UNIVERSITE DE BLIDA 1

Faculté de Technologie
Département d'Electronique.

MEMOIRE DE MAGISTER

Spécialité : Electronique
Option: Signaux et Systèmes

OPTIMISATION MULTICRITERE PAR L'HYBRIDATION D'UN ALGORITHME GENETIQUE ET UN ALGORITHME DE COLONIE DE FOURMIS.

Par

MAHIOUT Elhadj

Devant le jury composé de

H. SALHI	Professeur, U.de blida 1	Président
F.BOUDJEMA	Professeur à L'ENP	Examinateur
Z.A. BENSELAMA	Maitre de conférence, U.de blida 1	Examinateur
A .GUESSOUM	Professeur,U.de blida 1	Rapporteur
M.HADJ SADOK	Maitre de conférence, U.de blida 1	Co-rapporteur
M.A. BENCHERCHALI	,MAA, U. de Blida 1	Invité

Blida, Octobre 2016

REMERCIEMENTS

Je tiens à remercier tout qui ont contribué par leurs encouragements, chacun à sa façon, à l'atteinte de l'objectif que constitue la réalisation de cette recherche et le dépôt de ce mémoire, je cite en premier lieu mes parents ainsi mes frères et sœurs et mes amies.

Je tiens à remercier sincèrement mon encadreur de recherche monsieur **GUESSOUM Abderezzak** professeur à l'université de BLIDA 1**,** pour sa grande disponibilité tout au long de ce travail. Ses conseils et son expérience m'ont permis d'acquérir les connaissances indispensables à l'aboutissement de ce travail.

Je remercie également monsieur **HADJ SADOK Mhamed** maitre de conférence à l'université de BLIDA 1, mon encadreur assistant pour son support, ses propositions, et ses encouragements afin d'aboutir à la réalisation de ce travail.

Mes remerciements sont aussi destinés à monsieur **BOUDJEMA Fares** professeur à l'école national polytechnique L'ENP, et à monsieur **BENSELAMA Zoubir** maitre de conférences à l'université de BLIDA 1, qui ont accepté d'examiner mon travail.

Je tiens également ma profonde reconnaissance pour les enseignants : **KAZED, SALHI, MAMOUN, BENSEBTI, BOUGHERIRA, CHERFA, ASSAD, CHEIKHI** et beaucoup d'autres, ainsi que pour tout le staff administratif du département d'électronique - université de BLIDA1-, pour leurs sincère contribution à ma formation ; durant le cycle des études universitaires appliquées ainsi que celui de l'ingéniorat, et de magister.

Finalement, à tous les autres qui ont gravité autour de moi pendant ces dernières années, et qui m'ont supporté et encouragé, je leurs dis un gros merci. J'espère que ce travail saura combler leurs attentes.

TABLE DES MATIERES

ملخص

ترتكز هذه المذكرة حول حل المشاكل التوافقية من مجمل مشاكل إيجاد أفضل الحلول، ودرسنا كمثال على هذه المشاكل الحقيبة المتعددة الأهداف والأبعاد.

قمنا في **الجزء الأول** من هذه المذكرة بتقديم تعاريف حول مشاكل إيجاد أفضل الحلول المتعددة الأهداف منها وكذا مقاربات حل هذه المشاكل.

في **الجزء الثاني** قمنا بتعريف المشاكل التوافقية وتعقيدها النظري، وقمنا باقتراح خوارزمية هجينة لمشكلة الحقيبة المتعددة الاهداف والأبعاد و المتمثلة في **GAS^{MOKP}** و التي تم تصميمها من خلال الجمع بين **خوارزميتين**: الخوارزمية **الجينية** و**خوارزمية مستعمرة النمل.**

في **الجزء الثالث** قدمنا النتائج التي تم الحصول عليها في المحاكاة، وكانت النتائج جد مرضية، كون خوارزمية **GAS^{MOKP}** تعطي نتائج في فترة زمنية جد معقولة و ذات نوعية ممتازة.

RESUME

Cette mémoire porte sur la résolution des problèmes d'optimisation combinatoires, on a étudié comme exemple de ces problèmes le sac à dos multidimensionnel multiobjectif.

Nous avons donnés un état de l'art de l'optimisation multiobjectif dans la première partie de cette mémoire, nous avons ainsi présentés les principales approches de résolutions de ces problèmes.

Dans la deuxième partie on a définis les problèmes combinatoires et leur complexité théorique, et on a proposé un algorithme hybride pour le problème de sac à dos multidimensionnel multiobjectif, il s'agit de **GAS^{MOKP}** conçu en combinant deux métaheuristique ; l'algorithme génétique et l'algorithme de colonie de fourmis.

Dans la troisième partie on a présenté les résultats obtenus lors de la simulation, ces résultats ont été jugés très satisfaisants, puisque notre **GAS^{MOKP}** converge en un temps de calcul très raisonnable vers des solutions de qualités supérieurs

Summary:

This memory focuses on solving combinatorial optimization problems; we have studied as an example of these problems the multidimensional knapsack multiobjective.

We have given definitions around multi-objective optimization in the first part of this memory; we have presented the main approaches and resolutions of these problems.

In the second part we have defined combinatorial problems and their theoretical complexity, and we have proposed a hybrid algorithm for the problem of multidimensional knapsack multiobjective; the **GAS^{MOKP}**, it was designed by combining two metaheuristics ; the **Genetic Algorithm** and **Ant Colony Algorithm**.

In the third part we have presented the results obtained in the simulation, the results were highly satisfactory, since our **GAS^{MOKP}** converges in a very reasonable time computing, to solutions of superior quality.

Mots-clés : optimisation, problèmes multiobjectif, algorithmes évolutionnaires, recherche locale, approches hybrides, approches approchées, combinatoire, optimum, surface de compromis, population, individu, intensification, diversification, convexité, contrainte.

INTRODUCTION GENERALE

Présentation

Les problèmes d'optimisation sont souvent rencontrés dans de nombreux domaines et secteurs de l'économie (industrie, recherche scientifique, finance, agriculture …). La résolution de ces problèmes consiste à trouver la ou les meilleures solutions qui minimisent (resp-maximisent) une ou plusieurs fonctions objectifs tout en vérifiant un ensemble de contraintes simultanément,… et pour cela leurs résolution n'aboutisse pas à une solution unique mais à un ensemble de meilleurs solutions ce qu'on l'appel (ensemble de compromis). ces problèmes sont multicritère ou multiobjectif car ils intègrent plusieurs objectifs .

Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes appliqués: le traitement d'images, la conception de systèmes, la conception d'emplois du temps, . . . ou la majorité de ces problèmes sont qualifiés de difficiles, car leur résolution nécessite l'utilisation des algorithmes évolués, et il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable. Lorsqu'un seul critère est donné, par exemple un critère de minimisation de cout, la solution optimale est clairement définie, c'est celle qui a le cout minimal. Mais dans de nombreuses situations, un seul critère peut être insuffisant.

En effet, la plupart des applications traitées intègrent plusieurs critères simultanés souvent contradictoires .En effet Intégrer des critères contradictoires pose un réel problème.

Prenant l'exemple d'une personne qui veut acheter une maison bien située et d'un prix raisonnable ; cette exemple représente un problème d'optimisation à 02 critères, on remarque que ces 02 critères sont contradictoires, en effet une maison bien située correspond à un cout plus élevé, donc l'acheteur doit définir l'ensemble de solutions (compromis) possibles et choisir une qui convient à son budget.

L'utilisation des métaheuristique pour résoudre des problèmes multi-objectifs a fait l'objet d'un intérêt de plus en plus croissant. A présent, les métaheuristiques constituent un des champs de recherche les plus actifs dans l'optimisation multi-objectifs. Nous proposons, dans ce mémoire, un algorithme hybride composée de deux métaheuristique différentes ; un algorithme génétique et un algorithme de colonie de fourmis, pour les problèmes d'optimisation multicritères.L'objectif de ce travail était d'étudier la capacité de la métaheuristique ACO (algorithme de colonie de fourmis) combinée à un algorithme génétique à améliorer les résultats obtenus par ces algorithmes appliquées séparément.

