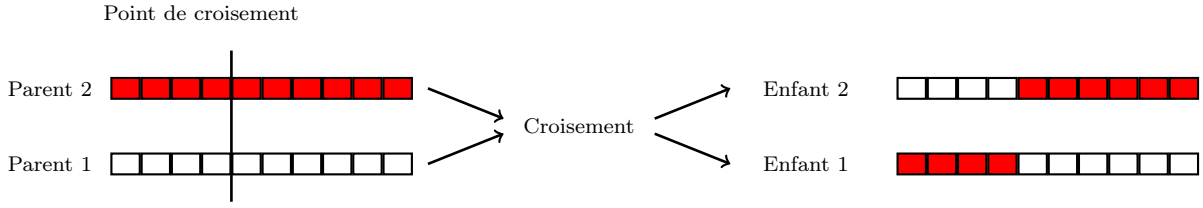


FIGURE 2.4 – Croisement en un point

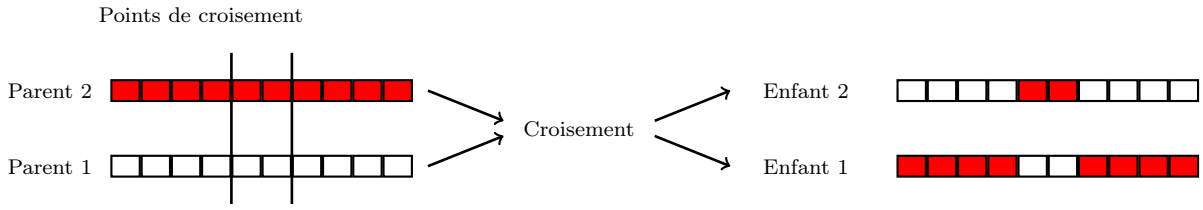


FIGURE 2.5 – Croisement en deux points

2.2.2.6 Opérateurs de Mutation

L'objectif majeur d'un opérateur de croisement est d'apporter à l'AG l'aléa nécessaire pour une exploration efficace de l'espace de recherche. Sa mise en oeuvre doit permettre à l'algorithme d'atteindre la plupart des sous-espaces de solution réalisables. En effet, la mutation joue le rôle d'un bruit et empêche l'évolution de se figer. Elle permet donc d'assurer une recherche complète aussi bien globale que locale de l'espace des solutions.
Les propriétés de convergence d'un AG sont fortement dépendantes de l'opérateur de mutation. En effet, dans [58], Cerf prouve qu'un AG peut converger vers une bonne solution sans croisement, rien qu'en utilisant un opérateur de Mutation.
L'opérateur de Mutation est utilisé avec une probabilité, appelée probabilité de mutation, généralement choisie faible. Initialement, si le codage utilisé des solutions est un codage binaire, la Mutation consiste à sélectionner aléatoirement un gène de l'enfant obtenu après croisement, et à le remplacer par son complément (voir figure 2.6). Ceci permet d'avoir des nouveaux gènes, donc des nouvelles caractéristiques, qui n'appartiennent à aucun des deux parents. Si l'Algorithme Génétique est à base d'un codage réel, la Mutation sur un enfant peut se faire en permutant deux gènes tirés aléatoirement (voir figure 2.7).

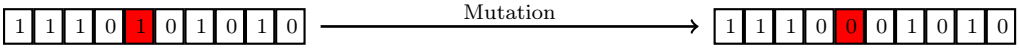


FIGURE 2.6 – Mutation binaire

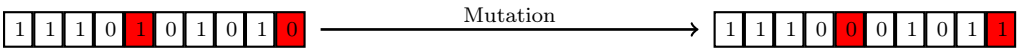


FIGURE 2.7 – Mutation par permutation

2.2.2.7 Paramétrage de l'Algorithme Génétique

La difficulté majeure des Algorithmes Génétiques ne réside pas dans la mise en oeuvre de l'algorithme lui même, mais plutôt, dans le choix des valeurs adéquates des différents

paramètres de ce dernier. Le réglage de ce paramètre est une question très délicate qui diffère d'un problème à un autre. Cette étape constitue une part très importante du travail de l'utilisateur qui doit les adapter à la nature du problème posé et de la fonction fitness à optimiser. Dans la pratique, les paramètres d'un algorithme génétique sont réglés approximativement par tâtonnement jusqu'à trouver une solution assez acceptable. La comparaison avec une autre méthode d'optimisation permettra de s'assurer du bon déroulement de l'algorithme.

Prenons par exemple la taille de la population : cette dernière agit fortement sur la rapidité de convergence de l'algorithme à trouver une bonne solution. En effet, une population trop petite évoluera probablement vers un optimum local qui risque d'être peu intéressent. Une population trop grande rendra le temps de calcul trop excessif.

De même, le test d'arrêt est déterminant dans un algorithme génétique du fait qu'il joue le rôle d'un outil d'évaluation de la qualité la solution retenue.
Les critères d'arrêts peuvent être de deux natures :

- Arrêt après un nombre fixé a priori de générations. Ce nombre doit être raisonnable et doit être choisi de telle façon à réaliser un bon compromis entre temps de calcul et la qualité de la solution trouvée.
- Arrêt dès que la population cesse d'évoluer ou n'évolue plus suffisamment. La population devient alors homogène, en espérant qu'elle se situe à proximité de l'optimum.

Nous avons au court de cette section établi les principes de base essentiels à la compréhension des Algorithmes Génétiques. Dans la littérature différentes variantes ont été proposées afin de les adapter à différents problèmes d'optimisation combinatoire. En effet, pour chacun des opérateurs de codage, de sélection, de croisement et de mutation, plusieurs possibilités peuvent être envisageables. Dans la prochaine section nous allons présenté une petite revue de littérature quand à l'adaptation des Algorithmes Génétiques pour la résolution des Problèmes d'Optimisation Multi-objectif.

2.3 Algorithmes Génétiques & Optimisation Multi-Objectif

Étant une approche basée sur une population de solution, les Algorithmes Génétiques représentent une méthode bien adaptée pour résoudre les problèmes d'Optimisation

Multi-Objectif. Un AG Mono-Objectif peut être modifié pour trouver un ensemble de multiples solutions non-dominées en un seul passage. Avec l'aptitude des AGs à explorer simultanément différentes régions d'un espace de solution, il est possible de trouver un ensemble diversifié de solutions à des problèmes difficiles non-convexes, discontinues, etc.

L'opérateur de Croisement dans cette Métaheuristique peut exploiter des structures de bons compromis par rapport à des objectifs différents pour créer de nouvelles solutions non dominées dans des régions non visitées pour calculer le front de Pareto. En outre, cette méthode ne nécessite pas à l'utilisateur d'établir des priorités, de l'échelle, ou peser les objectifs. Par conséquent, cette Métaheuristique est reconnue comme étant l'approche la plus populaire pour la conception et l'Optimisation des Problèmes Multi-Objectif.

Devant tous ces avantages, deux inconvénients majeurs doivent être abordés lorsqu'un AG est appliqué à l'Optimisation Multi-Objectif :

1. L'absence d'une relation d'ordre totale qui lie l'ensemble des solutions constituent la population à une étape donnée, ce manque apparaît dans la difficulté de concevoir un opérateur de Sélection qui affecte à chaque individu une probabilité de sélection proportionnelle à la performance de cet individu ;
2. La perte prématurée de la diversité ainsi que l'instabilité de la recherche. D'où la nécessité de concevoir des techniques de maintien de la diversité au sein de la population.

Dans ce qui suit, une catégorisation de techniques efficaces qui traitent ces inconvénients est présentée.

2.3.1 Adaptation des AGs à l'Optimisation Multi-Objectif

Pour une meilleure exploration de l'ensemble des solutions réalisable plusieurs techniques ont été développées afin d'augmenter la performances des AGs pour la résolution des problèmes Multi-Objectif. Parmi ces techniques nous trouvons celles qui améliorent la Sélection et d'autres techniques pour le maintien de la diversité de la population qui sont largement utilisées comme le *partage de la fitness* et le *surpeuplement* ou bien *crowding distance*. Une technique est aussi largement utilisée pour le maintien des bonnes solutions séparément des autres solutions afin de ne pas les perdre en cours d'exécution c'est bien l'élitisme.

2.3.1.1 Mécanismes de Sélection

Dans l'Optimisation Mono-Objectif, la fonction objectif et la fonction de Fitness sont identiques. En revanche pour l'Optimisation Multi-Objectif l'assignation de la valeur de Fitness et la Sélection doivent prendre en considération plusieurs objectifs. En effet l'absence d'une relation d'ordre total entre les différentes solutions possibles d'un problème Multi-Objectif, pour faire face à ce manque plusieurs techniques de Sélection ont été développées, un éventail de ces techniques est présenté dans ce qui suit :

1. **Sélection par somme pondérée des objectifs :** Cette technique est la plus connues dans la literature elle consiste à se ramener à un problème Mono-Objectif en combinant linéairement les différentes fonctions objectifs :

   $$f(x_i) = \sum_k \lambda_k f_k(x_i)$$

   Avec : $\lambda_k \in [0, 1]$ et $\sum_k \lambda_k = 1$.

   Avant d'appliquer la méthode de la somme pondérée, les objectifs doivent être normaliser, à cause les objectifs sont mesurer avec des unités différentes, la normalisation des objectifs se fait comme suit :

   $$f'_i = \frac{f_i - f_{i,min}}{f_{i,max} - f_{i,min}}, \qquad i = 1, 2, \ldots, m. \tag{2.1}$$

   Où :
   $f_{i,min}$: est la plus petite valeur du i^{ime} objectif dans la génération courante ;
   $f_{i,max}$: est la plus grande valeur du i^{ime} objectif dans la génération courante.

   Plusieurs techniques de calcule des poids à mettre sur les objectifs ont été proposées dans ce qui suit nous allons considérés deux techniques :

   Technique I : Dans cette stratégie proposée par Murata, Ishibuchi, et Tanaka (1996)[108], les poids sont déterminés aléatoirement pour chaque génération de l'Algorithme Génétique (voir équation 2.2).

   $$random_i \sim U(0, 1)$$

$$p_i = \frac{random_i}{random_1 + \ldots + random_m}, \qquad i = 1, 2, \ldots, m. \tag{2.2}$$

Cette approche explore entièrement l'espace des objectifs pour éviter la convergence prématurée en assurant une diversité des solution et en donnant des chances uniformes à toute recherche possible des point Pareto optimale toute au long de la frontière Pareto.

Technique II : Dans cette stratégie proposée par Zhou and Gen (1999) [109], les poids sont déterminés en fonction du point idéal calculé pour chaque génération de l'algorithme génétique(voir équation 2.3).

$$p_i = \frac{p'_i}{p'_1 + \ldots + p'_m}, \ldots, p_m = \frac{p'_m}{p'_1 + \ldots + p'_m}, \qquad i = 1, 2, \ldots, m. \tag{2.3}$$

Où :

– $p'_1 = f'_1(x) - f'_{1,min}(x), \ldots, p'_m = f'_m(x) - f'_{m,min}(x)$.

– $f'_{1,min}, \ldots, f'_{m,min}$ sont les coordonnées du point idéal dans la génération courante.

Cette approche explore particulièrement les zones les plus prometteuse (point idéal) dans l'espace des objectifs avec le but de converger le plus rapidement possible vers la frontière Pareto.

2. **Sélection NSGA (Non Dominated Sorting Genetic Algorithm) :** Cette technique été proposée par Srinivas et Deb en 1995 [67], dans la sélection NSGA les rangs des individus sont calculés d'une manière récursive à commencer par les individus dominants de la population.

- Soit E_1, l'ensemble des individus non dominés de la population courante.
- Soit E_2, l'ensemble des individus qui ne sont dominés que par des individus appartenant à E_1.

.....

- Soit E_k, l'ensemble des individus qui ne sont dominés que par des individus appartenant à $E_j, j < k$.

La probabilité de sélection d'un individu x_i appartenant à E_n suit l'expression de Baker (Talbi, 1999) suivante :

$$\pi(x_i) = \frac{S(N+1-R_n)+R_n-2}{N(N-1)} ;$$
$$R_n = 1 + \mid E_n \mid + 2 * \sum_{j \in [1..n-1]} \mid E_j \mid .$$

Avec :

– N : la taille de la population courante ;

– S : représente la pression de sélection. Une pression de sélection S égale à 1 donne la même probabilité de Sélection pour tous les individus ($p_n = 1/N$). Pour des valeurs de S supérieures à 2, les individus de dernier rang peuvent avoir des probabilités de sélection négatives, ce qui est équivalent à une probabilité nulle. Dans ce cas, ils ne seront jamais sélectionnés.

3. **Sélection WAR(Weighted Average Ranking) :** dans cette technique proposée par Bentley et Wakefield en 1997 [69], l'idée consiste à calculer le rang de chaque individu de la population par rapport aux différents objectifs séparément. Le rang d'un individu est alors pris comme étant la somme des rangs. Pour cela on commence par ordonner les individus par ordre croissant de la fonction f1 et par ordre croissant de la fonction f2 etc.
   Les probabilités de sélection sont alors calculées comme pour la sélection NSGA.

2.3.1.2 Maintenir la diversité de la population

Les AGs classiques sont réputés pour être très sensibles quant au choix de la population initiale ainsi qu'aux mauvais échantillonnages lors de la Sélection. Cette fragilité est observable sur le plan de la perte de diversité ou ce qu'on appelle aussi la dérive génétique. Pour palier à cet inconvénient plusieurs techniques visant à maintenir la diversité dans la population ont été proposées, les plus fréquemment utilisées sont brièvement résumé ici :

1. **La fonction partage (Sharing) :** Le principe du Sharing proposé par Goldberg Richardson en 1987 [59] est la dégradation de l'adéquation des individus appartenant à des espaces de recherche de forte concentration de solutions, autrement dit la fonction d'adaptation de chaque individu (xi) est dégradée par un compteur de niche $m(x_i)$, calculé pour ce même individu. Le compteur de niche calcule le degré de similarité qu'a un individu avec le reste de la population :

$$m(x_i) = \sum_{j \in pop} sh(d(x_i, x_j))$$

   où :
   – $d(x_i, x_j)$: distance entre l'individu i et j ;
   – sh_d : fonction décroissante de $d(i, j)$, tel que : $sh(0) = 1$ et et $sh(d \geq \sigma_{share}) = 0$. La fonction $sh(d)$, la plus communément utilisée, est la fonction triangulaire définie comme suit :

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}}) & \text{si } d < \sigma_{share}, \\ 0 & \text{sinon.} \end{cases}$$

Avec

σ_{share} : rayon de niche, fixé dans la plupart des cas par l'utilisateur en fonction de la distance minimale de séparation voulue entre les différents pics.

La nouvelle fonction de partage calculée ***shared fitness*** est obtenue en divisant la fonction d'adaptation de l'individu par le compteur de niche comme suit :

$$f'(x_i) = \frac{f(x_i)}{m(x_i)}.$$

Généralement la méthode précédente de la dégradation des fonctions des individus par le compteur de niche shared fitness n'est plus prise en compte. Ce qui compte le plus, c'est le compteur de niche de chaque individu. Entre deux individus qui sont ni dominés, ni non dominés, l'individu ayant le plus petit compteur de niche est sélectionné, de la sorte il est possible de maintenir une diversité le long du front de Pareto. La figure 2.8 montre l'exemple de deux individus (candidats) non dominés, en considérant une maximisation selon l'axe des x et une minimisation selon l'axe des y. Dans cette figure, les deux candidats (1 et 2) pour la sélection ne sont pas dominés par l'ensemble de comparaison. Pour maintenir la diversité le long du front de Pareto, le candidat qui a le plus petit compteur de niche est sélectionné, soit le candidat numéro 2.