En effet plusieurs algorithmes sont conçus en se basant sur l'hybridation de deux méthodes différentes dans le but de fournir des résultats supérieurs aux méthodes qui les composent , et produire des solutions de meilleure qualité possible avec un temps de calcul plus raisonnable, on cite l'algorithme **GTS** $^{\textbf{MOKP}}$ (Genetic + Tabu search), qui combine une méthode de recherche **Tabou** et une méthode évolutionnaire, chaque composant ayant une tache déterminée :**intensifier** ou **diversifier** ;
un algorithme **PICPA** combine des techniques de propagation de contraintes et des concepts évolutionnaires,,ces deux algorithmes sont proposés par **Vincent BARICHARD -2003-** dans le cadre de sa thèse de doctorat .

Jaszkiewicz a proposé en 2002 l'algorithme **"Multi-Objective Genetic Local Search" (MOGLS)** qu'il a appliqué au problème de voyageur de commerce multi-objectif et au problème du sac à dos multi-objectif. Cet algorithme utilise une méthode de descente pure pour l'intensification et une méthode évolutionnaire pour la diversification .

Dans le cadre de ce mémoire nous proposons un algorithme hybride **GAS** $^{\textbf{MOKP}}$ qui combine un algorithme génétique pour diversifier et un algorithme de colonie de fourmis pour intensifier la recherche ,dans le but de résoudre le problème le du sac a dos multidimensionnel multiobjectif.

PARTIE I
GENERALITES SUR L'OPTIMISATION

Introduction

Dans cette partie on va donner un état de l'art des problèmes d'optimisation, une présentation de problèmes d'optimisation multiobjectif (multicritère) qui sera le cadre de travail de ce mémoire, nous décrivons ainsi ses concepts fondamentaux, et les principales approches de la résolution de ces problèmes. Nous présenterons ensuite **la métaheuristique de l'algorithme génétique** et **la métaheuristique de l'algorithme de colonie de fourmis**, pour pouvoir les exploiter à la conception d'un algorithme hybride pour la résolution du problème de **sac à dos multidimensionnel multiobjectif.**

1.1/Optimisation multicritère

Les problèmes d'optimisation multi-objectifs sont une généralisation à **m fonctions objectifs** de problèmes d'optimisation mono-objectif. La solution de ces problèmes n'est pas unique comme dans le cas de l'optimisation mono-objectif mais un ensemble de compromis. Avant de définir l'optimisation multicritère on va présenter d'abord l'optimisation mono-objectif ou monocritère.

Comme il indique son nom l'optimisation monocritère consiste à trouver la solution qui minimise un seul critère prédéfinit, le problème est formulé comme suit :

Soit $f : \mathbb{R}^n \to \mathbb{R}$,
et soit $\mathcal{X}$ un ensemble fermé de $\mathbb{R}^n$
alors l'ensemble des minimiseurs est : $\hat{\mathcal{X}} = \{\overrightarrow{x} \in \mathcal{X} | \forall \overrightarrow{x'} \in \mathcal{X},\ f(\overrightarrow{x'}) \geq f(\overrightarrow{x})\}$
et le minimum du problème est : $f(\hat{\mathcal{X}})$

L'optimum est souvent un point unique $f(\overrightarrow{x})$

Partant de ça, on peut définir maintenant les problèmes d'optimisation multiobjectif qui sont une multitude de **m** fonctions objectifs à optimiser à la fois, ces problèmes on peut les définir formellement comme suit :

Soit $f : \mathbb{R}^n \to \mathbb{R}^m$,
et soit $\mathcal{X}$ un ensemble fermé de $\mathbb{R}^n$
alors l'ensemble des minimiseurs est :

$$\hat{\mathcal{X}} = \left\{\overrightarrow{x} \in \mathcal{X} | \forall \overrightarrow{x'} \in \mathcal{X}, \left(\forall i \in [1, \cdots, m]\ f_i(\overrightarrow{x}) < \text{ ou } = f_i(\overrightarrow{x'})\right) \vee \left(\exists i \in [1, \cdots, m]\ f_i(\overrightarrow{x}) < f_i(\overrightarrow{x'})\right)\right\}$$

et le minimum du problème est : $f(\hat{\mathcal{X}})$

L'ensemble $\mathcal{X}$ représente l'ensemble **des solutions potentielles du problème** ; Cet ensemble est déterminé à l'aide de contraintes décrites dans l'énoncé du problème.

Si on fait intervenir les contraintes (**q contraintes**), un problème d'optimisation multiobjectif est plus souvent définit sous la forme suivante :

Minimiser	$\vec{f}(\vec{x})$	(m fonctions à optimiser)
tel que	$\vec{g}(\vec{x}) \leq 0$	(q contraintes à satisfaire)
avec	$\vec{x} \in \mathbb{R}^n,\ \vec{f}(\vec{x}) \in \mathbb{R}^m,\ \vec{g}(\vec{x}) \in \mathbb{R}^q$	

Pour résoudre un problème d'optimisation multicritère, il faut déjà être capable de déterminer l'ensemble des solutions potentielles $\mathcal{X}$.

On définit ainsi :

L'espace de recherche: représente l'ensemble des valeurs pouvant être prises par le variable $\vec{x}$

L'espace réalisable : représente l'ensemble des valeurs prises par la variable $\vec{x}$ satisfaisant les contraintes.

Vecteur de décision: C'est le vecteur $\vec{x}$, Il représente l'ensemble des variables du problème.

Fonction objectif: la fonction f (c'est **la fonction de cout** ou **critère d'optimisation**).

1.2/Relations d'ordre et de dominance

Comme la résolution d'un problèmes d'optimisation multicritères s'aboutis à un ensemble de compromis (multitude de points de $\mathbb{R}^n$) , donc il faut d'abord définir les relations d'ordre entre les solutions pour Identifier les meilleurs compromis.

Les relations d'ordre sont appelées aussi **relations de dominance.**

La relation de dominance la plus célèbre et la plus utilisée est **la dominance au sens de Pareto** , C'est celle-ci que nous allons définir et utiliser dans la suite de cet mémoire… ;

Soient u et v, deux vecteurs de même dimension :
les relations = , < ou = , et < usuelles sont étendues aux vecteurs.

$$\begin{aligned} \mathbf{u} = \mathbf{v} \quad & ssi \quad \forall i \in \{1, 2, \cdots, m\},\ u_i = v_i \\ \mathbf{u} \leq \mathbf{v} \quad & ssi \quad \forall i \in \{1, 2, \cdots, m\},\ u_i \leq v_i \\ \mathbf{u} < \mathbf{v} \quad & ssi \quad \mathbf{u} \leq \mathbf{v} \wedge \mathbf{u} \neq \mathbf{v} \end{aligned}$$

Les relations $\geq$ et $>$ sont définies de manière analogue.

L'inconvénient des relations définies précédemment est qu'ils ne couvrent pas tous les cas possibles. En effet, il est impossible de classer 02 points : **a** = (3; 2) et **b** = (4; 1) à l'aide d'une de ces relations, et pour cela nous définissons la relation de dominance **au sens de Pareto** permettant de prendre en compte tous les cas de figures rencontrés lors de la comparaison de deux points (solutions) ; ici notés des vecteurs.

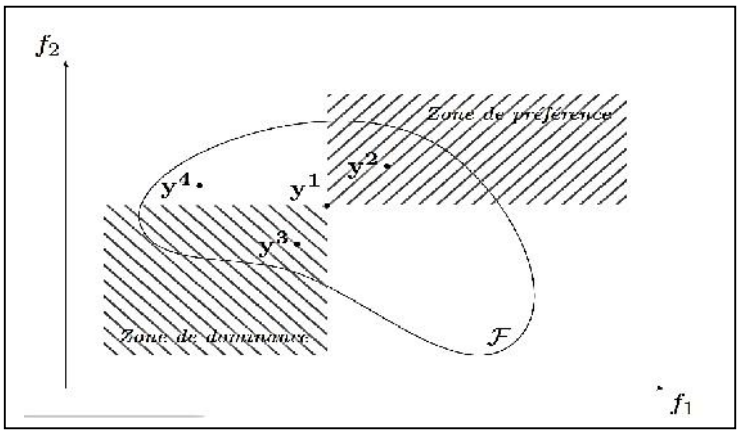
La dominance au sens de Pareto :

Soient u et v deux vecteurs de décision :

$$\begin{array}{rcl} \mathbf{u} \prec \mathbf{v}\ (\mathbf{u}\ \textit{domine}\ \mathbf{v}) & \textit{ssi} & \mathbf{f}(\mathbf{u}) < \mathbf{f}(\mathbf{v}) \\ \mathbf{u} \preceq \mathbf{v}\ (\mathbf{u}\ \textit{domine faiblement}\ \mathbf{v}) & \textit{ssi} & \mathbf{f}(\mathbf{u}) \leq \mathbf{f}(\mathbf{v}) \\ \mathbf{u} \sim \mathbf{v}\ (\mathbf{u}\ \textit{est incomparable (ou non-dominé) avec}\ \mathbf{v}) & \textit{ssi} & \mathbf{f}(\mathbf{u}) \npreceq \mathbf{f}(\mathbf{v}) \wedge \mathbf{f}(\mathbf{v}) \npreceq \mathbf{f}(\mathbf{u}) \end{array}$$

Pour un problème de maximisation, ces relations sont définies de manière symétrique.

Dans la figure suivante on peut illustrer cette relation :



Exemple de dominance

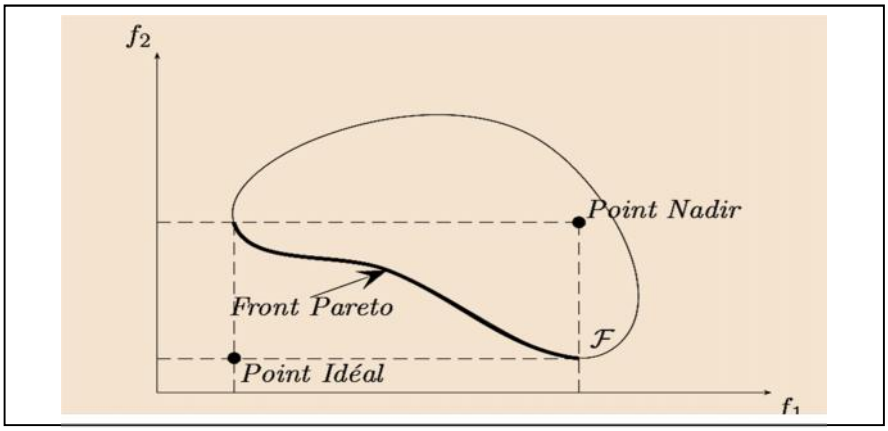
- $\mathcal{F}$ représente l'espace réalisable dans l'espace des objectifs c'est l'image de $\mathcal{X}$.
- chaque point $\mathbf{y^i}$ est l'image de $\mathbf{x^i}$ par $f : \mathbf{y^i} = f(\mathbf{x^i})$.

Si on Prend le point $\mathbf{y^1}$ comme point de référence. Nous pouvons distinguer trois zones :

Une zone de préférence c'est la zone contenant les points dominés par $\mathbf{x^1}$,
Une La zone de dominance c'est la zone contenant les points dominant $\mathbf{x^1}$,
Une La zone d'incompatibilité c'est la zone qui contient les points incomparables avec $\mathbf{x^1}$.

Ainsi, il est clair que $\mathbf{x^2}$ est dominé par $\mathbf{x^1}$ ($\mathbf{x^1}$ est préféré à $\mathbf{x^2}$), que $\mathbf{x^3}$ domine $\mathbf{x^1}$ ($\mathbf{x^3}$ est préféré à $\mathbf{x^1}$), et que $\mathbf{x^4}$ est non dominé (incomparable) avec $\mathbf{x^1}$.

A ce moment on peut définir l'ensemble de solutions (tous les compromis) d'un problème d'optimisation multicritère comme étant, **l'image** de l'ensemble de solutions non encore dominées, à l'égard de différents fonctions objectifs f constituants le problème. C'est le **front Pareto** .



Le front Pareto

Cette figure représente un front Pareto d'un problème d'optimisation à 02 objectifs (bi-objectifs).

On va définir par la suite **le point idéal** et **le point nadir** ; ces 02 points servent - respectivement- comme pôles **d'attraction**, **répulsion** de la recherche.

Le point Idéal

Ce point correspond aux meilleures valeurs de chaque objectif des points du front Pareto. Les coordonnées de ce dernier correspondent aux valeurs obtenues en optimisant chaque fonction objectif séparément.

Le point Nadir

Ce point correspond aux pires valeurs de chaque objectif des points du front Pareto.

1.3/Approches de résolution

Il existe plusieurs approches et méthodes conçus pour la résolution des problèmes d'optimisation multicritères Certaines approches fixent des préférences sur des critères, d'autres mettent tous les critères au même niveau d'importance.. , On distingue 03 catégories de ces approches :

01-les approches exactes

Ils examinent l'intégralité de l'espace de recherche et pour cela ils nécessitent des algorithmes **plus complexes** et leurs **temps de convergence** est moins raisonnable.

NB: en entend par le temps de convergence, la durée de temps mise par l'algorithme (la méthode) pour qu'il s'aboutisse à la résolution du problème.

02-les approches approchées

On cite dans cette catégorie d'approches les différentes métaheuristiques (**les algorithmes génétiques** , **les algorithmes de colonies de fourmis** , **les**

particules swarm…), ces approches n'examinent pas la totalité de l'espace de recherche, ils ont la propriété de concenter la recherche autour des zones spécifiques de l'espace de recherche globale, ces méthodes ne convergent pas souvent vers l'optimum de problème mais elles cherchent avant tout à produire une solution sous-optimale de meilleure qualité possible avec un temps de calcul plus raisonnable .

03-les approches hybrides

Ces méthodes sont conçues tout en hybridant deux différentes méthodes **(approches exactes et approches approchées)** ; et cela dans l'objectif d'atteindre des résultats supérieurs à ceux obtenus par les deux méthodes qui les composent appliquées seulement.

Dans le cadre de ce mémoire, nous proposerons une méthode hybride ; conçu en combinant deux **approches approchées (métaheuristique)** : **Algorithme génétique et Algorithme de colonie de fourmis**, pour la résolution du problème de sac à dos multidimensionnel multiobjectif, et on la dénomme **le GAS ^{MOKP}**.

Par la suite, on s'intéresse à la présentation des métaheuristiques (approches approchées) ; on va donner un état de l'art des algorithmes génétiques et les algorithmes de colonie de fourmis.

1.4/Les algorithmes génétiques

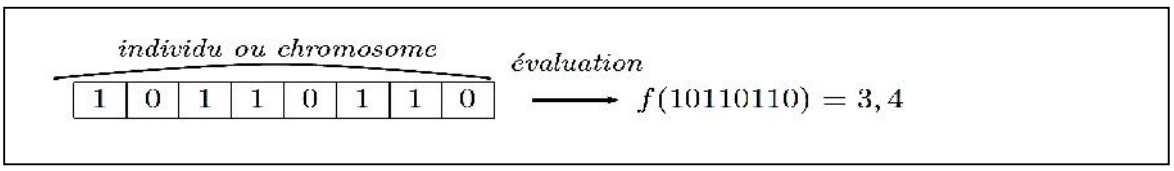
Les algorithmes génétiques **(AGs)** ont été introduits par Holland [Holland, 1975].ils sont typiquement constitués de de trois éléments fondamentaux :

01-Une population

Constituée de plusieurs **individus** représentants des solutions potentielles **(configurations)** du problème donné, l'individu représente un codage de l'information sous forme de chaines binaires (0/1) de longueur fixé ; sous ce codage l'individu est appelé aussi **un chromosome.**

02-Un mécanisme d'évaluation

Consiste à transformer la chaine binaire de 0/1 du chromosome en une valeur réelle, appelée **valeur d'adaptation** ou **valeur de profit** ou **fitness,** et pour cela on fait appel à **une fonction d'évaluation** ou bien **une fonction de cout ,** celle-ci dépend principalement du problème à résoudre .
La figure suivante représente un individu sous forme binaire (de 8 bits), c'est le codage de l'information.