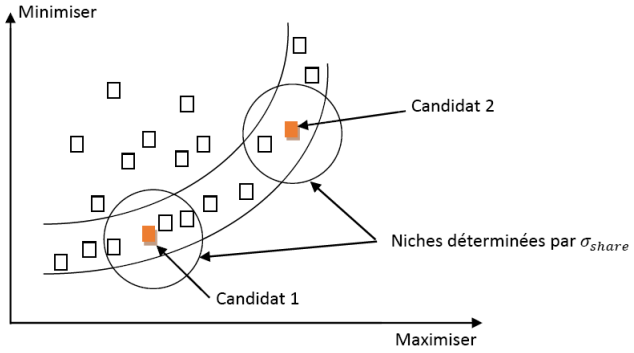


FIGURE 2.8 – Sélection d'individus avec le NPGA

2. **Distance de surpeuplement (crowding distance) :** La distance de crowding d'une solution x_i (ou un individu) se calcule en fonction du périmètre formé par les

points les plus proches de x_i sur chaque objectif. La figure 2.9 montre une représentation à deux dimensions associée à la solution x_i. Le calcul de la distance de crowding nécessite, avant tout, le tri des solutions selon chaque objectif, dans un ordre ascendant. Ensuite, pour chaque objectif, les individus possédant les valeurs limites (la plus petite et la plus grande valeur de fonction objectif) se voient associer une distance infinie (∞). Pour les autres solutions intermédiaires, on calcule une distance de crowding égale à la différence normalisée des valeurs des fonctions objectifs de deux solutions adjacentes. Ce calcul est réalisé pour chaque fonction objectif. La distance de crowding d'une solution est calculée en sommant les distances correspondantes à chaque objectif. Une fois toutes les distances sont calculées, il ne reste plus qu'à les trier par ordre décroissant et à sélectionner les individus possédant la plus grande valeur de crowding.

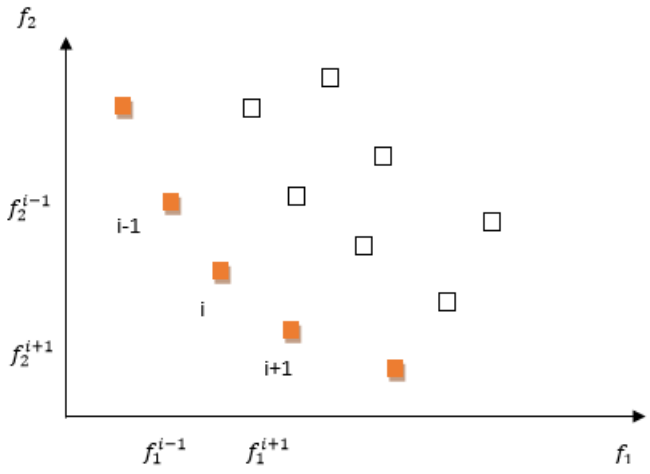


FIGURE 2.9 – Les points coloriés sont des solutions appartenant au même front

L'algorithme 1 montre la procédure de calcul de la distance de toutes les solutions non dominées de l'ensemble I. Dans cet algorithme, f_m^{i+1}, f_m^{i-1} représentent respectivement la valeur de la $m'ime$ fonction objectif de la solution $i+1$ et $i-1$, alors que les paramètres f_m^{Max} et f_m^{Min} représentent les valeurs maximale et minimale de la $m'ime$ fonction objectif. Après ce calcul, toutes les solutions de I auront une distance métrique :

Algorithm 1 Calcul de la distance de surpeuplement.

$l = |I|$ Nombre de solution dans l'ensemble I

Pour chaque i, poser $I[i]_{distance} = 0$ Initialiser les distances

Pour chaque objectif m

$I = Trier(I, m)$ Trier selon la valeur de l'objectif m

- $I[1]distance = \infty$
- $I[l]distance = \infty$
- For $i = 2$ to $(l - 1)$

  $I[i]distance = I[i]distance + ((f_m^{i+1} - f_m^{i-1})/(f_m^{Max} - f_m^{Min}))$

L'opérateur crowded-comparison ($\prec_n$) est utilisé pour guider le processus de sélection comme suit : chaque solution (i) de la population est identifiée par son rang (i_{rang}) et la distance de crowding ($i_{distance}$). L'opérateur ($\prec_n$) défini ci-dessous permet d'identifier un ordre de préférence entre deux solutions :

$i \prec_n j$ si ($i_{rang} < j_{rang}$)

ou (($i_{rang} = j_{rang}$) et ($i_{distance} > j_{distance}$))

Entre deux solutions de rangs différents, on préfère la solution avec le plus petit rang (ou le plus petit front). Pour deux solutions qui appartiennent au même front, on préfère la solution qui est localisée dans la région où la densité de solutions est moindre, soit l'individu possédant la plus grande valeur de distance de crowding.

2.3.1.3 L'élitisme

Lors de l'application des opérateurs génétiques (Croisements et Mutations), il se peut que le meilleur individu, l'élite, soit endommagé de façon irréversible. Ce problème peut également se produire pour certaines méthodes de sélection, comme par exemple la roulette ou le tournoi. Le meilleur élément est ainsi perdu et ne sera peut-être jamais reconstruit. Afin d'éviter ce problème, la notion d'élitisme a été introduite en 1975 par De Jong [60] où il a suggéré de toujours inclure le meilleur individu de la population P_t dans la population P_{t+1} dans le but de prévenir sa perte due aux effets d'échantillonnage ou aux perturbations des opérateurs. Cette stratégie, qui peut être étendue en copiant les n meilleures solutions à la génération suivante, est appelé ***élitisme***.

2.3.2 Revue de littérature des Algorithmes Génétiques Multi-Objectif

Comme nous l'avons déjà précisé auparavant le fondement des AGs sur la manipulation d'une population de solutions simultanément, ceci fait que les AGs représente une méthode bien adaptée pour résoudre les problèmes d'Optimisation Multi-Objectif, Un AG Mono-Objectif peut être modifié pour trouver un ensemble de multiples solutions non-dominées en un seul passage. Avec l'aptitude des AGs à explorer simultanément différentes régions d'un espace de solution, il est possible de trouver un ensemble diversifié de solutions à des problèmes difficiles, non-convexes, discontinues, etc.

Les premiers AGs Multi-Objectif qui ont été développés sont non-élitistes, c'est à dire qu'ils n'utilisent pas le principe d'archivage pour conserver les solutions meilleures durant le processus d'optimisation. Ces algorithmes comprennent entre autres le MOGA (Multi-Objective Genetic Algorithm) de Fonseca et Fleming en 1993 [73], le NSGA (Non-dominated Sorting Genetic Algorithm) de Srinivas et Deb en 1994 [56], et le NPGA (Niched-Pareto Genetic Algorithm) de Horn en 1994 [74]. Ils mettent en oeuvre le concept d'optimalité de Pareto pour classer les solutions en différents groupes de solutions non dominées, en vue d'aider à la convergence des algorithmes. Pour assurer la diversité de la population dans la région de Pareto, ces algorithmes utilisent la technique de la fonction de partage dont on sait que le choix du paramètre de réglage σ_{share} conditionne les performances des algorithmes. Ces algorithmes ne sont plus utilisés à cause de leur convergence lente et la difficulté de maintenir adéquatement une bonne diversité dans la population.

Plus tard, une autre génération des AGs Multi-objectif a vu le jour en intégrant le concept d'élitisme. Il s'agit entre autres de l'algorithme SPEA Strength Pareto Evolutionary Algorithm de Zitzler et Thiele en 1999 [75] qui utilise une méthode de regroupement des solutions. C'est une méthode qui permet de distribuer efficacement les solutions sur la frontière de Pareto. Cependant, pour des problèmes présentant plusieurs fronts de Pareto locaux, le clustering conduit à un élitisme extrême. Knowles et Corne en 2000 ont proposé l'algorithme PAES Pareto-Archived Evolution Strategy [76] qui s'inspire de l'algorithme $(\mu + \lambda)$-Evolution stratégie [77] où μ et λ sont respectivement le nombre de parents et d'enfants. Il inclut une population auxiliaire appelée archive. Pour promouvoir la diversité dans la population, le PAES utilise une technique basée sur un découpage en

grilles de l'espace des fonctions objectif. Cette méthode est efficace quand la distribution des solutions dans l'espace de recherche n'est pas uniforme. L'efficacité de la méthode dépend cependant du choix du paramètre de discrétisation de l'espace des fonctions objectif. Le NSGA-II de Deb en 2000 [78] est une version améliorée du NSGA développé antérieurement et qui n'utilisait pas le concept d'archive. Ainsi, le NSGA-II maintient les solutions non dominées dans une archive pour les utiliser dans les futures générations. Cet algorithme applique la méthode du surpeuplement pour promouvoir la diversité dans la population tout en ainsi de réduisant la complexité de l'algorithme, de créer une méthode élitiste et de supprimer l'utilisation de la fonction de partage.

Il existe bien d'autres algorithmes développés par la suite avec pour souci d'améliorer d'avantage les Algorithmes Évolutionnaires existants. Il s'agit par exemple de l'algorithme PESA (Pareto Envelope Based Selection Algorithm) de Corne et al. en 2000 [79]. Il reprend le principe de division en grilles utilisé dans le PAES et définit un paramètre appelé facteur de compression ("squeeze factor"). Le squeeze factor représente la mesure du surpeuplement d'une zone de l'espace. Le PESA est très élitiste et performe moins bien avec les problèmes discrets ou présentant plusieurs optima. Cet algorithme ne garantit pas non plus que les solutions Pareto optimales disposées aux extrémités du front de Pareto soient gardées dans l'archive à chaque génération. Bien qu'il ait une vitesse de convergence élevée, le PESA convient seulement aux problèmes continus et simples. Le NSGA-II avec contrôle de l'élitisme proposé par Deb et Goel en 2001 [80] est une autre version du NSGA-II qui vise à assurer une diversité de la population dans tout l'espace faisable. Zitzler et al en 2001 [81] ont également proposé une version révisée du SPEA, qu'ils ont nommé le SPEA2. Cette version calcule la densité de la population autour d'un individu dans l'espace des solutions et l'utilise pour corriger l'expression de sa fonction d'adaptation. Pour assurer la diversité de la population sur le front de Pareto, un opérateur de troncature est utilisé et permet de préserver les solutions aux extrémités du front de Pareto. Le PESA-II de Corne et al en 2001 [82] adopte une sélection basée sur les zones de l'espace occupées par les individus contrairement aux autres algorithmes qui utilisent la fonction d'adaptation des individus. Après avoir sélectionné une zone, un individu de celle-ci est choisie aléatoirement. Cette méthode répartit efficacement les solutions sur la frontière de Pareto à cause de sa capacité à choisir, avec une plus grande probabilité que le tournoi classique, les individus situés dans des zones moins denses. Bien que cette technique ait permis de faire évoluer positivement la sélection de manière à

privilégier les zones de l'espace les moins surpeuplées, elle dépend fortement du réglage du paramètre de discrétisation de l'espace de recherche.

Dans la pratique, l'optimisation des problèmes d'ingénierie rencontre diverses difficultés liées notamment aux multiples contraintes dont il faut tenir compte, aussi au nombre et la nature des objectifs à considérés, devant ces difficultés les Algorithmes Génétiques multi-objectif ont connus une application intense dans le domaine de l'ingénierie pour aider à concevoir des systèmes opérant dans des conditions optimales. Le développement des Algorithmes Génétiques Multi-Objectif suscitent de nos jours un grand intérêt et différents travaux ont été présentés dans la littérature qui proposent des algorithmes d'optimisation généralement adaptés pour certains types de problèmes. Leur utilisation pour traiter d'autres problèmes peut donner des résultats non satisfaisants, ou peut requérir des modifications majeures qui ne sont pas toujours à la portée de l'utilisateur.

2.4 Conclusion

Nous avons présenté au cours de ce chapitre les Algorithmes Génétiques, leurs historique, leurs principes de base, aussi les différentes techniques permettant d'adapter ces algorithmes pour la résolution de différents types de problèmes d'optimisation combinatoire, et finalement nous avons résumés quelques travaux présentés pour la résolution des problèmes d'optimisation multi-objectif en utilisant les Algorithmes Génétiques.

Nous proposons dans le chapitre prochain une adaptation d'un Algorithme Génétique pour résoudre un problème d'optimisation multi-objectif d'une chaîne logistique globale, en comparent plusieurs techniques de sélection, l'Algorithme Génétique adapté sera aussi doté d'une technique de maintien de la diversité et de l'élitisme. Les différentes phases de l'AG développé sont décrites dans le chapitre suivant.

Chapitre 3

Optimisation Multi-Objectif & Chaînes Logistiques

3.1 Introduction

Dans ce chapitre nous présentons les Chaînes Logistiques : leurs définitions, leurs conceptions, leur gestion, et de la prise de décisions aux différents niveaux de la chaîne, nous présentons les travaux en relation avec le management et l'Optimisation des Chaînes Logistiques. Nous nous intéressons aussi à un problème d'Optimisation Multi-Objectif dans une Chaîne Logistique globale en adoptons une modélisation de ce problème, et nous terminerons par l'adaptation d'un Algorithme Génétique avec quatres variantes différentes pour la résolution de ce problème reconnu pour être NP-difficile.

3.2 Chaîne Logistique concepts de base et définitions

Dans un contexte économique instable, sous la pression de la globalisation, d'une concurrence croissante, nombreuses sont les entreprises qui constatent les limites de l'optimisation seule de leurs systèmes de production et cherchent à explorer de nouvelles sources de compétitivité à travers l'optimisation de leurs chaînes logistiques et de la relation avec leurs partenaires. Fournir le produit et/ou le service désiré par le client, rapidement, moins cher et plus performant que celui proposé par l'entreprise concurrente sur le marché est de nos jours le souci majeur de chaque entreprise existant dans un marché local et/ou international. La concurrence dans un futur proche ne sera pas entre différentes entreprises mais entre différentes chaînes logistiques (Supply Chain).

Le terme logistique ou gestion des flux vient du mot grec **"logistike"** qui signifie l'art du raisonnement et du calcul. La logistique est apparue en premier lieu dans un contexte militaire où elle définit l'ensemble des techniques mises en oeuvre pour assurer l'approvisionnement, et le maintien en conditions opérationnelles des troupes armées en temps de guerre. Après la logistique militaire vient la logistique industrielle, celle-ci repose plus particulièrement sur les activités de soutien à la production. Elle est apparue à la fin de la seconde guerre mondiale. Le concept de logistique a évolué depuis, avec l'évolution du marché et des systèmes industriels.

3.2.1 Définitions

Nombreuses sont les définitions de la chaîne logistique qui ont été proposées dans la littérature, en 1999 Tayur, Ganeshan et Magazine [83] définissent la chaîne logistique par *« la chaîne logistique est un réseau composé de sous-traitants, de producteurs, de distributeurs, de détaillants et de clients entre lesquels s'échangent des flux matériels dans le sens des fournisseurs, et des flux d'information dans les deux sens».*
Une autre définition a été proposée par Ganeshan et Harisson en 1995 [84] *«Une chaîne logistique est le réseau des moyens de production et de distribution qui assurent les tâches d'approvisionnement en matières premières, la transformation de ces matières premières en produits semi finis et en produits finis, et la distribution de ces produits finis aux clients».*

3.2.2 Fonctions de la Chaîne Logistique

Depuis la deuxième définition de la chaîne logistique, nous pouvons déduire que les fonctions d'une chaîne logistique commencent par l'approvisionnement par les matières premières et allant jusqu'à la vente des produits finis en passant par la production, le stockage et la distribution. Ces différentes fonctions seront brièvement résumé comme suit :

- **Le processus Approvisionnement :** La fonction approvisionnement est la fonction la plus en amont de la chaîne logistique, ce processus se concentre sur la fourniture de tous les composants nécessaires à la fabrication. Il s'agit d'abord de sélectionner les fournisseurs de l'entreprise, ce choix peut se faire sur différents critères comme la qualité, le prix et/ou les délais de réapprovisionnement des matières premières. Une fois les fournisseurs déterminés, la seconde phase du processus approvisionnement consiste à passer les commandes des composants à ces fournisseurs en fonction de la production à réaliser, et de vérifier que ces composants sont livrés dans de bonnes

conditions. Le processus approvisionnement regroupe ainsi toutes les relations avec les fournisseurs pour assurer les niveaux de stocks en composants nécessaires et suffisants pour la fabrication.