Un individu codé sous forme binaire

03-Un mécanisme d'évolution **:**

Ce mécanisme permet, Grace à des opérateurs prédéfinis à partir d'une population initiale, d'avoir des populations plus évolués, tout en éliminant certains individus et d'en créer de nouveaux. Après une succession de générations (itérations de l'algorithme), on aura une population finale évoluée.
Le mécanisme d'évolution s'appuie sur un ensemble d'opérateurs génétiques **: la sélection , le croisement ,et la mutation.**

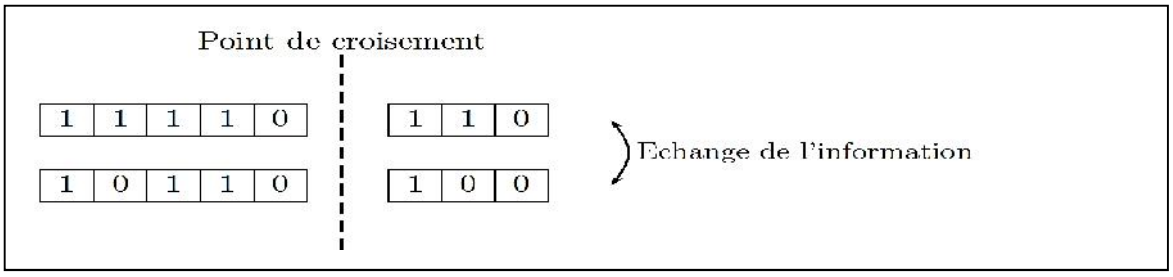
La sélection

A pour objectif de sélectionner les meilleurs individus (généralement la moitié de la population), et on va l'appeler **population intermédiaire** .les individus de la population intermédiaire peuvent se coupler entre eux **deux à deux** pour créer des nouveaux individus, c'est l'opérateur de **croisement**.

Le croisement

Ou **la recombinaison** permet de construire de nouveaux individus enfants à partir des individus parents sélectionnés précédemment dans la population intermédiaire.

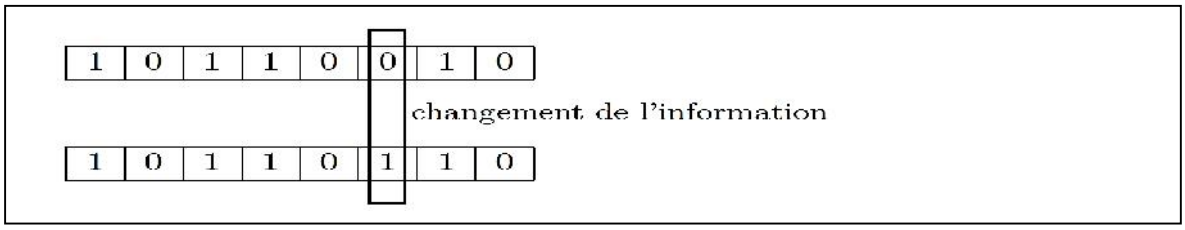
Dans la figure suivante on présente un exemple **de croisement** mono-point,qui consiste à échanger les segments (à droite) déterminés par le point de croisement des deux parents.



L'opérateur de croisement.

La mutation

Cet opérateur permet d'effectuer des légères modifications sur certains individus, (Les bits à modifier sont souvent choisies aléatoirement).



L'opérateur de mutation.

Ainsi, la nouvelle population construite à partir des opérateurs génétiques présentés devient la population de référence de la prochaine génération. Ce cycle d'opérations continue tant que la méthode n'a pas rencontré une condition d'arrêt défini préalablement : un nombre maximal de générations par exemple ou un critère prédéfinit atteint.

Les algorithmes génétiques, ont permis de résoudre un grand nombre de problèmes, notamment en optimisation.

1.5/Algorithme de colonie de fourmis

Cette métaheuristique a été initialement introduite par Dorigo, Maniezzo et Colorni [Dorigo, 1992, Dorigo et al. , 1996] .

L'optimisation par colonie de fourmis a été conçue pour résoudre le problème du voyageur de commerce en proposant le premier algorithme **ACO**. Par la suite, un nombre considérable d'applications de l'ACO a été proposé.

Ici, on va présenter **l'analogie biologique** à partir de laquelle ACO a été inspirée.

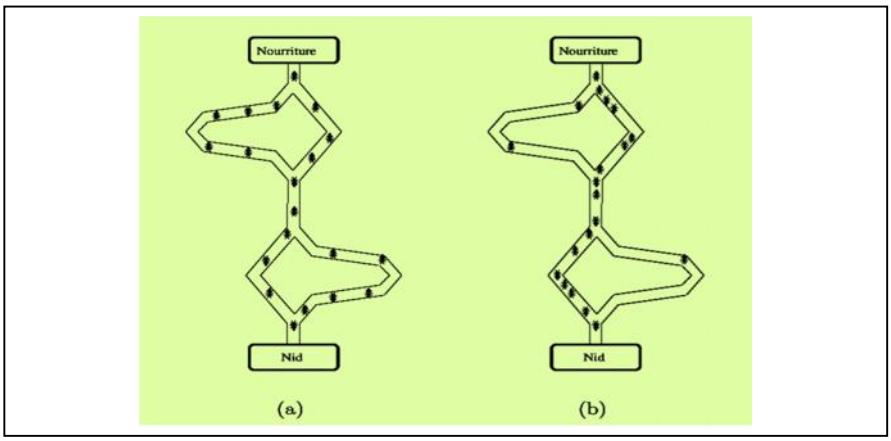
Analogie biologique

Deneubourg et al [Deneubourg et al. ,1983] ont constatés que les fourmis, résolvent des problèmes relativement complexes. Les fourmis ont la particularité de communiquer entre eux grâce à des substances volatiles appelées **phéromones** qu'elles les déposent au sol en marchant, en effet les fourmis sont attirées par ces substances, en les percevant grâce à des récepteurs situées dans leurs antennes.

Les fourmis forment des pistes de phéromone pour marquer leur trajet, par exemple entre le nid et la source de nourriture ; Le comportement collectif émerge d'un processus **autocatalytique** où plus les fourmis suivent une trace, plus cette trace devient **attirante** ; d'un autre côté et de moment que les phéromones déposées ont la propriété de **se vaporiser** avec le temps, les traces qui sont moins suivies par les fourmis devient **moins attirante** ; c'est le principe **de stigmergie.**

En effet les fourmis les plus rapidement arrivées au nid, après avoir visité la source de nourriture, sont celles qui empruntent les deux branches les plus courtes. Et donc durant une période définit le nombre de fourmis qui suivent les deux branches les plus courtes est beaucoup plus grand que le nombre de ceux qui suivent les deux branches les plus longues, et donc la quantité de phéromone qui va se présenter sur les plus courts trajets (les deux plus courtes branches) sera plus importante que celle présente sur les longs trajets (les deux plus longues branches). Or, une piste présentant une plus grande concentration en phéromone sera plus attirante pour d'autres fourmis, par conséquent à la limite ; les deux courts trajets seront choisis par la grande majorité des fourmis.

En se basant sur ce principe une colonie de fourmis sera ainsi capable de choisir (sous certaines conditions) le plus court chemin du nid vers une source de nourriture.



Principe de stigmergie chez les fourmis

le comportement des fourmis est transposé par inspiration à un algorithme général d'optimisation, et pour cela on va faire une analogie entre les solutions prises par les fourmis (les trajets suivies) en cherchant de la nourriture et l'ensemble des solutions admissibles du problème (l'espace de recherche du problème), et entre la quantité (la qualité) de la nourriture et la fonction objectif à optimiser, et enfin entre les traces de phéromones et la mémoire adaptative de l'algorithme.

La métaheuristique de colonie de fourmis pour les problèmes d'optimisation ACO

Si on considère le problème d'optimisation (S, f, Ω) où :
S est l'ensemble des solutions candidates.
f est la fonction objective qui associe à chaque solution candidate $s \in S$ une valeur de la fonction objectif f (s).
Ω et est l'ensemble des contraintes.
L'objectif est de trouver une solution optimale globale $\mathbf{s_{opt}} \in S$ qui est une solution minimisant f et qui satisfait les contraintes.

en se déplaçant, Les fourmis artificielles construisent leurs solutions sur un graphe construit G(C,L) ou :
Les composants **C** sont les sommets du graphe.
L représente l'ensemble qui connecte les composants de **C** (**L** est une connexion entre deux sommets). Les contraintes du problème sont exigées (en rejetant les solutions qui violent les contraintes).

Dans les algorithmes ACO, les fourmis se résolvent un problème d'optimisation en se basant sur la quantité des traces de phéromone τ peuvent être déposées sur **les composants C** ou sur **les connexions du graphe L** (τ_i pour un composant, τ_{ij} pour une connexion), et d'une information heuristique spécifique au problème traité représentée par **le facteur heuristique** η (η_i, η_{ij} respectivement), qui désigne une information sur l'instance du problème ou une information sur le temps d'exécution fourni par une source autre que les fourmis.

Dans la plupart des cas, η est le coût ou une estimation du coût de l'extension de l'état courant. **Des coefficients** α et β permettent de contrôler l'importance relative des deux éléments (la quantité des traces de phéromone déposée τ et le facteur heuristique η).

Une fois qu'une fourmi construise une solution, ou pendant la construction d'une solution, la fourmi évalue la solution (partielle) et dépose la phéromone sur **le composant** ou **la connexion** qu'elle l'a utilisée. Cette information de phéromone sert à enrichir la mémoire qui dirigera la recherche des futures fourmis afin de trouver des solutions beaucoup plus **admissibles**.

L'intensité de ces traces décroît dans le temps par un facteur constant appelé **coefficient d'évaporation**. Pratiquement, l'évaporation de phéromone est nécessaire pour éviter une convergence prématurée de l'algorithme vers une région sous-optimale. Ce processus implémente une forme utile d'oubli favorisant l'exploration de nouvelles aires de recherche.

Conclusion

Dans cette partie, on a présenté les métaheuristique des algorithmes génétiques et de l'algorithme de colonie de fourmis, ces approches approchées sont utilisées pour résoudre les problèmes d'optimisations multicritères, dans notre travail nous nous intéressons à ces deux métaheuristiques afin de concevoir un algorithme hybride les renfermant les deux à la fois , dans le but de résoudre le problème de sac à dos multidimensionnel multiobjectif le (**MOKP)**, tout en produisant des solutions de meilleure qualité possible avec un temps de calcul assez raisonnable, et on lui dénomme **Algorithme GAS^{MOKP}.**

PARTIE II
OPTIMISATION COMBINATOIRE
PROBLEME DE SAC A DOS MULTIDIMENSIONNEL MULTIOBJECTIF

Introduction

Nous définissons dans ce chapitre les problèmes d'optimisation combinatoire et les éléments de base relatifs à ces problèmes. Nous présentons par la suite le problème de sac à dos multidimensionnel multiobjectif, tout en proposant une approche de résolution hybride pour ce problème ; il s'agit d'une combinaison entre l'algorithme génétique et l'algorithme de colonie de fourmis, c'est l'Algorithme GAS^{MOKP}.

2.1/Définition

En entend par les problèmes d'optimisation **«combinatoires»** les problèmes dont leurs résolution se ramène à l'examen d'un nombre fini de combinaisons (contrairement aux problèmes d'optimisation continue dont la recherche s'effectue sur un nombre infini de combinaisons).

Un problème d'optimisation combinatoire est défini comme suit :
Etant donné un ensemble S de combinaisons, et une fonction $f : S \rightarrow I\!R$,
La solution du problème est la combinaison de S minimisant f, i.e., $s^* \in S$, :

$$f(s^*) \leq f(s_i), \forall s_i \in S.$$

f est appelée **la fonction de coût**, et aussi **la fonction économique** ou **la fonction objectif**,...

L'optimisation combinatoire trouve des applications dans différents domaines tels que la gestion, l'ingénierie, la conception, la production, les télécommunications, les transports, l'énergie, les sciences sociales et l'informatique.

Modélisation pour les problèmes d'optimisation combinatoire

Parmi les principaux outils de modélisation utilisés, nous trouvons la théorie des graphes (plus court chemin,...).
Les problèmes d'optimisation combinatoire peuvent aussi se formuler comme des programmes linéaires où les variables sont représentés par des valeurs entières.

Les problèmes de **type sac à dos** peuvent être représentés par un **programme linéaire** en nombres **entiers**, un ensemble de contraintes linéaires et une fonction objectif linéaire portant sur des variables à **valeurs entières**.

2.2/Le sac à dos multidimensionnel multiobjectif (MOKP)

Le problème du sac à dos multidimensionnel multiobjectif est l'un des problèmes les plus étudiés dans la communauté multiobjectif.

Présentation

Le problème du sac à dos multidimensionnel est représenté comme suit :

Soit un ensemble **d'items (ou objets)**, à chacun d'entre eux sont associées des valeurs de profit ainsi que des poids. Résoudre un problème du sac à dos multidimensionnel multiobjectif **(MOKP)** consiste à sélectionner le sous-ensemble d'objets maximisant simultanément un nombre **j de fonctions objectifs,** exprimées en fonction des valeurs de profit, tout en vérifiant un ensemble de contraintes **de type sac à dos**. Le problème du **MOKP** est défini formellement comme suit :

$$\text{MOKP01} \begin{cases} \max & z^j(x) = \sum_{i=1}^{n} c_i^j x_i & j = 1, ..., m \\ \text{t.q.} & \sum_{i=1}^{n} w_i^l x_i \leq b_l & l = 1, ..., q \\ & x_i \in \{0,1\} & i = 1, ..., n \end{cases}$$

Ou :
n est le nombre d'objets, x_i c'est la variable de décision, **m** c'est le nombre d'objectifs, $\mathbf{z^j}$ est la $\mathbf{j^{eme}}$ composante de la fonction multiobjectif **z** , et **q** le nombre de contraintes sac à dos du problème.

Le **MOKP** peut être utilisée pour modéliser de nombreux problèmes réels, comme la répartition de budgets et l'allocation de ressources.

Plusieurs algorithmes heuristiques ont été développés pour résoudre le **MOKP**. En cite, comme exemples :

Un algorithme hybride combinant une approche approchée (algorithme génétique) et une méthode exacte (la méthode de descente pour la recherche locale) « multiple objective genetic local search » (MOGLS) [Jaszkiewicz, 2002].

Un autre algorithme hybride combinant une approche approchée (algorithme génétique) et une méthode exacte (recherche tabou), c'est l'algorithme $\mathbf{GTS^{MOKP}}$ proposé par « Vincent BARICHARD 2003 **»**.

2.3/Un algorithme hybride pour le problème de sac à dos multiobjectif

Pour résoudre le **MOKP** on a conçu une approche hybride tout en combinant deux approches approchées ; un algorithme génétique et un algorithme de colonie de fourmis.

Le principe se base sur l'intégration d'une méthode de recherche de voisinage (l'algorithme de colonies de fourmis) au schéma génétique (recherche globale).

La diversification de la recherche assurée par la méthode de recherche globale (l'algorithme génétique), consiste à interdire la recherche de ce coincer dans des parties particulières de l'espace de recherche et assure une exploration du maximum de l'espace de recherche. Quant à **l'intensification**, elle est assurée par la recherche locale (l'algorithme de colonie de fourmis) celle-ci guide la recherche pour qu'elle se concentre autour des points spécifiques de l'espace de recherche (les dernières solutions trouvées), et par conséquent le choix de ces approches influe grandement sur la qualité des résultats. Dans le but de développer un algorithme hybride **GAS^{MOKP}** pour **le MOKP**, Nous avons Choisi un algorithme génétique pour assurer **la diversification**, et de lui greffer une méthode **d'intensification puissante** (L'algorithme de colonie de fourmi).