- **Le processus Production :** La fonction de production regroupe les compétences dont dispose l'entreprise pour fabriquer, développer ou transformer les matières premières en produits finis, elle donne quelle capacité a la chaîne logistique pour produire et ainsi un indice sur sa réactivité face aux demandes du marché. Les méthodes utilisées pour la gestion de la production cherchent à améliorer le flux des produits dans les ateliers de fabrication à travers la planification, l'ordonnancement et la détermination de la taille optimale des lots de production.
- **Le processus Stockage :** La fonction stockage commence par le stock des matières premières, des composants, puis finalement le stock des produits finis. Il est indispensable lors de ce processus de trouver l'équilibre entre une meilleure réactivité et la réduction des coûts, car avoir des stocks engendre des coûts et des risques surtout dans le cas de produits périssables. La gestion des stocks est l'une des clés de la réussite et l'optimisation de toute une chaîne logistique.
- **Le processus Distribution :** La fonction distribution concerne la livraison des produits finis aux clients, et reprend les questions d'optimisation des réseaux de distribution c'est à dire l'organisation et le choix des moyens de transport. Les coûts de transport et distribution constituent le tiers des coûts opérationnels globaux d'une chaîne logistique, ce qui rend leur optimisation un défi majeur pour l'entreprise
- **Le processus Vente :** La fonction de vente est la fonction ultime dans une chaîne logistique, si on a bien optimisé pendant les étapes précédentes alors on facilite la tâche du personnel commercial, car ils pourront offrir des prix plus compétitifs aux clients. Ce processus de l'entreprise est également chargé de définir la demande prévisionnelle et d'intégrer des aspects commerciaux comme la durée de vie du produit pour anticiper l'évolution de ses ventes.

3.2.3 Les décisions dans les Chaînes Logistiques

La conception d'une chaîne logistique nécessite de prendre un ensemble de décisions, le plus souvent, l'architecture décisionnelle d'une entreprise est divisée en trois niveaux : stratégique, tactique et opérationnelle correspondant respectivement à des horizons à long, moyen et court terme.

- **Les décisions stratégiques :** Ces décisions normalement prises par la direction

générale de l'entreprise, définissent la politique de l'entreprise sur le long terme, comme par exemple la recherche de nouveaux partenaires industriels, les décisions d'ouverture ou de fermeture de certains sites de production ou leur délocalisation, ou encore le choix des moyens de transports entre les différentes localisations.

- **Les décisions tactiques :** Ces décisions sont généralement prises à moyen terme par les cadres de la production et les chefs d'atelier, il s'agit de produire au moindre coût pour les demandes prévisibles, donc avec connaissance des ressources matérielles et humaines et de faire la planification dépendant de la structure conçue au niveau stratégique.
- **Les décisions opérationnelles :** Ces décisions, prises par les chefs d'équipe et éventuellement les opérateurs de production, ont une portée plus limitée dans le temps (décisions prises à très court terme) pour assurer la gestion des moyens et le fonctionnement au jour le jour de la chaîne logistique. L'objectif à ce niveau est de répondre aux requêtes des clients de façon optimale en respectant les politiques de planification choisies aux niveaux stratégique et tactique.

3.2.4 Modélisation des Chaînes Logistiques

La chaîne logistique étant un système complexe et dynamique, ajouté à cela un environnement instable qui génère de nombreuses incertitudes. La difficulté de la prise en compte de ces incertitudes fait que la plupart des modèles proposés pour modéliser les chaînes logistiques utilisent des hypothèses restrictives, et parfois simplistes [85]. La modélisation de ces systèmes complexes permet une meilleure compréhension et une meilleure gestion de ces systèmes. Un modèle n'est qu'une représentation simplifiée d'un système réel, qui permet de l'analyser, le contrôler et le piloter. Ils sont à la base des systèmes d'aide à la décision. Nous allons voir trois types de modélisations : modèles conceptuels, modèles mathématiques, et modèles par simulation [86].

- **Les modèles conceptuels :** Les modèles conceptuels sont de loin les plus simples. Il s'agit en fait d'une description basique d'un système économique comme la chaîne logistique qui peut s'exprimer sous formes de diagrammes ou d'explications verbales. Le format utilisé dépend en grande partie de l'expérience du modélisateur, ceux avec une grande expérience font des diagrammes détaillés pour réduire l'ambiguïté, tandis que ceux avec une moindre expérience se basent sur une analyse par scénario. Dans ces modèles, il faut trouver un bon équilibre entre précision et aisance de communication. Ces modèles sont limités car difficiles à mettre en oeuvre dans le cas d'organisations

très complexes, et surtout ils ne donnent pas d'orientations quant au contrôle et au pilotage de la chaîne.

- **Modèles mathématiques :** Les modèles mathématiques sont très utilisés pour la conception des chaînes logistiques et pour l'optimisation des coûts. Ils consistent à modéliser un système réel par un ensemble d'équations exprimant les contraintes et les objectifs. Contrairement aux modèles conceptuels qui eux aident seulement à la compréhension du système, les modèles mathématiques résolvent les problèmes d'optimisation. Une autre différence avec les modèles conceptuels est que l'utilisation des modèles mathématiques requiert des compétences spéciales dans les mathématiques et la recherche opérationnelle. L'une des techniques les plus utilisées est la programmation linéaire et la programmation dynamique. Ces outils de recherche opérationnelle sont à la base de beaucoup de systèmes d'optimisation des supply chain management. L'inconvénient avec les modèles mathématiques est qu'ils font des restrictions trop importantes sur certaines hypothèses. Un autre inconvénient, et pas des moindres, est le temps d'exécution nécessaire pour résoudre des problèmes de tailles réalistes. Ainsi, pour des problèmes avec de grandes tailles, les industriels préfèrent utiliser des solutions approchées obtenues dans des délais raisonnables.
- **Modèles par simulation :** Les modèles par simulations sont très pratiques dans le cas de systèmes où il est difficile de représenter toutes les hypothèses par des équations, et de ce fait, on ne peut pas utiliser les modèles mathématiques. Ces modèles essayent d'imiter le comportement des composants d'un modèle et donc de pouvoir faire des prévisions et des évaluations de performances. Ils ont la capacité de capturer les incertitudes et de traiter l'aspect dynamique des systèmes complexes et des systèmes à grandes échelles. De nombreux modèles ont été proposés pour la simulation et la modélisation des chaînes logistiques. Hermann et al [87] proposent un nouveau cadre de simulation et des modèles hiérarchiques pour capturer les activités spécifiques au sein de la chaîne logistique. Dans sa thèse, Ding [88] propose une approche d'optimisation basée sur la simulation pour la conception des chaînes logistiques appliquée à l'industrie automobile et textile. Beaucoup de travaux de recherches se sont intéressés à ces modèles, nous pouvons citer les travaux de Jain en 2001 [89] et Bhasharan 1998 [90].

3.2.5 Conception et Gestion des Chaînes Logistiques

Une chaîne logistique est un ensemble d'équipements, de fournitures, les clients, les produits et les méthodes de contrôler les processus d'approvisionnement, de stockages et de distribution. Une chaîne logistique relie les fournisseurs et les clients, en commençant par la production de matières premières par un fournisseur, et se terminant par la consommation d'un produit par le client.

La conception et la gestion des chaînes logistiques est un sujet qui a gagné beaucoup d'importance dans le domaine de la gestion dans les dernières années en raison de l'augmentation de la compétitivité mis en place par la mondialisation des marchés. Dans de tel environnement les entreprises sont tenues de maintenir des niveaux de service élevés à la clientèle tout en même temps ils sont obligées de réduire les coûts et maintenir les marges bénéficiaires. Traditionnellement, les opérations de marketing, de distribution, de planification, de fabrication et d'approvisionnement sont traitées de façon indépendante par les décideurs même si les chaînes d'approvisionnement ont des objectifs contradictoires. Pour surmonter ces contradictions entre les différents opérations d'une même chaîne logistique les décideurs sont tenus de fixer un mécanisme où ces dernières peuvent être intégrées ensemble. La gestion de la chaîne d'approvisionnement est une stratégie par laquelle une telle intégration peut être réalisée.

Le problème de gestion des chaînes logistiques est l'un des problèmes de décision stratégiques, il représente une tâche difficile en raison de la complexité intrinsèque des principaux sous-systèmes impliqués et les nombreuses interactions entre ces sous-systèmes, ainsi que des facteurs externes tels que la diversité des fonctions objectifs considérées [91]. C'est une tache où les décisions, nombre, emplacement et capacité des usines et des centres de distribution (CDs) à installer sont prises, les réseaux de distribution sont établit et les quantités de matière première et demande clients sont calculées, etc.
Cette complexité a forcé la plupart des recherches dans ce domaine pour se concentrer sur les sous-systèmes individuels de la chaîne logistique. Récemment, toutefois, l'attention s'est convertie de plus en plus sur la performance, la conception et la gestion de la chaîne logistique dans son ensemble (chaîne logistique globale).

Les mesures de performances des chaînes logistique sont classés comme qualitative et quantitative. La satisfaction du client, la flexibilité et la gestion efficace des risques appartiennent à des mesures de performance qualitative. Les mesures de performance quantitatifs sont également classés en deux classes, la première englobe les objectifs basés directement sur le coût ou de profit, comme la réduction des coûts, la maximisation des ventes, la

maximisation du profit, etc. La deuxième classe contient les objectifs basés sur une mesure de la réactivité du client tels que remplissage maximisation du taux, la minimisation des délais de réponse des clients, minimisation des temps de livraison, etc.[92].

Ces mesures de performances ont été intégrées dans des approche traditionnelle séparément c.à.d considérant un seul objectif, comme la minimisation des coûts ou maximisation du profit. Par exemple, les auteurs dans [93, 94, 95, 96, 97, 98, 99, 100] ont estimé le coût total de la chaîne logistique comme une fonction objective dans leurs études. Cependant, pour une conception ou gestion plus réaliste et plus performante ces mesures doivent être intégrées ensemble c.à.d. considérant plusieurs objectifs conflictuels. Récemment beaucoup sont les chercheurs qui se sont intéressés à cette approche pour la conception et la gestion de la chaîne logistique, Sabri et Beamon (2000) dans [101] ont développé un modèle multi-objectifs pour la planification stratégique et opérationnelle d'une chaîne logistique sous incertitudes de la demande. Où le coût, taux de remplissage, et la flexibilité ont été considérés comme objectifs, pour la résolution de ce dernier ils ont adapté la méthode e-contrainte. Chan et Chung (2004) dans [102] ont proposé une procédure d'Optimisation Génétique Multi-Objective pour le problème de distribution dans un chaîne logistique, Ils ont considéré la minimisation du coût total du système, nombre total de jours de prestation et l'équité du taux d'utilisation des capacités des fabricants comme objectifs. D'autres chercheurs se sont intéresser pour l'optimisation des chaînes logistiques avec des modèles stochastiques comme Guillen et al (2005)[103] où ils ont formulé le problème de conception de d'une chaîne logistique comme un modèle stochastique de programmation linéaires multi-objectif avec des variables mixtes et entière, qui a été résolu par la méthode e-contrainte, et la méthode évaluation séparation, les objectifs étaient maximiser le profit sur l'horizon de temps et le niveau de satisfaction de la clientèle.

3.3 Adaptation d'un Algorithme Génétique Multi-Objectif pour l'Optimisation d'une Chaîne Logistique

3.3.1 Description du problème

Considérons une chaîne logistique globale d'une entreprise spécialisée dans la production d'un seul type de produit (produit unique). Cette chaîne logistique est constituée d'un ensemble de fournisseurs qui disposent de la matière première nécessaire pour la production,

d'un ensemble d'usines de production potentielles à installer et d'un ensemble de Centres de Distribution (CDs) potentiels à ouvrir. Chacun de ces usines (resp. CDs) a une capacité de production (resp. de stockage) et son installation engendre un coût fixe pour l'entreprise. Pour des raisons stratégiques l'entreprise décide de ne pas installer ou ouvrir plus d'un nombre maximal d'usines et de dépôts (ce nombre est fixé par le décideur au préalable).
L'architecture de cette chaîne logistique est organisée en trois niveaux Usine-Fournisseur, Usine-Dépôt et Dépôt-Client :

- Niveau 1 (Usine-Fournisseur) : Ce stage assure, pour la production, l'achat et le transport de la matière première nécessaire pour chaque usine.
- Niveau 2 (Usine-Dépôt) : Ce stage assure la production et le transport du produit fini depuis les usines au dépôts.
- Niveau 3 (Dépôt-Client) : Ce stage assure le stockage et la distribution du produit pour la satisfaction des demandes des clients.

Tout ces systèmes coopèrent ensemble pour assurer la satisfaction de la demande des clients exprimée par la quantité du produit dont le client a besoin et elle est connue à priori.
L'objectif est de concevoir une platform efficace pour la gestion optimale de la chaîne logistique, c.à.d déterminer l'ensemble des usines à installer et des dépôts à ouvrir et construire un réseau de distribution respectant les capacités des ressources et satisfaisant les demandes exigées par les clients.
Généralement dans la réalité plusieurs objectifs conflictuels interviennent dans l'évaluation de l'efficacité d'une chaîne logistique, dans notre problème nous considérons trois objectifs à optimiser, minimisation du coût total d'utilisation des ressources, maximisation du taux de satisfaction de la clientèle en termes de délai de livraison imposé et la maximisation de l'équilibre d'utilisation des capacités des ressources (CDs et Usines) en considérons les hypothèses suivantes :

- Les ensembles des clients, Dépôts, Usines et Fournisseurs sont bien définis.
- Le nombre maximal des Usines qui peuvent être installer est connu.
- Le nombre maximal des Dépôts qui peuvent être ouvrir est connu.
- Un client ne peut être servi que par un seul dépôt.
- La demande de chaque client est connue au préalable.
- Chaque dépôt a une capacité de stockage limitée et son fonctionnement engendre un coût fixe pour l'entreprise.
- Chaque usine a une capacité de production limitée et son fonctionnement engendre un coût fixe pour l'entreprise.

– L'achat et le transport de la matière première sur le stage 1 engendrent un coût unitaire pour l'entreprise.
– La production et le transport du produit finis sur le stage 2 engendrent un coût unitaire pour l'entreprise.
– Le stockage et la distribution du produit finis sur le stage 3 engendrent un coût unitaire pour l'entreprise.

La figure 3.1 résume une chaîne logistique d'une telle entreprise :

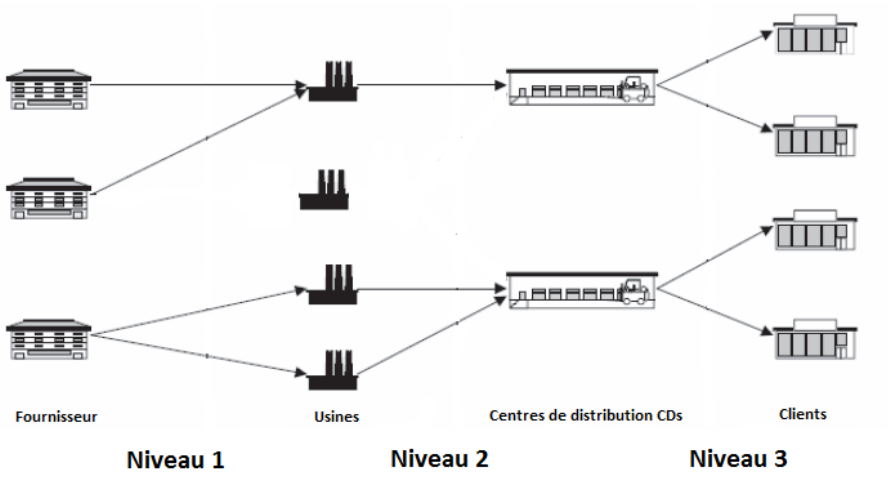


FIGURE 3.1 – Illustration d'une Chaîne Logistique Globale

3.3.2 Modélisation mathématique

Différentes sont les modélisations mathématiques proposées dans la literature pour mieux représenter le problème d'optimisation dans les chaînes logistique, mais rares sont travaux qui proposent une modélisation globale considérant les différents niveau de ce problème i.e. traiter les trois niveaux en même temps, dans ce travail nous considérons une modélisation proposée par Fulya Altiparmak et al [104] :

3.3.2.1 Les indices du modèle

– i est un indice pour les clients ($i \in I$).
– j est un indice pour les centres de distribution CDs ($j \in J$).
– k est un indice pour les usines de fabrication ($k \in K$).
– s est un indice pour les fournisseurs ($s \in S$).