Algorithme génétique combiné à un algorithme de colonie de fourmi pour le MOKP (GAS^{MOKP})

Pour hybrider un algorithme génétique et une méthode de recherche locale (l'algorithme de colonie de fourmis), on a décidé de remplacer l'opérateur de mutation de l'algorithme génétique par un opérateur de recherche locale. A chaque génération, un nombre de croisements est effectué (l'opérateur de croisement est spécifique au problème). À Chaque nouvelle configuration (individu) **s** issue après croisement, on lui applique un opérateur de recherche locale **LS(s)** pour améliorer sa qualité. Et finalement, ce nouvel individu doit se passer par un mécanisme de sélection pour décider si ce dernier peut être introduit dans la population.

Dans cette section, nous présentons notre algorithme hybride **GAS^{MOKP}** pour **le MOKP**.

2.3.1/Algorithme général

GAS^{MOKP} suit le schéma classique utilisé pour l'hybridation entre algorithme génétique et une méthode de recherche locale. Nous donnons le pseudo-code de **GAS^{MOKP}**. Plus de détails sont fournis ultérieurement :

```
P ← Population initiale (P)
Tant que critère d'arrêt non rencontré faire :
λ ← Vecteur Aléatoire (utilisée pour l'évaluation,)
Evaluation :
Evaluer chaque individu S dans P selon f'(s; r; λ)
Sélection :
TP ← les N "meilleurs" individus de P
(p1; p2) ← Sélection des Parents(TP) qui vont participer à la reproduction.
Croisement :
S ← Croisement (p1; p2)
S ← Recherche par algorithme de colonie de fourmi (S ;N nombre d'itération fini)
P ← Ajouter l'individu s amélioré à la Population(s; P)
Fin Tant Que
Renvoyer P
```

Une explication plus détaillé de l'algorithme sera présentée dans les sections qui suivent.

2.3.2/Espace de recherche et valeurs d'évaluation

Une configuration (ou un individu) **s** est un vecteur binaire de **n** composantes, cette configuration doit satisfaire toutes les **q** contraintes du problème (c'est L'espace de recherche S).Pour évaluer une configuration $s \in S$, nous utilisons, une fonction d'agrégation linéaire pondérée:

$$f(s, r, \lambda) = \sum_{i-1}^{m} \lambda_i * (r_i - z_i(s))$$

Ou :
s est la configuration à évaluer,
$z_i(s)\ (i = 1 \cdots m)$ est la valeur de **s** pour la i^{eme} fonction objectif du problème,
λ est un vecteur de poids composé de **m** composantes comprises dans l'intervalle [0 … 1] dont la i^{eme} composante λ_i correspond au poids associé au i^{eme} objectif,
r est un vecteur de référence à **m**-composantes.

Ainsi, pour chaque configuration **s** et pour un vecteur de référence **r** donné, une valeur d'adaptation **« fitness »** est attribuée à **s** selon l'évaluation et le poids associé à chaque objectif de **s**. Cette valeur d'adaptation définit un ordre total pour les configurations de l'espace de recherche **S**.

2.3.3/La recherche globale par algorithme génétique

Le codage des individus dans **GAS^{MOKP}** consiste à les représenter par des chaines binaires de **0/1** de longueur **n**, ou chaque bit correspond à la présence ou l'absence d'un objet du sac à dos. Ainsi, changer un bit de **0 à 1** correspond à ajouter un objet au **sac à dos**, et changer le bit de **1 à 0** correspond à retirer un objet au sac à dos. L'objet sera représenté par sa valeur.

Au début de la recherche génétique, on constitue une population initiale, l'algorithme **GAS^{MOKP}** construit aléatoirement autant d'individus qu'il y a de places dans la population. Pour construire chaque individu aléatoirement, une procédure prend au hasard un sous-ensemble d'objets. Si la somme des poids de tous les objets viole une des contraintes de sac à dos, alors les objets influant le moins sur le profit sont retirés de l'individu, et ceci tant qu'il reste des contraintes violées **(la correction)**. Puis la recherche génétique va s'itérer tant qu' **un critère d'arrêt** spécifié au départ n'est pas rencontré.

Lors d'une itération, la première action effectuée par l'algorithme est de générer un vecteur λ de dimension **m** , afin de calculer la valeur d'adaptation de différents individus de la population **P**.

L'opérateur **de croisement** utilisé dans la partie génétique de **GAS^{MOKP}** opère sur un sous-ensemble de la population **P** (population temporaire), ce sous-ensemble est sélectionné en utilisant une **roue de fortune.** Puis les individus (de cette population temporaire) sont sélectionnés deux à deux d'une manière aléatoire pour être recombinés grâce à un opérateur de croisement **monopoint**.

L'individu résultant d'un croisement peut violer des contraintes. Nous restaurons donc la **faisabilité** de cet individu de la même manière que lors de la construction de la population initiale **(la correction)**.

Après la correction, on procède à la **mutation** par une recherche locale assuré par **l'algorithme de colonie de fourmis** (voir prochaine section), dans le but d'améliorer l'individu et de lui remplacer par un autre individu **s***, beaucoup mieux, issu de son voisinage.

Enfin, il faut décider si l'individu **s*** il fera partie de la prochaine génération ou non. Pour cela, nous le comparons avec le plus mauvais individu de la population temporaire. Si s* est mieux que le mauvais individu de la population temporaire, alors **s*** sera ajouté à la population courante et remplacera le mauvais individu. Dans les autres cas, **s*** sera rejeté.

2.3.4/Recherche locale par algorithme de colonie de fourmis

La recherche locale par l'algorithme de colonie de fourmis à pour but d'améliorer une configuration **s** produite par l'opérateur de croisement. Cette recherche est effectuée avant d'insérer l'individu **s** dans la population.

Le principe de l'algorithme est basé sur le dépôt de phéromone, il existe différentes stratégies de dépôt et d'exploitation des traces de phéromone. Avant d'utiliser la métaheuristique **ACO**, il faut d'abord décider sur quels composants de solutions les fourmis déposent de la phéromone.

Algorithme ACO pour le MKP

Dans les algorithmes **ACO**, les fourmis déposent les traces de phéromone sur les composants des meilleures solutions construites pour attirer les autres fourmis vers ces solutions. Ainsi pour résoudre un problème avec la métaheuristique **ACO**, il faut décider d'abord sur quels composants des solutions les fourmis déposent de la phéromone et comment exploiter ces phéromones lors de la construction de nouvelles solutions. Une solution d'un **MKP** est un ensemble d'objets sélectionnés **S** = {$\mathbf{o_1}$, **. . . ,** $\mathbf{o_k}$},(on considèrera qu'un objet $\mathbf{o_i}$ est sélectionné si la variable de décision correspondante $\mathbf{x_{oi}}$ a été mise à 1).

Etant donnée une telle solution **S**, nous avons choisi le dépôt des traces de phéromone sur chaque objet sélectionné dans **S** (les sommets). Dans ce cas, lors de la construction des solutions suivantes, la probabilité de sélectionner chaque objet de **S** sera augmentée, c'est l'algorithme **Vertex-AK**.

L'algorithme Ant-Knapsack

L'algorithme **Ant-Knapsack** est décrit comme suit : A chaque cycle de cet algorithme, chaque fourmi construit sa propre solution. Ensuite, lorsque toutes les fourmis ont construit leurs solutions, les traces de phéromone sont mises à jour. L'algorithme s'arrête lorsqu'une fourmi a trouvé une solution optimale, ou lorsqu'un nombre maximal de cycles a été atteint.

les traces de phéromone sont limitées à l'intervalle $[\tau_{\min}, \tau_{\max}]$ et sont initialisées à $\tau_{\max}$.

Pour construire une solution, les fourmis choisissent aléatoirement un objet initial, puis ajoutent itérativement des objets qui sont choisis à partir d'un ensemble

de candidats qui contient tous les objets qui peuvent être sélectionnés sans violer les contraintes de ressources.

Pour la construction d'une solution **S_k** , à chaque étape la fourmi **k** choisit l'objet **o_i** à ajouter à la solution parmi un ensemble de sommets (candidats) selon une probabilité **$p_{Sk}(o_i)$**,telle que :

$$p_{S_k}(o_i) = \frac{[\tau_{S_k}(o_i)]^{\alpha}.[\eta_{S_k}(o_i)]^{\beta}}{\sum_{o_j \in Candidats}[\tau_{S_k}(o_j)]^{\alpha}.[\eta_{S_k}(o_j)]^{\beta}}$$

$$\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{o_i\}$$

Cette probabilité est proportionnel à un **facteur phéromonal** $\tau_{S_k}(o_i)$ et **un facteur heuristique** $\eta_{S_k}(o_i)$, ces deux facteurs étant pondérés par deux paramètres α et β qui déterminent leur importance relative.

- **Le facteur heuristique** pour un objet **j** est défini comme suit: $\eta_{S_k}(j) = \frac{p_j}{h_{S_k}(j)}$

Avec : **pj** est le profit de l'objet **j**

et $h_{S_k}(j) = \sum_{i=1}^{m} \frac{r_{ij}}{d_{S_k}(i)}$

$h_{S_k}(j)$ Représente la dureté de l'objet **j** par rapport à toutes les contraintes **i** ∈ **1…m** et relativement à la solution construite **Sk**, de sorte que plus ce ratio est faible plus l'objet est intéressant.

Avec : $d_{S_k}(i) = b_i - \sum_{g \subset S_k} r_{ig}$ est la quantité restante de la ressource **i** lorsque la fourmi a construit la solution **S_k** ;

L'algorithme ACO pour le MKP

Initialiser les traces de phéromone à τ_{max}.
Répéter
  Pour chaque fourmi **k** dans **1..nbAnts,** construire une solution S_k comme suit :
    Choisir aléatoirement un premier objet **o1** ∈ **1..n**
    $S_k \leftarrow \{o1\}$
    Candidats {$o_i \in 1..n$/ o_i peut être sélectionné sans violer des contraintes de ressources}
    Tant que Candidats $\neq \emptyset$ **faire**
      Choisir un objet $o_i \in$ Candidats avec la probabilité :

$$p_{S_k}(o_i) = \frac{[\tau_{S_k}(o_i)]^{\alpha}.[\eta_{S_k}(o_i)]^{\beta}}{\sum_{o_j \in Candidats}[\tau_{S_k}(o_j)]^{\alpha}.[\eta_{S_k}(o_j)]^{\beta}}$$

$$\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{o_i\}$$

      Enlever de Candidats chaque objet qui viole des contraintes de ressources
    Fin tant que
  Fin pour
  mettre à jour les traces de phéromone en fonction de {$S_1, \ldots, S_{nbAnts}$}
  Si une trace de phéromone est inférieure à τ_{min} alors la mettre à τ_{min}
  Si une trace de phéromone est supérieure à τ_{max}. alors la mettre à τ_{max}.

Jusqu'à nombre maximal de cycles atteint ou solution optimale trouvée

Définition des composants phéromonaux

Les fourmis de l'algorithme Ant-Knapsack (**Vertex-AK**) déposent de la phéromone sur l'ensemble des objets (sur chaque objet sélectionné dans **S**)
-La quantité de phéromone déposée sur un objet $\mathbf{o_i}$ est notée $\tau(o_i)$. Cette quantité représente la **« désirabilité »** de choisir l'objet $\mathbf{o_i}$ lors de la construction d'une solution.
- Le facteur phéromonal $\tau_{S_k}(o_i)$ traduit l'expérience passée de la colonie et est utilisé dans le calcul de la probabilité de choisir un objet par les fourmis, il dépend de la quantité déposée sur l'objet candidat : $\tau_{S_k}(o_i) = \tau(o_i)$

Mise à jour de la phéromone :

Après que toutes les fourmis aient fini la construction de leurs solutions, les traces de phéromone sont mises à jour. Cette mise-à-jour s'effectue en deux étapes :

-Dans une première étape, toutes les traces de phéromone sont diminuées, pour simuler l'évaporation, en multipliant chaque composant phéromonal par **un ratio de persistance** $(1-\rho)$ tel que $0 \leq \rho \leq 1$**.**

-Dans un deuxième temps, la meilleure fourmi du cycle dépose de la phéromone. plus précisément, soit $\mathbf{S_k} \in \mathbf{\{S_1, \ldots, S_{nbAnts}\}}$ la meilleure solution construite durant le cycle courant (celle ayant un profit maximal) et $\mathbf{S_{best}}$ est la meilleure solution construite depuis le début de l'exécution (y compris le cycle courant).

La quantité de phéromone déposée par la fourmi **k** est inversement proportionnelle à la différence de profit entre $\mathbf{S_k}$ et $\mathbf{S_{best}}$, i.e., elle est égale à $\mathbf{1/(1 + profit(S_{best}) - profit(S_k))}$. Cette quantité de phéromone est déposée sur les composants phéromonaux de $\mathbf{S_k}$,

Influence des paramètres α et ρ sur la résolution :

α et ρ sont des paramètres qui ajustent la diversification, et l'intensification de la recherche

en particulier, la diversification peut être augmentée soit en diminuant la valeur du poids du **facteur phéromonal** α (de sorte que les fourmis deviennent moins sensibles aux traces phéromonales), soit en diminuant le **taux d'évaporation** ρ (de sorte que la phéromone s'évapore plus doucement et les écarts d'une trace à l'autre évoluent plus doucement). Lorsque l'on augmente ainsi la capacité exploratoire des fourmis, on trouve généralement de meilleures solutions.

Conclusion

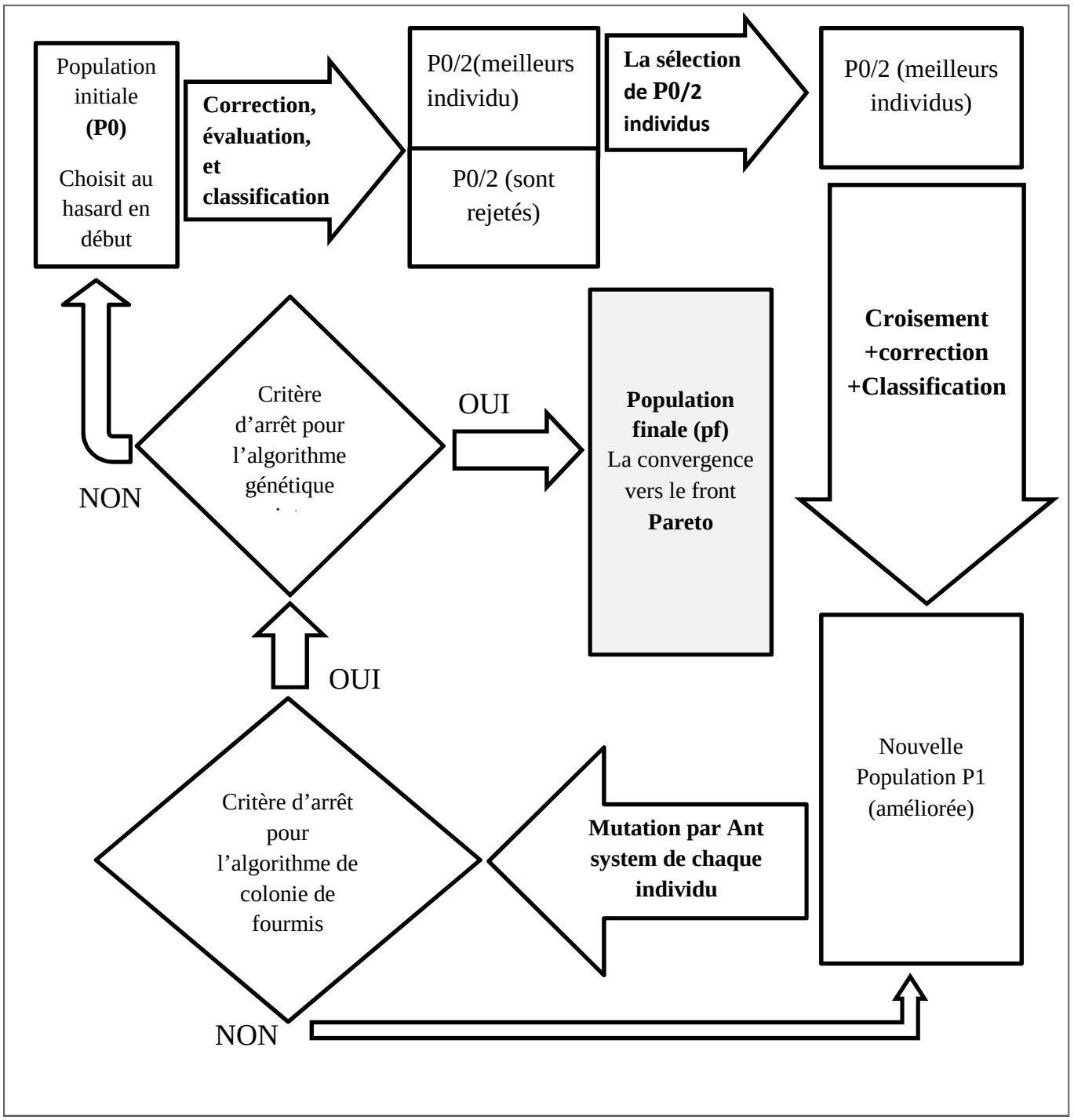
Dans cette partie on a présenté l'algorithme **GAS** $^{\mathbf{MOKP}}$ pour le problème de sac à dos multidimensionnel multiobjectif, nous montrons les résultats obtenus l'ors de l'exploitation de cet algorithme dans la prochaine partie.