3.3.2.2 Les variables du modèle

Les variables du model sont les suivantes :

b_{sk} : est la quantité de matière première à expédier du fournisseur s à l'usine k.

f_{kj} : est la quantité de produit à expédier de l'usine k au CD_j.

q_{ij} : est la quantité de produit à expédier du CD_j au client i.

$$Z_j = \begin{cases} 1 & \text{si le } CD_j \text{ est ouvert,} \\ 0 & \text{sinon.} \end{cases}$$

$$P_k = \begin{cases} 1 & \text{si l'usine k est ouverte,} \\ 0 & \text{sinon.} \end{cases}$$

$$Y_{ji} = \begin{cases} 1 & \text{si } CD_j \text{ sert client i,} \\ 0 & \text{sinon.} \end{cases}$$

3.3.2.3 Les paramètres du modèle

D_k : est la capacité de l'usine k.

W_j : est la capacité du dépôt j (CD_j).

sup_s : est la capacité du fournisseur s de la matière première

d_i : est la demande du client i.

W_{max} : est le nombre maximum des CDs ouverts.

P_{max} : est le nombre maximum des usines ouvertes.

v_j : est le coût fixe annuel de fonctionnement du dépôt j.

g_k : est le coût fixe annuel de fonctionnement de l'usine k.

c_{ji} : est le coût de transport unitaire du produit de CD j au client i.

a_{kj} : est le coût de transport unitaire du produit de l'usine k à CD_j.

t_{sk} : est le coût de revient d'une unité de la matière première à l'usine k depuis le fournisseur s.

h_{ji} est le temps de livraison de dépôt j au client i.

τ : est le délai de livraison maximal admissible à partir des dépôts aux clients.

C_j : est l'ensemble des clients qui peut être atteint à partir de CD_j en respectant le paramètres τ, c.à.d $C_j = \{i | h_{ji} \leq \tau\}$.

o_D est l'ensemble des CDs déjà ouverts.

o_P est l'ensemble des usines déjà ouvertes.

r_1 et r_2 sont les poids des usines et des CDs, respectivement.

3.3.2.4 Modèle mathématique

Les fonctions objectifs :

$$\begin{aligned} Minf1 &= \sum_{k\in K} g_k * P_k + \sum_{j\in J} v_j * Z_j + \sum_{s\in S}\sum_{k\in K} t_{sk} * b_{sk} \\ &+ \sum_{k\in K}\sum_{j\in J} a_{kj} * f_{kj} + \sum_{j\in J}\sum_{i\in I} c_{ji} * q_{ji}. \end{aligned} \tag{3.1}$$

$$Maxf2 = \frac{\sum_{j\in x}\sum_{i\in x} q_{ji}}{\sum_{i\in I} d_i}. \tag{3.2}$$

$$\begin{aligned} Minf3 &= r_1 \left[\sum_{k\in o_P} \left[(\sum_{j\in o_D} \frac{f_{kj}}{D_k}) - (\sum_{k\in o_P}\sum_{j\in o_D} f_{kj} / \sum_{k\in o_P} D_k) \right]^2 / |o_P| \right]^{1/2} \\ &+ r_2 \left[\sum_{j\in o_D} \left[(\sum_{i} \frac{q_{ji}}{W_j}) - (\sum_{j\in o_D}\sum_{i} q_{ji} / \sum_{j\in o_D} W_j) \right]^2 / |o_D| \right]^{1/2} \end{aligned} \tag{3.3}$$

Les contraintes :

$$\sum_{j \in J} Y_{ji} \leq 1, \qquad \forall i \in I. \tag{3.4}$$

$$\sum_{i \in I} d_i * Y_{ji} \leq W_j * Z_j, \qquad \forall j \in J. \tag{3.5}$$

$$\sum_{j \in J} Z_j \leq W_{max}. \tag{3.6}$$

$$q_{ji} = d_i * Y_{ji}, \qquad \forall i \in I, \ \forall j \in J. \tag{3.7}$$

$$\sum_{k \in K} f_{kj} = \sum_{i \in I} q_{ji}, \qquad \forall j \in J. \tag{3.8}$$

$$\sum_{k \in K} b_{sk} \leq sup_s, \qquad \forall s \in S. \tag{3.9}$$

$$\sum_{j \in J} f_{kj} = \sum_{s \in S} b_{sk}, \qquad \forall k \in K. \tag{3.10}$$

$$\sum_{j \in J} f_{kj} \leq D_k * p_k, \qquad \forall k \in K. \tag{3.11}$$

$$\sum_{k \in K} P_k \leq P_{max}. \tag{3.12}$$

3.3.2.5 Interprétation des objectifs

- **Objectif** $f1$: sert à minimiser le coût total du réseau supply chain. Il comprend les coûts d'achat de la matières premières, coûts fixes d'exploitation et d'ouverture des usines et CDs respectivement, les coûts unitaires de transport dans les trois stages.
- **Objectif** $f2$: sert maximiser le taux de satisfaction de la clientèle (en %) dans un délai de livraison imposé τ fixé au préalable.
- **Objectif** $f3$: sert à maximiser l'équilibre d'utilisation des capacités des ressources usines et des CDs, ceci est traduit par la minimisation de l'erreur quadratique moyenne de taux d'utilisation des capacités.

3.3.2.6 Interprétation des contraintes

- La contrainte 3.4 assure qu'un client ne doit être servi que par un seul dépôt.
- La contrainte 3.5 assure le respect des capacités des centres de distribution(CDs).
- La contrainte 3.6 assure que le nombre des centres de distribution ouverts ne dépasse pas W_{max}.

– La contrainte 3.7 assure la satisfaction des demandes des clients.
– La contrainte 3.8 assure l'équilibre entre le niveau 3 et le niveau 2 de la chaîne.
– La contrainte 3.9 assure le respect des capacités des fournisseurs.
– La contrainte 3.10 assure l'équilibre entre la niveau 2 et niveau 1 de la chaîne.
– La contrainte 3.11 assure le respect des capacités des usines.
– La contrainte 3.12 assure que le nombre des usines ouvertes ne dépasse pas P_{max}.

3.3.2.7 Analyse du modèle

Le modèle présenté est un modèle de Programmation Multi-Objectif Non-Linéaire en Nombre Mixte Entier. Notons que la dimension du modèle adopté se traduit par le nombre de variables et de contraintes utilisées. Le modèle adopté est composé de :
$(S * K) + (K * J) + 2 * (J * I) + I + J$ variables ;
$I + 2 * J + (I * J) + S + 2 * K + 2$ constraints.

A titre d'exemple :

Classes de problème	S	K	J	I	Pmax	Wmax	Variables	Contraintes
Classe I	10	10	10	21	6	6	651	283
Classe II	20	15	12	50	9	7	1802	731
Classe III	10	6	8	100	4	5	1848	944
Classe IV	20	15	12	100	9	7	3052	1381

TABLE 3.1 – Nombre de variables et de contraintes

3.3.3 Une approche de résolution basée sur les Algorithmes Génétiques

Les différents mécanismes du l'Algorithme Génétique adapté sont décrits ci-dessous :

3.3.3.1 Codage des solutions

Comme mentionné dans le chapitre précédent l'application des AGs à un problème donné nécessite un codage chromosomique des solutions. Ce dernier est un des enjeux importants qui affectent les performances des AGs. Le codage basé sur l'arbre est un moyen pour coder les solutions des problèmes d'optimisation dans les réseaux. Fondamentalement, il existe trois façons de coder cette arbre : le codage basé sur les arêtes, le codage basé sur les sommets, et le codage sommets-arêtes [106].

La première application des AGs aux problèmes de transport/distribution a été effectuée par Michalewicz [105]. Il a utilisé un codage matriciel basé sur les arêtes pour coder l'arbre de transport dont la taille de la matrice représentative est égale à $|K|.|J|$ où $|K|$ est le nombre de sources (fournisseurs, usines ou dépôts) et $|J|$ est le nombre de puits (usines, dépôt ou client), Ce codage nécessite non seulement trop de mémoire sur l'environnement de l'ordinateur, mais aussi des opérateurs génétiques spéciaux pour réparer les solutions non réalisables. Un autre codage de l'arbre est Prüfer number basé sur les sommets dont la taille du vecteur représentant les solutions est égale $|K|+|J|-2$ pour coder un arbre de transport de $|K|$ sources et $|J|$ puits. Ce codage a été développé initialement pour coder les arbres couvrant, par la suite il a été appliqué avec succès par les auteurs Gen and Cheng dans [] pour le problème de transport, ce codage a besoin de quelques mécanismes de réparation pour obtenir des solutions réalisables après l'application des opérateurs génétiques.

Dans notre étude nous avons codé les solutions avec des chromosomes de trois parties, chacune représente un niveau de la chaîne considérée voir la figure 3.2.

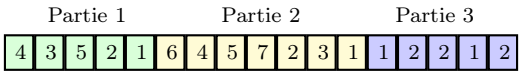


FIGURE 3.2 – Codage chrosomique des solution.

Pour les deux premiers niveaux nous avons utilisé un codage basé sur la priorité qui nous permet de maîtriser les mécanismes de réparation des chromosomes après application des opérateurs génétiques, c.à.d ces mécanismes seront moins compliqués, ce qui augmente la vitesse et la performance de l'algorithme génétique. Pour le troisième niveau nous avons une contrainte 3.4, qui impose q'un client ne sera servi par qu'un seul dépôt ce qui nous a amené à utiliser un codage entier dont le principe est simple, la taille du vecteur est égale au nombre de clients. Une position dans le vecteur représente un client et sa valeur représente le dépôt qui correspond. La figure 3.3 illustre un exemple.

Le codage basé sur la priorité a été proposer par les auteurs Gen and Cheng (2000) dans [106], dont l'application a été pour la résolution des problèmes du plus court chemin et la planification de projet respectivement. La première application de ce codage à un problème de transport multi-niveau avec produit unique a été proposée par Gen et al en 2006 [104], par la suite une généralisation pour un problème multi-produits a été proposée par les mêmes auteurs en 2009 [107]. Dans le codage basé sur la priorité de l'arbre de transport correspondant à une solution est représenté par un vecteur de taille $|K| + |J|$ où $|K|$ est le nombre de sources et $|J|$ es le nombre de puits dans un réseau de transport.

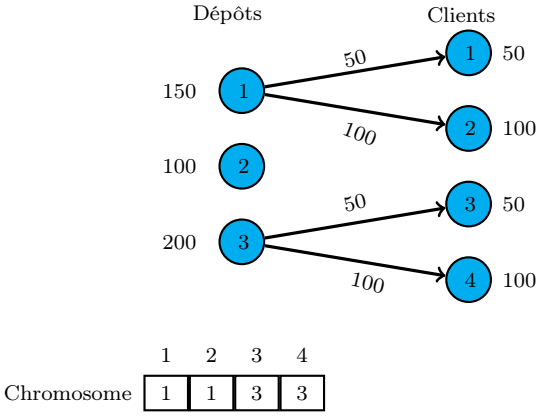


FIGURE 3.3 – Construction d'un arbre de transport depuis un codage entier.

Une position dans le vecteur représente un noeud de l'arbre de transport et sa valeur est utilisée pour représenter la priorité de ce noeud (source ou puits) dans la construction de l'arbre.

La construction de l'arbre représentant la solution depuis le vecteur des priorités (chromosome) est faite ainsi, commençons par déterminer le noeud (source/puits) avec la plus grande priorité, et compléter le couplet source/puits par le noeud (source ou puits) avec la plus faible valeur de coût dans la matrice des coûts de transport et finalement connecter le couplet source/puits par un arc, par la suite, si la capacité du noeud source est supérieure à la demande du noeud puits la valeur de la position du noeud puits dans le vecteur (chromosome) est mise à 0 et la quantité transportée est égale à la demande du noeud puits, si la capacité du noeud source est égale à la demande du noeud puits les valeurs des positions des deux noeuds source et puits dans le vecteur (chromosome) sont mises à 0 et la quantité transportée est égale à la demande du noeud source, sinon la valeur de la position du noeud source dans le vecteur (chromosome) est mise à 0 et la quantité transportée est égale à la capacité du noeud source et ainsi de suite (voir algorithme 2).

Algorithm 2 Décodage d'un chromosome basé sur la priorité.

Entrées

ES : Ensemble de Sources, |ES|=K ; EP : Ensemble de Puits, |EP|=J ;

Dm_j : la demande du puits $j \in J$; Cp_k : la capacité de la source $k \in K$;

V : Vecteur de taille |K|+|J| ; C_{kj} : Matrice des coûts de transport.

Début

- **étape 1 :** $g_{kj} = 0, \forall k \in K, \forall j \in J$;
- **étape 2 :** $l = \arg\max\{V(t), t \in |K| + |J|\}$; //choix du noeud 1
- **étape 3 :**
  Si $l \in K$ alors $k^* = l$; // si noeud 1 est une source
  $j^* = \arg\min\{C_{kj} | V(j) \neq 0, j \in J\}$; choisir le noeud 2 (puits)
  Sinon $j^* = l$; //si noeud 1 est un puits
  $k^* = \arg\min\{C_{kj} | V(k) \neq 0, k \in K\}$; choisir le noeud 2 (source)
- **étape 4 :**
  $g_{k^*j^*} = \min\{Cp_{k^*}, Dm_{j^*}\}$; //calculer la quantité à transporter
  $Cp_{k^*} = Cp_{k^*} - g_{k^*j^*}$;
  $Dm_{j^*} = Dm_{j^*} - g_{k^*j^*}$;
- **étape 5 :**
  Si $Cp_{k^*} = 0$ alors $V(k^*) = 0$;
  Si $Dm_{j^*} = 0$ alors $V(j^*) = 0$;
- **étape 6 :**
  Si $V(j) \neq 0, \forall j \in J$ alors aller à **étape 2** ;
  Sinon **Arrêter.**

Fin

Sortie g_{kj} : Matrice de distribution (Arbre de distribution).

La figure 3.4 et le tableau 3.2 schématisent un exemple [104] de construction d'un arbre de transport depuis un vecteur codé par un codage basé sur la priorité :

Itération	Vecteur	Capacités	Demandes	Source	Puits	Quantité
0	[374|2615]	(100,100,150)	(50,150,100,50)	2	2	100
1	[304|2615]	(100,0,150)	(50,50,100,50)	3	2	50
2	[304|2015]	(100,0,100)	(50,0,100,50)	3	4	50
3	[304|2010]	(100,0,50)	(50,0,100,0)	3	3	50
4	[300|2010]	(100,0,0)	(50,0,50,0)	1	1	50
5	[300|0010]	(50,0,0)	(0,0,50,0)	1	3	50
6	[300|0000]	(0,0,0)	(0,0,0,0)			

TABLE 3.2 – Tableau des itérations de construction de l'arbre de transport.

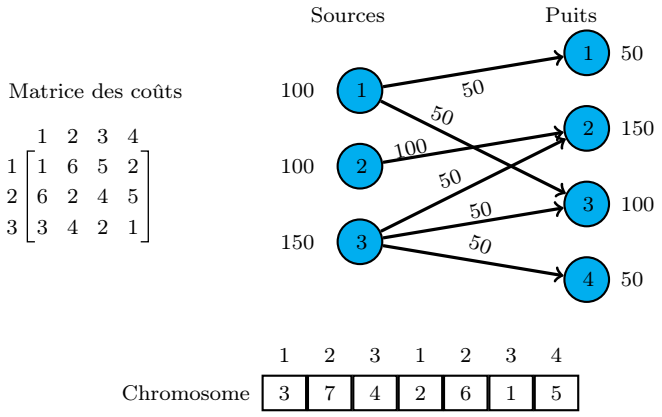


FIGURE 3.4 – Construction d'un arbre de transport depuis un codage basé sur la priorité.

Le décodage des chromosomes ce fait en arrière c.à.d. commence par la construction de l'arbre de transport qui correspond au troisième niveau (dépôts/clients), par la suite satisfaire les nouvelles demandes calculées au niveau des dépôts pour construire l'arbre de transport qui correspond au deuxième niveau (usines/dépots), et finalement satisfaire les nouvelles demandes calculées au niveau des usines pour construire l'arbre de transport qui correspond au premiere niveau (fournisseurs/usines).

3.3.3.2 Génération de la population initiale

La génération de la population initiale conditionnent fortement la rapidité de convergence d'un Algorithme Génétique, pour cela nous avons appliqué une méthode heuristique basée d'une part sur l'aléatoire (diversité), dont le principe de génération des chromosomes réalisables est le suivant :

- La génération de la partie 1 se fait par un remplissage aléatoire d'un vecteur de taille $|S| + |K|$ par des valeurs entières sans répétition dans l'intervalle $[1, |S| + |K|]$;
- La génération de la partie 2 se fait par un remplissage aléatoire d'un vecteur de taille $|K| + |J|$ par les valeurs entières sans répétition dans l'intervalle $[1, |K| + |J|]$;
- La génération de la partie 3 se fait par un remplissage aléatoire d'un vecteur de taille $|I|$ par des valeurs représentaient des sources (CDs).

3.3.3.3 Opérateur Croisement

Pour permettre à notre algorithme génétique une évolution au cours des génération, nous avons utilisé un opérateur de croisement pour produire de nouveaux chromosomes, c'est un opérateur de croisement uniforme basé sur l'échange des segments dans chromo-

somes parents. Le principe de cet opérateur est de générer un masque binaire de trois valeurs, chaque valeur corresponde à l'action de l'échange des deux segments représentant les niveaux de la chaîne logistique dans les chromosomes parents, si la valeur dans le masque est égale à "0" les deux enfants aurons les mêmes segment que les parents, si la valeur dans le masque est égale à "1" les deux segments seront inverser, voir figure 3.5.

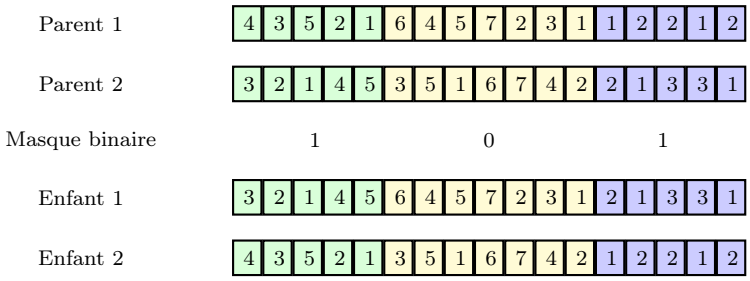


FIGURE 3.5 – Opérateur de croisement.

3.3.3.4 Opérateur de Mutation

L'opérateur de mutation évite d'établir des populations uniformes incapable d'évoluer. Il consiste à modifier les valeurs des gènes des chromosomes d'une manière aléatoire, dans notre AG nous avons utilisé deux opérateurs de mutation, voir figure 3.6.

- Pour le 3^{me} niveau de la chaîne logistique où le codage entier la mutation est faite ainsi, choisir aléatoirement une position dans le segment et faire changer le contenu aléatoirement, c.à.d. choisir aléatoirement un client et changer aléatoirement le dépôt qui le sert.
- Pour le 1^{ier} et 2^{me} niveau de la chaîne logistique où le codage est basé sur la priorité la mutation est faite ainsi, choisir deux positions dans le segment et inverser les contenus.

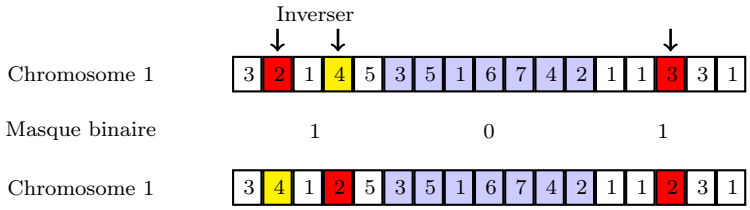


FIGURE 3.6 – Opérateur de Mutation.

- **Réparation des chromosomes mutés** :
  Après avoir appliquer la mutation, les chromosomes mutés risquent de se retrouver

pour des solutions non réalisables, dans ce cas nous avons prévu un mécanisme de réparation dont le principe est dans l'algorithme 3.

Algorithm 3 Réparation d'un chromosome.

Entrées

- $Source$: Ensemble des Sources ;
- $Puits$: Ensemble des puits ;
- $Souv$: Ensemble des sources ouvertes ;
- $Souf$: Ensemble des sources fermées ;
- Lim_{max} : Nombre maximum de sources à ouvrir ;
- Dem : Demande totale des puits ;
- Cap_{Souv} : Capacité des sources ouvertes.

Tant que ($|Souv| > Lim_{max}$ **ou** $Cap_{Souv} < Dem$)

Début

1. Si $|Souv| > Lim_{max}$
   - Calculer l'ensemble CS dans l'ensemble $Souv$ tel que : $CS = \{source_j \in Souv | Cap_{Souv} - Cap_{source_j} \geq Dem\}$;
   - Fermer aléatoirement une $source_j \in CS$ jusqu'à $|Souv| \leq Lim_{max}$.
   - Obtenir un nouveau chromosome réalisable par l'affectation des clients affectés déjà à la $source_j \in CS$ à une $source_r \in Souv$.
2. Si $Cap_{Souv} < Dem$
   - Ouvrir aléatoirement une $source_j \in Souf$ jusqu'à $Cap_{Souv} \geq Dem$;
   - Obtenir un nouveau chromosome réalisable par l'affectation des clients affectés déjà à une $source_j$ dépassée par la capacité à une $source_r \in Souv$; Aller à (1).

Fin

Sortie Chromosome réalisable.

3.3.3.5 Les variantes de l'Algorithme Génétique

Comme dans la nature, il est nécessaire de prévoir des mécanismes de sélection permettant de mesurer l'aptitude des individus pour survivre. Dans le chapitre précédent, nous avons précisé qu'il existe plusieurs techniques définissant ces mécanismes de sélection des individus participant aux opérateurs d'évolution. Dans l'Algorithme Génétique proposé pour la résolution approchée du problème d'optimisation multi-objectif dans les chaînes logistiques, nous avons utilisé quatre stratégies de sélection. Nous décrivons dans la suite ces variantes (AG-1, AG-2, AG-3 et AG-4 les) correspondant aux différentes techniques de sélection utilisées.

- AG-1 : Algorithme Génétique utilisant une sélection basée l'agrégation des objectif avec des poids générés de manière aléatoire (Murata, Ishibuchi, et Tanaka (1996)).

- AG-2 : Algorithme Génétique utilisant une sélection basée l'agrégation des objectif avec des poids calculés en fonction du point idéale (Zhou and Gen (1999)).
- AG-3 : Algorithme Génétique utilisant une sélection NSGA (Srinivas et Deb 1995).
- AG-4 : Algorithme Génétique utilisant une sélection WAR (Bentley et Wakefield 1997).

Rappelons que ces quatre variantes ont les mêmes étapes de l'Algorithme Génétique détaillés dans les sections précédentes (codage, croisement et mutation) et utilisant la même population initiale.

3.4 Conclusion

Nous avons consacré ce chapitre à l'introduction des concepts de bases sur les chaînes logistiques, leurs définitions, leurs conceptions, leur gestion, et de la prise de décisions dans différents niveaux de la chaîne. Nous avons présenté ensuite les travaux en relation avec la gestion et l'optimisation des chaînes logistiques. Dans un second temps, nous avons proposé quatre variantes de l'Algorithme Génétique pour la résolution approchée du problème d'optimisation multi-objectif dans les chaînes logistiques.

Le chapitre suivant détaille les tests expérimentaux effectués pour comparer l'efficacité de ces quatre variantes proposées.

Chapitre 4

Expérimentations et Résultats

4.1 Introduction

Dans ce chapitre, nous décrivons les expérimentations réalisées pour la comparison des performances des quatre variantes de l'Algorithme Génétique proposées pour la résolution approchée du problème d'optimisation multi-objectif dans les chaînes logistiques. Ensuite, nous analysons les différents résultats obtenus.

Nous rappelons que les deux premières variantes sont basées sur l'agrégation des objectifs en un seul objectif à l'aide de la technique de pondération. Dans la première variante, le calcul des poids associés aux objectifs utilisé pour évaluer et sélectionner les nouveaux individus est effectué de manière aléatoire. Tandis que dans la seconde variante, le calcul se fait en utilisant le point idéal correspondant à la population courante. Par contre, la troisième variante est dotée de la technique de sélection NSGA, où les individus sont triés par ordre de dominance. La dernière variante est basée sur la sélection WAR, où les individus sont scindés en k groupes (k est le nombre d'objectifs) et chaque groupe est trié par ordre de valeur de leur objectif.

Dans un second temps, nous étudions l'impact de l'élitisme sur la variante de l'AG basée sur la sélection NSGA. Et finalement nous testons l'impact de la technique de surpeuplement (crowding distance). L'implementation informatique est effectuée en C++ Builder XE4 sous Windows 8.1 (voir annexe 2), pour la résolution d'une série d'instances de différentes tailles générées aléatoirement. Les expériences numériques sont effectuées sur un PC DELL Intel I3 3.3GHz de 4GB de RAM.

Nous détaillons les différentes étapes de l'expérimentation et une analyse des résultats.

4.2 Génération des problèmes test

La génération aléatoire des instances utilisées pour la validation des quatres variantes de l'AG est tirée des travaux de Costa A. et al., (2010) [110]. Dans ces travaux, quatre classes de problèmes sont proposées, voir tableau 4.1. **Où :**

Classes de problème	S	K	J	I	Pmax	Wmax
Classe I	10	10	10	21	6	6
Classe II	20	15	12	50	9	7
Classe III	10	6	8	100	4	5
Classe IV	20	15	12	100	9	7

TABLE 4.1 – Classes des problèmes générés

- S : Nombre de Fournisseurs ;
- K : Nombre d'Usines ;
- J : Nombre de Dépôts ;
- I : Nombre de Clients ;
- P_{max} : Nombre maximum d'usines à ouvrir ;
- W_{max} : Nombre maximum de dépôts à ouvrir.

Les données des instances sont générées comme suit :

1. Les demandes des clients sont générées selon une distribution uniforme discrète de probabilité $p = 0.2$ pour chaque élément de l'ensemble $100, 150, 200, 250, 300$.
2. Les capacités de chacune des installations potentielles est générée suivant la distribution normale $N(\mu, \sigma^2)$ avec la formule suivante : $cap = 110 + \lfloor rand(N(\mu, \sigma^2)) \rfloor$, où $\mu = \sum_i d_i/T$ et $\sigma = \sum_i d_i/4T$; avec T égale à W_{max}, P_{max} et S pour les dépôts, usines et fournisseurs respectivement.
3. Les coûts fixes d'installation des CDs et des usines sont générés par les formules suivantes :
   $v_j = rand\{N(2.5.\bar{W}; (2.5/4\bar{W})^2)\}, \qquad j = 1, \ldots, J.$
   $g_k = rand\{N(3.\bar{D}; (3/4\bar{D})^2)\}, \qquad k = 1, \ldots, K.$
   où $\bar{W} = \sum_j W_j/J$ et $\bar{D} = \sum_k D_k/K$
4. Les coûts de transport dans chaque niveau de la chaîne logistique sont générés selon la distribution uniforme discrète dans les intervalles : $c_{ji} \in [3, 4, 5, \ldots, 9]$, $a_{kj} \in [3, 4, 5, \ldots, 8]$ et $t_{sk} \in [3, 4, 5, \ldots, 7]$ pour les niveaux 3, 2 et 1 respectivement.

5. Dans notre cas, le délai de livraison maximal admissible à partir des dépôts aux clients noté τ est fixé à 97.5 %.

4.3 Expérimentations et discussions

4.3.1 Évaluation des variantes de l'Algorithme Génétique

Dans le but d'évaluer les performances des quatre variantes d'Algorithme Génétique proposées pour la résolution du problème d'optimisation multi-objectif des chaîne logistique, nous avons testé ces approches sur des classes de problèmes générés de manière aléatoire (voir section précédente). Pour chaque classe 10 problèmes test sont générés.

Pour les variantes proposées les paramètres de l'Algorithme sont fixés comme suit :

- La taille d'une population est fixée à 400 individus.
- Le nombre de générations est égale à 1000.
- La probabilité de croisement est de 0.7.
- La probabilité de mutation est de 0.5.

La mesure de performance utilisé pour évaluer l'efficacité de ces variantes est le taux de solutions non dominées générées par la variante AG-i en % est calculé comme suit :

$$T(SN_i) = \frac{|SNDi - \{X \in SNDi | \exists Y \in SND : X \prec Y\}|}{|SND|}$$

Où :

- SND1 : ensemble de solutions non dominées générées par la variantes AG-1.
- SND2 : ensemble de solutions non dominées générées par la variantes AG-2.
- SND3 : ensemble de solutions non dominées générées par la variantes AG-3.
- SND4 : ensemble de solutions non dominées générées par la variantes AG-4.
- SND = SND1 $\cup$ SND2 $\cup$ SND3 $\cup$ SND4.

Le résumé des résultats est présenté dans le tableau 4.2 et le détail de ces résultats est dans l'annexe1,

Instances	Nbr-Sol	Stratégie 1		Stratégie 2		NSGA		WAR	
-	-	Temps	T(SN1)	Temps	T(SN2)	Temps	T(SN3)	Temps	T(SN4)
Classe I	163.8	21	22.14	21	37.45	150	63.95	196	6.52
Classe II	164.3	29	16.14	27	27.70	176	68.54	244	19.26
Classe III	116.5	34	14.61	33	32.01	187	60.96	254	14.87
Classe IV	172.3	50	08.63	47	14.41	243	79.59	371	22.17
Moyenne	-	-	15.38	-	27.89	-	68.26	-	15.71

TABLE 4.2 – Expérimentations et résultats

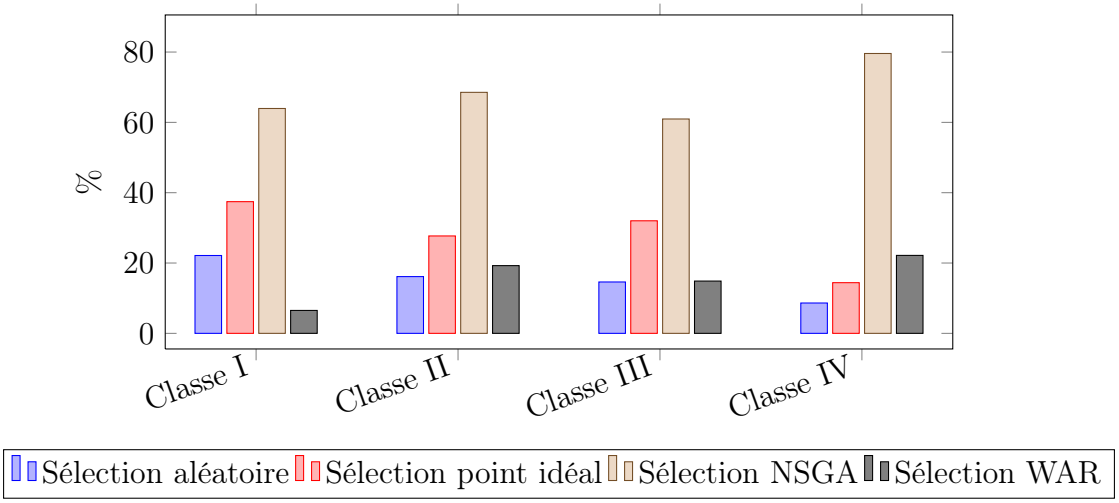


FIGURE 4.1 – Taux de solutions non dominées des quatres variantes

Une analyse sur les expérimentations comparant les quatre variantes de l'Algorithme Génétique réalisées sur quatre classes d'instances générées de manière aléatoire (figure 4.1), nous conduit à faire le constat que la variante AG-3 (sélection NSGA) est plus performante que les autres. En effet, elle est apte de trouver plus de 60% de l'ensemble des solutions non dominées générées par toutes les variantes (63.95%, 68.54%, 60.96% et 79.59% pour les Classes I, II, III et IV respectivement), suivi de la variante AG-2 (37.45%, 27.70%, 32.01%, 14.41% pour les Classes I, II, III et IV respectivement), puis la variante AG-4 (6.52%, 19.27%, 14.87%, 22.17% pour les Classes I, II, III et IV respectivement), et enfin la variante AG-1 basée sur la sélection aléatoire (22.14%, 16.14%, 14.61% et 08.63% pour les Classes I, II, III et IV respectivement).
Signalons que le temps moyen d'exécution des deux variantes AG-1, AG2 est meilleur, suivi du temps moyen d'execution de la variante AG-3, et enfin du temps moyen d'execution de la variante AG-4 (figure 4.2).

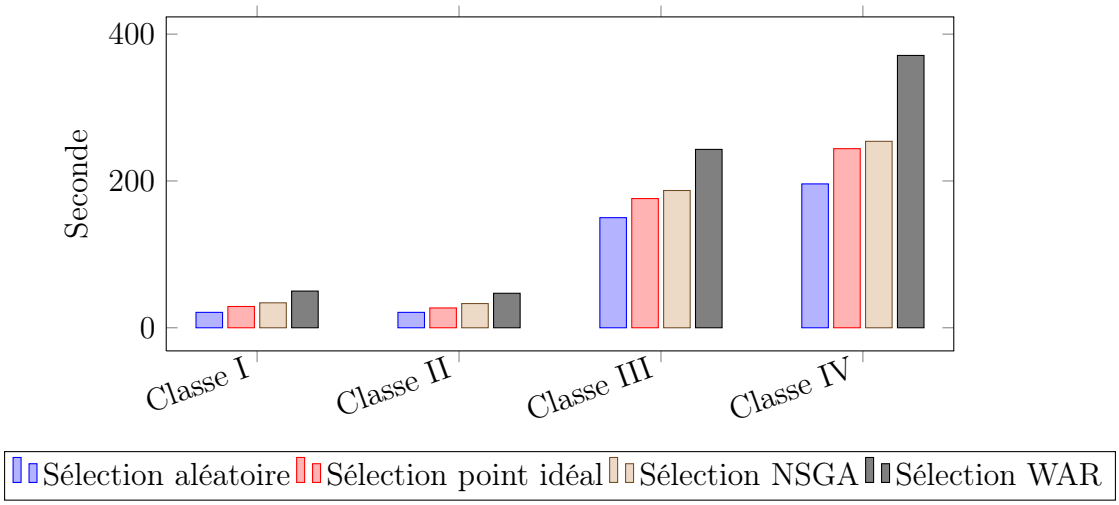


FIGURE 4.2 – Temps moyen d'exécution (seconde)

4.3.2 Impact de l'élitisme et du surpeuplement (NSGA-II)

L'algorithme NSGA-II a été proposé par Deb et al [111]. Il intègre un opérateur de sélection, basé sur un calcul de la distance de surpeuplement (ou crowding) détaillé dans le chapitre II, ce calcul estime la densité de chaque individu dans la population. Comparativement à NSGA, NSGA-II obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de K. Deb, ce qui fait de cet algorithme un des plus utilisés aujourd'hui car :

- Il utilise une approche élitiste qui permet de sauvegarder les meilleures solutions trouvées lors des générations précédentes.
- Il utilise une procedure de tri basée sur la non-dominance, plus rapide.
- Il utilise l'opérateur de comparaison basé sur le calcul de la distance de crowding contrairement à celui de NSGA basé sur la technique du partage.

Dans le but d'étudier l'impact des deux techniques constituant la différence entre le NSGA (partage) et le NSGA-II (élitisme + surpeuplement) nous avons adapté l'Algorithme NSGA-II afin de résoudre le problème d'optimisation multi-objectif dans les chaînes logistiques avec les mêmes opérateurs génétiques cités dans le chapitre précèdent (codage, croisement et mutation) et avec la même population initiale, les résultats comparatifs de cet algorithme avec la variante AG-3 basée sur la sélection NSGA sont résumés dans le tableau 4.3 et le détail de ces résultats est dans l'annexe1,

Instances	Nbr-Sol	AG-3		NSGA-II	
-	-	Temps(s)	T(SN1)(%)	Temps(s)	T(SN2)(%)
Classe I	146.5	150	64.40	201	80.13
Classe II	128.5	176	78.24	245	85.78
Classe III	107.8	187	76.41	253	84.82
Classe IV	134.8	243	80.61	312	90.64
Moyenne			74.91		85.34

TABLE 4.3 – Comparaison variante AG-3 et NSGA-II

Une analyse sur les expérimentations comparant la variante AG-3 avec NSGA-II réalisées sur quatre classes d'instances générées de manière aléatoire (figure 4.3) montre que le NSGA-II est plus performante où elle est capable de se rapprocher plus que la variante AG-3 vers le front Pareto-optimal, sur l'ensemble des tests réalisés, nous constatons que la NSGA-II est apte de trouver plus de 85% de l'ensemble des solutions non dominées générées par les deux algorithmes, la variante AG-3 est capable de calculer plus de 74% de cet même ensemble, cette performance est expliquée par la capacité de l'élitisme de sauvegarder les bonnes solutions trouvées au cours de passage d'une génération à une autre, et aussi par le maintien de diversité lors de la sélection assuré par la technique du calcul de crowding. Signalons que le temps d'exécution de la méthode NSGA-II est plus important et ceci est dû au calcul des distances de surpeuplement et aux opérations de comparaisons afin de sauvegarder les solution élites (élitisme).

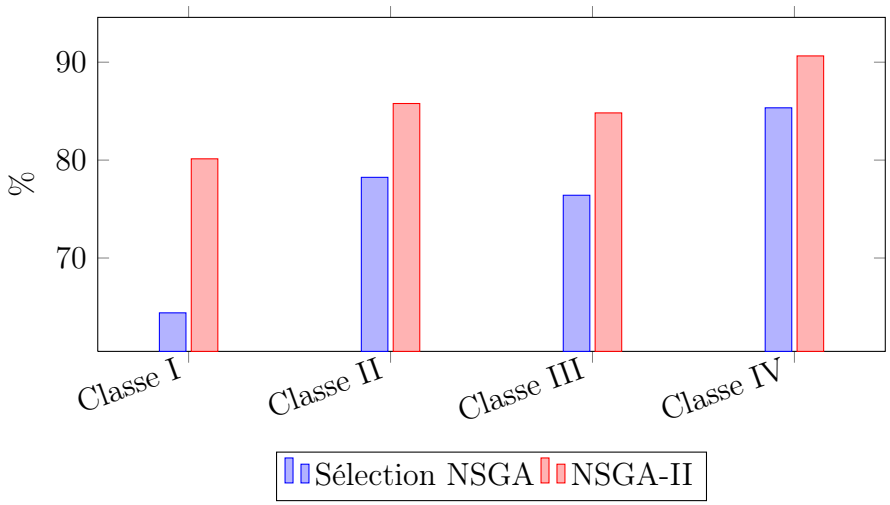


FIGURE 4.3 – Taux de solutions non dominées Sélection NSGA, et l'algorithme NSGA-II

4.4 Conclusion

Nous avons, dans ce chapitre, réalisé une série d'expérimentations afin d'évaluer la performance des variantes de l'Algorithme Génétique proposées, et de juger sur l'efficacité de la variante AG-3 basée sur la sélection NSGA de se rapprocher du front Pareto-optimal plus que les autres techniques de sélection.

Dans un second temps nous avons adapté la méthode la plus connues dans la literature pour résoudre les problèmes d'optimisation multi-objectif NSGA-II pour la résolution du problème traité dans cette thèse, et nous avons terminer par une comparaison de la variante AG-3 et NSGA-II, nous avons séduit que la NSGA-II est plus performante que la variante AG-3.

Conclusion Générale

Partant de l'importance d'une bonne gestion de la chaîne logistique qui constitue une préoccupation majeure des entreprises dans un environnement fortement concurrentiel où la performance du système de gestion doit être optimale. Nous nous sommes intéressés dans notre travail de magister au problème d'optimisation d'une chaîne logistique globale en considérant trois objectifs et ce pour mieux maîtriser cette problématique.

Il s'agit de determiner simultanément les meilleures configurations des sous problèmes (affectation, sélection, localisation) constituant le modèle global de la chaîne logistique. La modélisation de ce problème nous a amené à un programme d'optimisation multi-objectif non-linéaire en nombres entiers mixtes, appartenant à la classe des problèmes NP-difficiles. Devant les limites des méthodes exactes pour la résolution de ce programme nous avons opté pour les Algorithmes Génétique qui sont reconnus comme étant une méthode bien adapté pour la résolution de ce type de programmes, par leurs performance d'explorer l'espace des solution réalisables qui dépende pas de la nature des objectifs considérés, aussi par leur manipulation de plusieurs solutions réalisables en même temps, cette dernière permet la détermination de l'ensemble du frontière de Pareto. Au cours de l'adaptation de l'Algorithme Génétique quatre procédures de sélection de chromosomes ont été appliquées constituant quatre variantes d'Algorithme Génétique.

Nous rappelons que les deux premières variantes sont basées sur l'agrégation des objectifs en un seul objectif à l'aide de la technique de pondération. Dans la première variante, le calcul des poids associés aux objectifs utilisé pour évaluer et sélectionner les nouveaux individus est effectué de manière aléatoire. Tandis que dans la seconde variante, le calcul se fait en utilisant le point idéal correspondant à la population courante. Par contre, la troisième variante est dotée de la technique de sélection NSGA, où les individus sont triés par ordre de dominance. La dernière variante est basée sur la sélection WAR,

où les individus sont scindés en k groupes (k est le nombre d'objectifs) et chaque groupe est trié par ordre de valeur de leur objectif.

Afin de mettre en pratique notre étude nous avons conçu une application d'aide à la décision pour la résolution du problème considéré. L'application est mise en oeuvre par le langage C++ basée sur la notion de pointeurs. Cette notion permet de définir des structures dynamiques évoluant au cours du temps. Celle-ci est implémentée sur la plate-forme Embarcadero C++Builder XE3 qui est une solution standard pour développer avec une seule base de code des applications fonctionnant à la fois sous Windows et Mac OSX.

Les expérimentations effectuées comparant les quatres variantes ont montré que la variante AG-3 (sélection NSGA) est plus performante que les autres. En effet, elle est apte de trouver plus 60% l'ensemble des solutions non dominées générées par toutes les variantes. Dans un second temps nous avons adopté l'algorithme génétique NSGA-II pour la résolution du problème traité dans cette thèse, nous avons terminé par une comparaison entre cet algorithme et la variante AG-3 de l'algorithme génétique proposé dans le chapitre 3, les résultats ont confirmé la performance de l'algorithme NSGA-II de se rapprocher du front pareto.

Ce modeste travail que nous venons d'effectuer, n'est qu'une parcelle de ce qui pourrait être fait dans l'avenir pour ce nouveau créneau qu'est la Gestion de la chaîne logistique. Nous souhaitons qu'il fasse l'objet de critiques et d'améliorations pour de futures recherches dans ce domaine.

Bibliographie

[1] Mayr, E. 1942 Systematics and the origin of species, from the viewpoint of a zoologist. Harvard University Press.

[2] Michalewicz, (2004) How to solve it : modern heuristics.

[3] Michael.Sakarovitch. *Techniques Mathématiques de la recherche opérationelle "Programmation linéaire".* E.N.S.I.M.A.G, Octobre 1979.

[4] Papadimitriou, C. (1994) Computational Complexity. AddisonWesley.

[5] Glover, F. (1986) Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research, 13(5), 533-549.

[6] Kirkpatrick, S., Gelatt, C. D. et Vecchi, M. P. (1983) Optimization by Simulated Annealing. Science, 220(4598), 671-680.

[7] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. et Teller, E. (1953) Equation of State Calculations by Fast Computing Machines. J. Chem. Phys., 21, 1087-1092.

[8] James Kennedy and Russell Eberhart. (1995) Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, volume IV, pages 1942-1948, Piscataway, NJ, 1995. IEEE Press.

[9] BELLMAN R. (1957) Dynamic programming, Princeton University Press.

[10] Geoffrion A.M. (1968) Proper Effeciency and the Theory of Vector Maximization, Journal of Mathematical Analysis and Applications 22, No-3, 618-630.

[11] Coello Coello C. A., Christiansen A. (1999) A Multiobjective Optimization Tool for Engineering Design, Engineering Optimization, volume 31, numéro 3, 337-368.

[12] Pareto V. (1896) Cours d'économie politique, vol.1 et 2, F. Rouge, Lausanne.

[13] Goldberg, D.E. (1989) Genetic algorithms for search, optimization, and machine learning. In : Addison-Wesley, MA : (ed), Reading.

[14] Schaffer, J.D. (1985) Multiple objective optimization with vector evaluated genetic algorithms. Pages 93-100 of : Grefenstette, J.J (ed), ICGA International Conference on Genetic Algorithms. Lecture Notes in Computer Science.

[15] Parmee I. C., Cevtkovic D., Watson A. W. et Bonham C. R. (2000) Multiobjective satisfaction within an interactive evolutionary design environment. Evolutionary Computation, 8(2) : 197-222.

[16] Haimes Y. Y.,Lasdon L. S. et Wismer D. A. (1986) On a bicriterion formulation of the problems of intergrated system identification and system optimization. IEEE Trans. On Systems, Man and Cybernetics, SMC-16 : 122-128.

[17] Coello C. A. C. (1996) n Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design, ph. D. Thesis, Department of Computer Science, Tulane University New Orleans.

[18] Hajela, P. et Lin, C-Y. (1992) Genetic search strategies in multicriterion optimal design. Structural Optimization, 99-107.

[19] Dahl, G., Jornsten, K., et Lokketangen, A. (1995) A tabu search approach to the channel minimization problem. Pages 369-377 of : In G. Liu, K-H. Phua, J. Ma J. Xu F. Gu, et C. He, editors (eds), Optimization- Techniques ans Applications, ICOTA'95, vol. 1. Chengdu, China : World Scientific.

[20] Serafini, P. (1992) Simulated annealing for multiple objective optimization problems. Pages 87-96 of : Tenth Int. Conf. on Multiple Criteria Decision Making.

[21] Ulungu, E.L., Teghem J., Fortemps Ph. et Tuyttens, D. (1999) MOSA method : a tool for solving multiobjective combinatorial optimization problems. Journal of MultiCriteria Decision Analysis, 8, 221-236.

[22] Fonseca, C.M., et Fleming, P.J. (1993) Genetic algorithms for multi-objective optimization : formulation, discussion and generalization. Pages 416-423 of The Fifth International Conference on Genetic Algorithms.

[23] Czyzak, P., et Jaszkiewicz, A. (1998) Pareto simulated annealing a metaheuristic technique for multiple-objective combinatorial optimisation. Journal of Multi-Criteria Decision Analysis, 7, 34-47.

[24] Suppapitnarm, A. et Parks, T. (2001) Simulated annealing : an alternative approach to true multiobjective optimization. In : Genetic and Evolutionary Computation Conference.

[25] Gandibleux, X., Mezdaoui, N., & Fréville, A. (1997) A multiobjective tabu search procedure to solve combinatorial optimization problems. Advances in Multiple Objective and Goal Programming. Lecture Notes in Economics and Mathematical Systems, 455, 291-300.

[26] Hansen, M.P. (1998) Metaheuristics for multiple objective combinatorial optimization. Ph.D. Thesis, Technical University of Denmark, Lyngby.

[27] Cohon, J. L. (1978) Multiobjective Programming and Planning. New York : Academic Press.

[28] Charnes A. and Cooper W.W. Management models and industrial applications of linear programming, John Wiley, New York, 1961.

[29] Y. Y. Haimes, L.S. Lasdon, and D. A. Wismer a becriterion formulation of the problems of integrated system identification and system optimization, IEEE. 1971

[30] Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research, 13(5), 533 549.

[31] L.J. Fogel, A.J. Owens et M.J. Walsh. 1966. Artificial intelligence through simulated evolution. New York : John Wiley, 1966. 11

[32] J.H. Holland. 1975. Adaptation in natural and artificail systems. University of Michigan Press.

[33] I.Rechenberg. 1973 Evolution strategie : Optimierung technisher systeme nach prinzipien des biologischen evolution. Fromman-Hozlboog Verlag, Stuttgart.

[34] J.R. Koza. 1992 Genetic programming : on the programming of computers by means of natural selection. MIT Press, Cambridge, MA, USA.

[35] J.R. Koza. 1994 Genetic programming ii : Automatic discovery of reusable programs. MIT Press, Cambridge, MA, USA.

[36] Holland 1975 John Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Harbor.

[37] Goldberg 1989 David E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts.

[38] Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms In J. J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA, pp. 93-100.

[39] Fourman, M. P. (1985). Compaction of symbolic layout using genetic algotihms In J. J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA, pp. 141-153.

[40] Kursawe, F. (1991). A variant of evolution strategies for vector optimization In H.-P. Schwefel and R. Männer (Eds.), Parallel Problem Solving from Nature - Proc. 1st Wotkshop PPSN, Berlin, pp. 193-197.

[41] Horn, J. (1997). F1.9 multicriteria decision making. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds), Handbook of Evolutionary Computation Bristol (UK) : Institute of Physics Publishing

[42] Hajela, P. and C.-Y. Lin (1992). Genetic search strategies in multicriterion optimal design Structural Optimization 4, 99-10

[43] Ishibuchi, H. and T. Murata (1996). Multi-objective genetic local search algorithm In Proceedings of 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), Piscataway, NJ, pp. 119-124 IEEE

[44] Veldhuizen, D. A. V. (1999, June). Multiobjective Evolutionary Algorithms : Classification, Analyses, and new Innovations Ph. D. thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University

[45] Golberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning Reading, Massachusetts : Addison-Wesley

[46] Deb, K. (1999b). Evolutionary algorithms for multi-criterion optimization in engineering design In Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGE'99)

[47] Fonseca, C. M. and P. J. Flemming (1993). Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization In S. Forest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, pp. 416-423

[48] Horn, J., N. Nafpliotis, and D. E. Goldberg (1994). A niched pareto genetic algorithm for multiobjective optimization In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation, Volume 1, Piscataway, NJ, pp. 82-87, IEEE

[49] Srinivas, N. and K. Deb (1994). Multiobjective optimization using non-dominated sorting in genetic algorithms. Evolutionary Computation 2(3), 221-248

[50] Fonseca, C. M. and P. J. Flemming (1995b). An overview of evolutionary algorithms in multiobjective optimization : Evolutionary Computation 3(1), 1-16

[51] Veldhuizen, D. A. V. and G. B. Lamont (1998b). Multiobjective evolutionary algorithm research : A history and analysis Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio

[52] Corne, D. W., Knowles, J. D., Oates, M. J. (2000). The Pareto envelope-based selection algorithm for multiobjective optimization. Proceedings of the Parallel Problem solving from Nature VI conference, p. 839-848.

[53] Goldberg, D. E. and J. Richardson (1987) *Genetic algorithms with sharing for multimodal function optmization In J. J. Grefenstette (Ed.), Genetic Algorithms and their Applications : Proceedings of the Second International Conference on Genetic Algorithms, Hillsdale.*

[54] De Jong, K. A. (1975) *An analysis of the behavior of a class of genetic adaptive systems Ph. D. thesis, University of Michigan.*

[55] Fonseca, C. M., Fleming P. J. (1993) *Genetic algorithms for multi-objective optimization : formulation, discussion and generalization. Proceedings of the 5th International Conference on Genetic Algorithms, pp. 416-423, San Mateo, California.*

[56] Srinivas, N., Deb K. (1994) *Multiobjective optimization using non-dominated sorting in genetic algorithms. Evolutionary Computation.*

[57] A. S. Fraser. (1962) *Simulation of Genetic Systems, Journal of theorical biology.*

[58] Cerf R. 1994 Une théorie asymptotique des algorithmes génétiques. Thèse de doctorat de l'Université de Montpelier.

[59] Goldberg D. E., Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications, pp. 41-49.

[60] De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems Ph. D. thesis, University of Michigan.

[61] Nils Aall Baricelli. (1957). Symbiogenetic Evolution Processes realized by Artificial Methods, Methodos, 9, p. 143-182, 1957.

[62] Nils Aall Baricelli. (1962). Numerical Testing of Evolution Theories, Part II Preliminary Tests of Performance, Symbiogenesis and terrestrial life, Acta Biotheoretica, XVI, p.99-126, 1962.

[63] A. S. Fraser. (1962). Simulation of Genetic Systems, Journal of theorical biology, 2, p. 329-364, 1962.

[64] John Holland. (1975) Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Harbor.

[65] David E. Goldberg. (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts.

[66] Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In Grefenstette, J., editor, ICGA Int. Conf. on Genetic Algorithms, pages 93-100. Lawrence Erlbaum.

[67] Srinivas, N. and Deb, K. (1995) Multiobjective optimisation using non-dominated sorting in genetic algorithms. Evolutionary Computation, 2(8) :221-248.

[68] Fonseca, C. and Fleming, P. (1995a) Multiobjective genetic algorithms made easy : Selection, sharing and mating restrictions. In IEEE Int. Conf. on Genetic Algorithms in Engineering Systems : Innovations and Applications, pages 45-52, Sheffield, UK.

[69] Bentley, P. and Wakefield, J. (1997) Soft Computing in Engineering Design and Manufacturing, chapter Finding acceptable Pareto-optimal solutions using multiobjective genetic algorithms, pages 231-240. Springer Verlag, London.

[70] Holland, J.H., (1975) Adaptation in natural and artificial systems. Michigan Press Univ., Ann Arbor, MI, USA.

[71] Fujita, K., Hirokawa, N., Akagi, S., Kimatura, S., and Yokohata, H. (1998) Multiobjective optimal design of automotive engine using genetic algorithm. In Design Engineering Technical Conferences DETC'98, pages 1-11, Atlanta, Georgia.

[72] Loughlin, D. and Ranjithan, S. (1997) The neighborhood contraint method : A genetic algorithmbased multiobjective optimization technique. In Back, T., editor, Seventh Int. Conf. on Genetic Algorithms ICGA'97, pages 666-6773, San Mateo, California. Morgan Kaufmann.

[73] Fonseca, C. M., Fleming P. J. (1993) Genetic algorithms for multi-objective optimization : formulation, discussion and generalization. Proceedings of the 5th International Conference on Genetic Algorithms, pp. 416-423, San Mateo, California.

[74] Horn, J., Nafpliotis N., Goldberg D. E. (1994) *A niched Pareto genetic algorithm for multiobjective optimization. Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Evolutionary Computation, vol. 1, pp. 82-87, Piscataway, NJ, IEEE Press.*

[75] Zitzler et Thiele en 1999 *Multiobjective evolutionary algorithms : A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation.*

[76] Knowles, J. D., Corne D. W. (2000) *Approximating the non-dominated front using the Pareto archived evolution strategy. Evolutionary Computation Journal.*

[77] H.-P. Schwefel, 1981. *Numerical Optimization of Computer Models, Wiley, Chichester.*

[78] Deb, K., Agrawal S., Pratap A., Meyarivan T. (2000) *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization : NSGA-II. Parallel Problem Solving from Nature (PPSN VI), pp. 849-858, Berlin.*

[79] Corne, D. W., Knowles, J. D., Oates, M. J. (2000). The Pareto envelope-based selection algorithm for multiobjective optimization. Proceedings of the Parallel Problem solving from Nature VI conference, p. 839-848.

[80] Deb, K., Goel, T. (2001) *Controlled elitist non-dominated sorting genetic algorithms for better convergence. Proceedings of the first International Conference on Evolutionary Multi-criterion Optimization.*

[81] Zitzler, E., Laumanns, M., Thiele, L. (2001) *SPEA2 : improving the strength Pareto evolutionary algorithm. Technical report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich.*

[82] Corne, D. W., Jerram, N. R., Knowles, J.D., Oates, M. J. (2001) *PESA-II : region-based selection in evolutionary multiobjective optimization. Proceedings of the Genetic and Evolutionary Computation Conference.*

[83] Tayur S.,Ganeshan R., Magazine M. (1999). Quantitative models for supply chain management. Kluwer Academic Publishers.

[84] Ganeshan, Ram, and Terry P. Harrison, (1995). An Introduction to Supply Chain management. Department of Management Sciences and Information Systems, 303 Beam Business Building, Penn State University, University Park, PA, USA.

[85] Vidal C.J., et Goetschalckx M., (1997) Strategic production-distribution models : a critical review with emphasis on global supply chain models. European Journal of Operational Research, vol (98), 1-18.

[86] Taylor D.A., (2003) Supply Chains : A Manager's Guide. Addison Wesley, USA.

[87] Hermann J.W., Lin E., Pundoor G., (2003) Supply Chain Simulation Modeling Using The Supply Chain Reference Model. Proceedings of DETC'03, 1-9, Chicago, USA.

[88] Ding H., (2004) Une Approche d'Optimisation Basée sur la Simulation pour la Conception de Chaînes Logistiques : Application dans les Industries Automobiles et Textiles. Thèse de doctorat, Université de Metz.

[89] Jain S., Workman R.W., Collins L.M, Ervin E.C., (2001) Development of a High-Level Supply Chain Simulation Model. Proceedings of the 2001 Winter Simulation Conference, 1129-1137.

[90] Bhasharan S., (1998) Simulation Analysis of a Manufacturing Supply Chain. Decision Sciences, vol(29), 633-657.

[91] Gumus, A. T., Guneri, A. F., Keles, S., (2009). Supply chain network design using an integrated neuro-fuzzy and MILP approach : A comparative design study, Expert Systems with Applications.

[92] Fulya Altiparmak, Mitsuo Gen, Lin Lin, Ismail Karaoglan. (2007). A steady-state genetic algorithm for multi-product supply chain network design, Computers & Industrial Engineering 56 (2009) 521-537.

[93] Jayaraman, V., & Pirkul, H. (2001). Planning and coordination of production and distribution facilities for multiple commodities. European Journal of Operational Research, 133, 394-408.

[94] Jayaraman, V., & Ross, A. (2003). A simulated annealing methodology to distribution network design and management. European Journal of Operational Research, 144, 629-645.

[95] Yan, H., Yu, Z., & Cheng, T. C. E. (2003). A strategic model for supply chain design with logical constraints : formulation and solution. Computers and Operations Research, 30(14), 2135-2155.

[96] Syam, S. S. (2002). A model and methodologies for the location problem with logistical components. Computers and Operations Research, 29, 1173-1193.

[97] Syarif, A., Yun, Y., & Gen, M. (2002). Study on multi-stage logistics chain network : a spanning tree-based genetic algorithm approach. Computers and Industrial Engineering, 43, 299-314.

[98] Amiri, A. (2006). Designing a distribution network in a supply chain system : formulation and efficient solution procedure. European Journal of Operational Research, 171(2), 567-576.

[99] Gen, M., & Syarif, A. (2005). Hybrid genetic algorithm for multi-time period production/distribution planning. Computers and Industrial Engineering, 48(4), 799-809.

[100] Truong, T. H., & Azadivar, F. (2005). Optimal design methodologies for configuration of supply chains. International Journal of Production Researches, 43(11), 2217-2236.

[101] Sabri, E. H., & Beamon, B. M. (2000). A multi-objective approach to simultaneous strategic and operational planning in supply chain design. Omega, 28, 581-598.

[102] Chan, F. T. S., Chung, S. H., & Wadhwa, S. (2004). A hybrid genetic algorithm for production and distribution. Omega, 33, 345-355.

[103] Guillen, G., Mele, F. D., Bagajewicz, M. J., Espuna, A., & Puigjaner, L. (2005). Multiobjective supply chain design under uncertainty.Chemical Engineering Science, 60, 1535-1553.

[104] Fulya Altiparmak, Mitsuo Gen, Lin Lin, Turan Paksoy. (2006). A genetic algorithm approach for multi-objective optimization of supply chain networks, Computers & Industrial Engineering 51 (2006) 196-215.

[105] Michalewicz, Z., Vignaux, G. A., & Hobbs, M. (1991). A non-standard genetic algorithm for the nonlinear transportation problem. ORSA Journal on Computing, 3, 307-316.

[106] Gen, M., & Cheng, R. (2000). Genetic algorithms and engineering optimization. New York : Wiley.

[107] Fulya Altiparmak, Mitsuo Gen, Lin Lin, Ismail Karaoglan. (2007). A steady-state genetic algorithm for multi-product supply chain network design, Computers & Industrial Engineering 56 (2009) 521-537.

[108] Murata, T., Ishibuchi, H., and Tanaka, H. (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. Computers and Industrial Engineering, 30(4), 957-968.

[109] Zhou, G., and Gen, M. (1999). Genetic algorithm approach on multi-criteria minimum spanning tree problem. European Journal of Operational Research, 114, 141-152.

[110] Antonio Costa, Giovanni Celano, Sergio Fichera, Enrico Trovato. (2010). A new efficient encoding/decoding procedure for the design of a supply chain network with genetic algorithms Computers & Industrial Engineering 59 (2010) 986-999.

[111] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T,(2002). A Fast and Elitist Multiobjective Genetic Algorithm : NSGA-II , IEEE Transactions on Evolutionary Computation.

Annexe I

4.5 Résultats des expérimentations effectuées

Classe	Instance	nbr-sol	AG-1		AG-2		AG-3		AG-4	
			SN1(%)	Temps	SN2(%)	Temps	SN3(%)	Temps	SN4(%)	Temps
Classe I	Inst-1	117	35.90	21	38.46	21	58.12	150	7.69	195
	Inst-2	121	24.79	20	43.80	21	61.16	152	4.96	195
	Inst-3	159	19.50	21	25.16	22	59.12	153	11.32	198
	Inst-4	145	1.38	21	54.48	22	63.45	147	3.45	196
	Inst-5	141	24.82	20	28.37	20	60.99	150	1.42	196
	Inst-6	170	28.24	23	28.82	21	62.35	151	5.88	197
	Inst-7	85	23.53	21	32.94	21	75.29	148	8.24	195
	Inst-8	125	5.60	21	59.20	20	60.00	149	9.60	195
	Inst-9	111	22.52	21	35.14	20	81.98	150	0.90	195
	Inst-10	128	35.16	21	28.13	21	57.03	150	11.72	196
Classe II	Inst-1	132	14.39	29	28.79	27	58.33	175	25.76	244
	Inst-2	102	8.82	29	39.22	28	64.71	177	18.63	244
	Inst-3	108	13.89	30	12.04	28	87.04	176	22.22	245
	Inst-4	151	19.21	28	34.44	26	55.63	176	20.53	245
	Inst-5	140	16.43	28	27.86	25	84.29	177	5.00	242
	Inst-6	132	14.39	28	40.91	28	56.82	177	23.48	246
	Inst-7	128	21.88	31	29.69	25	72.66	175	14.84	244
	Inst-8	123	9.76	30	42.28	28	67.48	175	8.13	244
	Inst-9	128	24.22	29	2.34	28	73.44	176	33.59	243
	Inst-10	103	18.45	31	19.42	27	65.05	176	20.39	245
Classe VI	Inst-1	98	14.29	50	29.59	50	55.10	245	16.33	375
	Inst-2	105	17.14	50	31.43	46	57.14	243	20.00	374
	Inst-3	87	9.20	50	42.53	46	67.82	245	12.64	370
	Inst-4	112	8.93	51	45.54	46	57.14	240	13.39	370
	Inst-5	65	7.69	52	78.46	45	12.31	241	13.85	370
	Inst-6	114	2.63	50	33.33	49	78.07	244	6.14	369
	Inst-7	84	21.43	50	10.71	50	67.86	243	21.43	371
	Inst-8	106	30.19	47	13.21	52	61.32	247	17.92	371
	Inst-9	101	21.78	50	10.89	47	86.14	243	3.96	371
	Inst-10	78	12.82	50	24.36	51	66.67	245	23.08	370

Classe	Instance	nbr-sol	AG-1		AG-2		AG-3		AG-4	
			SN1(%)	Temps	SN2(%)	Temps	SN3(%)	Temps	SN4(%)	Temps
Classe V	Inst-1	141	5.67	196	21.28	244	65.96	254	26.95	375
	Inst-2	142	7.04	200	8.45	243	85.92	254	26.06	373
	Inst-3	139	2.88	199	18.71	242	83.45	255	12.23	369
	Inst-4	120	11.67	194	13.33	246	78.33	255	16.67	370
	Inst-5	100	10.00	195	7.00	245	96.00	252	16.00	371
	Inst-6	139	10.07	196	14.39	245	76.26	253	23.74	369
	Inst-7	125	17.60	195	11.20	245	83.20	255	15.20	373
	Inst-8	162	7.41	194	19.14	242	72.84	254	20.37	370
	Inst-9	161	8.07	195	5.59	243	78.26	254	32.92	370
	Inst-10	152	5.92	196	25.00	245	75.66	254	31.58	370

TABLE 4.4 – Tableau détaillé des résultats des expérimentations effectuées des quatre variantes proposées

Classe	Instance	nbr-sol	AG-1		AG-2	
			SN1(%)	Temps	SN2(%)	Temps
Classe I	Inst-1	158	59.49	150	78.48	200
	Inst-2	141	63.82	152	82.26	199
	Inst-3	158	59.49	151	78.48	203
	Inst-4	141	63.82	150	82.26	203
	Inst-5	150	65.33	148	79.33	200
	Inst-6	151	60.26	147	79.47	202
	Inst-7	162	72.83	149	71.60	201
	Inst-8	116	69.82	150	86.20	198
	Inst-9	146	65.06	151	81.50	201
	Inst-10	142	64.08	149	81.69	201
Classe II	Inst-1	126	73.02	176	79.37	247
	Inst-2	131	82.44	175	87.79	244
	Inst-3	123	73.17	173	91.06	245
	Inst-4	112	83.93	179	85.71	243
	Inst-5	123	75.61	180	86.18	246
	Inst-6	137	73.72	175	88.32	243
	Inst-7	131	79.39	177	89.31	244
	Inst-8	153	71.90	178	86.93	243
	Inst-9	133	80.45	176	86.47	245
	Inst-10	116	88.79	175	76.72	246
Classe VI	Inst-1	96	73.96	188	85.42	292
	Inst-2	139	76.26	188	75.54	292
	Inst-3	100	83.00	186	83.00	291
	Inst-4	99	79.80	183	78.79	296
	Inst-5	130	71.54	188	90.77	291
	Inst-6	125	70.40	185	88.00	290
	Inst-7	79	81.01	188	84.81	297
	Inst-8	100	77.00	187	88.00	290
	Inst-9	115	66.96	189	88.70	291
	Inst-10	95	84.21	188	85.26	293
Classe V	Inst-1	125	88.00	245	92.00	310
	Inst-2	133	83.46	247	90.98	309
	Inst-3	160	76.88	241	88.13	315
	Inst-4	127	87.40	240	88.98	313
	Inst-5	116	81.03	244	92.24	312
	Inst-6	136	77.94	245	91.91	313
	Inst-7	146	74.66	245	91.78	314
	Inst-8	117	74.36	246	89.74	309
	Inst-9	125	80.80	243	93.60	311
	Inst-10	163	81.60	245	87.12	313

TABLE 4.5 – Résultats des expérimentations effectuées comparant AG-3 et NSGA II

Annexe II

4.6 Implementation

L'application est mise en oeuvre par le langage C++ basée sur la notion de pointeurs. Cette notion permet de définir des structures dynamiques évoluant au cours du temps (par opposition aux tableaux, par exemple, qui sont des structures de données statiques dont la taille est figée a priori).
A l'exécution du programme, celui-ci est stocké en mémoire. Plus précisément, chaque variable que l'on a définie possède une zone de mémoire qui lui est réservée, et la taille de cette zone correspond au type de variable que l'on a déclaré. Rappelons que la mémoire est constituée de plein de petites cases de 8 bits (un octet). Une variable, selon son type (où sa taille), va ainsi occuper une ou plusieurs de ces cases (une variable de type char occupera une seule case, tandis qu'une variable de type long occupera 4 cases consécutives). Chacune de ces cases (appelées blocs) est identifiée par un numéro. Ce numéro est appelé adresse et l'accès a une variable se fait de deux façons : soit par son nom ou par l'adresse du premier bloc alloue a la variable.
Il suffit donc de stocker l'adresse de la variable dans un pointeur afin de pouvoir acceder a celle-ci (dans notre cas toutes les adresses des sommets pendants de l'arbre sont stockées dans une liste chaînée).
L'utilisation des pointeurs à un grand nombre d'intérêts :

- Elle permet de manipuler de façon simple des données pouvant être importantes (au lieu de passer a une fonction un element très grand (en taille) on pourra par exemple lui fournir un pointeur vers cet element...)
- En stockant des pointeurs dans les cases d'un tableau, il sera possible de stocker des elements de taille diverse, et même de rajouter des elements au tableau en cours d'utilisation (tableau dynamique), a l'inverse des tableaux statiques qui ne permettent de stocker qu'un nombre fixe d'elements de mémé type.

La plate-forme d'implementation utilisée est **l'Embarcadero C++Builder XE3**. Cette plate-forme est une solution standard pour développer avec une seule base de code des applications fonctionnant a la fois sous Windows et Mac OSX.

4.6.1 Motivation du choix de la plate-forme

Avec C++Builder XE3, les applications sont développées en C++ dans le framework multi systèmes avant de les compiler pour différentes plate-forme. Cette plate-forme offre plusieurs avantages, nous citons entre autres :

- C++ Builder XE3 prend en charge les derniers standards C et C++ y compris C++98, C++TR1 et C++11, en complement d'ANSI C, ISO C, C99 et C11.
- C++Builder XE3 prend en charge des environnements 32 et 64 bits, ce qui permet de développer des applications plus puissantes (prenant en charge une mémoire étendue et les sous-systèmes 64 bits) tout en criblant la base de systèmes d'exploitation 32 et 64 bits.
- Les applications développées en C++Builder XE3 fonctionnent a pleine vitesse directement sur le CPU de l'appareil (et non dans un moteur de script ou une machine virtuelle).

Toutes les informations complémentaires liées aux avantages de la plate-forme et son utilisation peuvent être consultées sur le site officiel http ://www.embarcadero.com.

4.6.2 Présentation du logiciel

4.6.2.1 Interface

L'interface de l'application implémentée est subdivisée en 2 parties principales 4.4, permettant de manipuler et d'évaluer les performances des variantes de l'algorithme génétique pour la résolution du problème considéré.

Partie 1 : Dans cette partie nous avons un écran d'affichage des résultats après exécution des différentes variantes de l'Algorithme Génétique implémenté.

Partie 2 : Cette partie est dédiée aux différentes manipulations permettant de calculer les ensembles de solutions non dominées des différentes variantes implémentées.

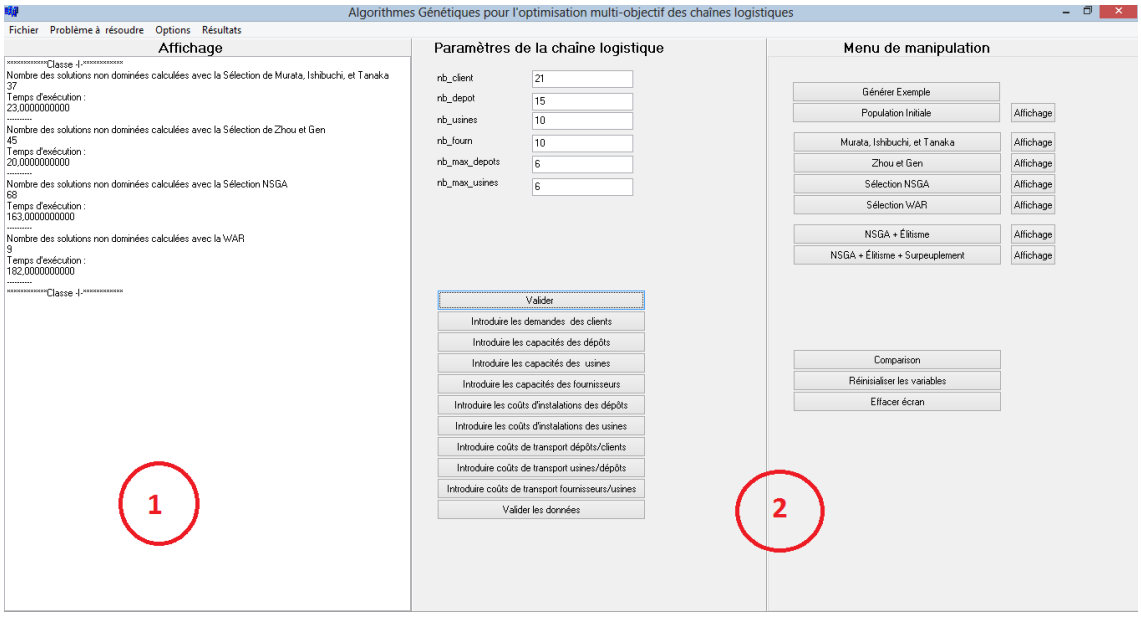
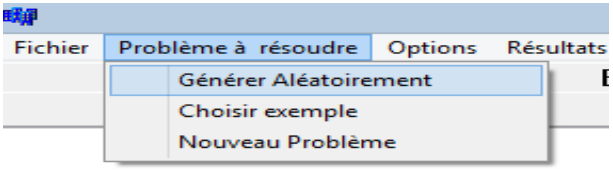


FIGURE 4.4 – Interface de l'application implémentée

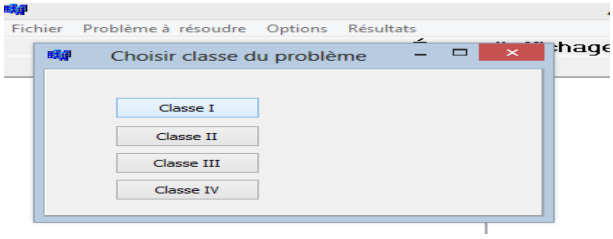
4.6.2.2 Manipulations

Pour manipuler l'application développée nous avon trois façons, générer un problème aléatoirement, traiter un exemple déjà enregistré ou introduire un nouveau problème.

1- Générer un problème aléatoirement

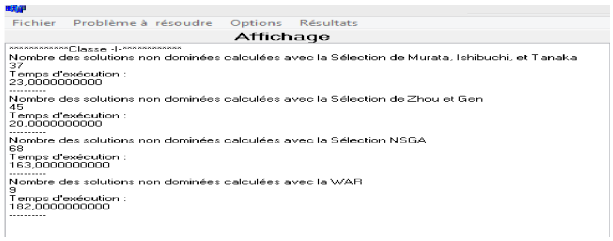


(a) Manipulation générer un problème



(b) Choisir la classe du problème

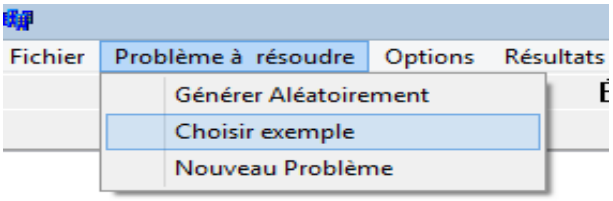
(c) Exécuter les quatres variantes



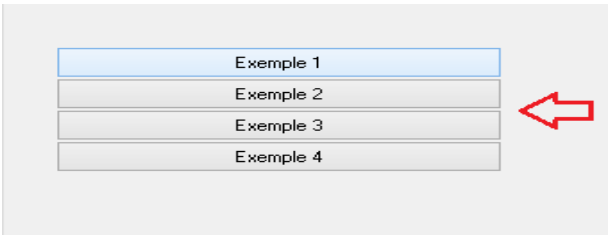
(d) Affichage des résultats

FIGURE 4.5 – Résolution d'un problème généré aléatoirement

2- Traiter un exemple :

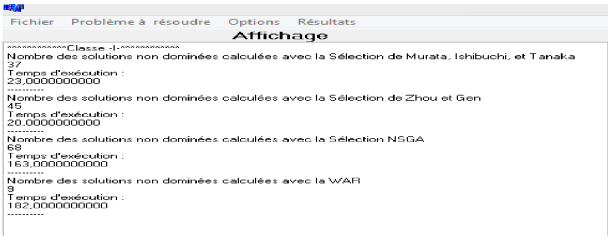


(a) Manipulation choisir exemple



(b) Choisir un exemple enregistré

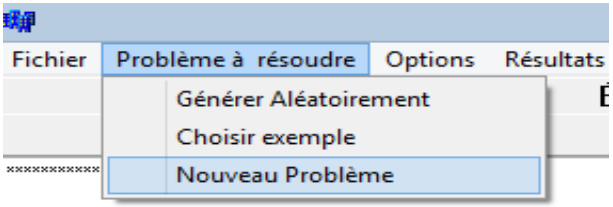
(c) Exécuter les deux variantes



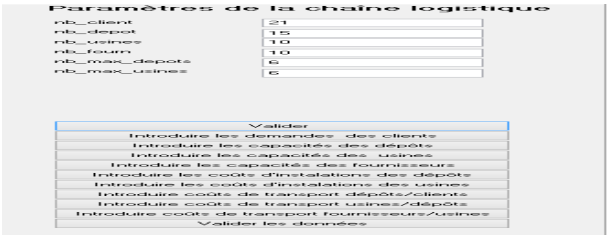
(d) Affichage des résultats

FIGURE 4.6 – Résolution d'un exemple enregistré

3- Introduire un nouveau problème

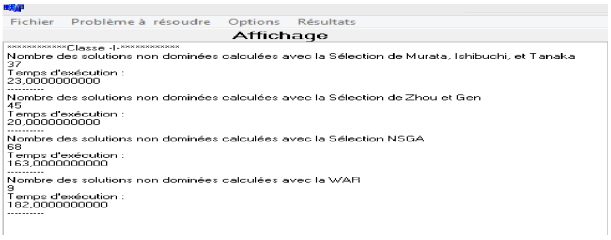


(a) Manipulation résoudre nouveau problème



(b) Introduire les données du problème

(c) Exécuter les deux variantes



(d) Affichage des résultats

FIGURE 4.7 – Résolution d'un nouveau problème