PARTIE III : RESULTATS

Introduction

Dans cette partie nous présentons L'algorithme **GAS** ^{MOKP} d'une manière plus explicite, nous donnons l'organigramme global de l'algorithme et nous exposons les résultats obtenus lors de lancement de l'algorithme pour 01, 05, 10, et 30 itérations et nous commentons ces résultats.

3.1/Organigramme globale



3.2/Explication de l'algorithme et résultats

On a pris un exemple d'une **population initiale** constituée de 20 individus.
Chaque individus représente une solution du problème ; c'est une combinaison des 0 et des 1 indiquant la présence ou l'absence d'un objet formant ainsi une solution (16 objets au maximum comme exemple).

0=**absence** d'un objet pour une solution donnée.
1=**présence** d'un objet pour une solution donnée.

On fait correspondre à chaque objet du sac à dos **une valeur de profit** et **un poids.**

La nomination **multi objectif** vient de faite que le problème exige plusieurs objectifs à satisfaire simultanément, dans notre cas on a fixé le nombre de fonctions objectifs à **6** (à titre d'exemple)

$\mathbf{f_1}= a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 + a_5 x_5 + a_6 x_6 + a_7 x_7 + a_8 x_8 + a_9 x_9 + a_{10} x_{10} + a_{11} x_{11} + a_{12} x_{12} + a_{13} x_{13} + a_{14} x_{14} + a_{15} x_{15} + a_{16} x_{16}$

$\mathbf{f_2}=b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + b_{11} x_{11} + b_{12} x_{12} + b_{13} x_{13} + b_{14} x_{14} + b_{15} x_{15} + b_{16} x_{16}$

$\mathbf{f_3}=c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4 + c_5 x_5 + c_6 x_6 + c_7 x_7 + c_8 x_8 + c_9 x_9 + c_{10} x_{10} + c_{11} x_{11} + c_{12} x_{12} + c_{13} x_{13} + c_{14} x_{14} + c_{15} x_{15} + c_{16} x_{16}$

$\mathbf{F_4}=d_1 x_1 + d_2 x_2 + d_3 x_3 + d_4 x_4 + d_5 x_5 + d_6 x_6 + d_7 x_7 + d_8 x_8 + d_9 x_9 + d_{10} x_{10} + d_{11} x_{11} + d_{12} x_{12} + d_{13} x_{13} + d_{14} x_{14} + d_{15} x_{15} + d_{16} x_{16}$

$\mathbf{F_5}= e_1 x_1 + e_2 x_2 + e_3 x_3 + e_4 x_4 + e_5 x_5 + e_6 x_6 + e_7 x_7 + e_8 x_8 + e_9 x_9 + e_{10} x_{10} + e_{11} x_{11} + e_{12} x_{12} + e_{13} x_{13} + e_{14} x_{14} + e_{15} x_{15} + e_{16} x_{16}$

$\mathbf{F_6} = g_1 x_1 + g_2 x_2 + g_3 x_3 + g_4 x_4 + g_5 x_5 + g_6 x_6 + g_7 x_7 + g_8 x_8 + g_9 x_9 + g_{10} x_{10} + g_{11} x_{11} + g_{12} x_{12} + g_{13} x_{13} + g_{14} x_{14} + g_{15} x_{15} + g_{16} x_{16}$

Tel que $\mathbf{x_1, x_2,\ldots\ldots x_{16}}$ ce sont les paramètres du problème (les variables du problème) discontinus c'est pour cela vient la nomination **optimisation combinatoire**. Dans les calculs ils seront remplacés soit par des **0** ou par des **1** (absence ou présence des objets).

Pour chaque objectif, les variables sont pondérés aux valeurs de **profits :**

$a_1 \ldots\ldots a_{16}$ pour la première fonction objective $\mathbf{F_1}$.$b_1 \ldots\ldots b_{16}$ pour la deuxième fonction objective $\mathbf{F_2}$ $c_1 \ldots\ldots c_{16}$ pour la troisième fonction objective $\mathbf{F_3}$.$d_1 \ldots\ldots d_{16}$ pour la quatrième fonction objective $\mathbf{F_4}$ $e_1 \ldots\ldots e_{16}$ pour la cinquième fonction objective $\mathbf{F_5}$.$g_1 \ldots\ldots g_{16}$ pour la sixième fonction objective $\mathbf{F_6}$.

(Chaque objet de sac à dos à sa propre valeur de profit pour chaque fonction objectif).

Le problème exige aussi un nombre **q** de contraintes à satisfaire**,** dans notre cas on a placé **6** fonctions contraintes, chaque fonction contrainte représente le poids maximal à ne pas dépasser :

$h_1 x_1 + h_2 x_2 + h_3 x_3 + h_4 x_4 + h_5 x_5 + h_6 x_6 + h_7 x_7 + h_8 x_8 + h_9 x_9 + h_{10} x_{10} + h_{11} x_{11} + h_{12} x_{12} + h_{13} x_{13} + h_{14} x_{14} + h_{15} x_{15} + h_{16} x_{16} <= p_{max1}$

$j_1 x_1 + j_2 x_2 + j_3 x_3 + j_4 x_4 + j_5 x_5 + j_6 x_6 + j_7 x_7 + j_8 x_8 + j_9 x_9 + j_{10} x_{10} + j_{11} x_{11} + j_{12} x_{12} + j_{13} x_{13} + j_{14} x_{14} + j_{15} x_{15} + j_{16} x_{16} <= p_{max2}$

$k_1 x_1 + k_2 x_2 + k_3 x_3 + k_4 x_4 + k_5 x_5 + k_6 x_6 + k_7 x_7 + k_8 x_8 + k_9 x_9 + k_{10} x_{10} + k_{11} x_{11} + k_{12} x_{12} + k_{13} x_{13} + k_{14} x_{14} + k_{15} x_{15} + k_{16} x_{16} <= p_{max3}$

$l_1 x_1 + l_2 x_2 + l_3 x_3 + l_4 x_4 + l_5 x_5 + l_6 x_6 + l_7 x_7 + l_8 x_8 + l_9 x_9 + l_{10} x_{10} + l_{11} x_{11} + l_{12} x_{12} + l_{13} x_{13} + l_{14} x_{14} + l_{15} x_{15} + l_{16} x_{16} <= p_{max4}$

$m_1 x_1 + m_2 x_2 + m_3 x_3 + m_4 x_4 + m_5 x_5 + m_6 x_6 + m_7 x_7 + m_8 x_8 + m_9 x_9 + m_{10} x_{10} + m_{11} x_{11} + m_{12} x_{12} + m_{13} x_{13} + m_{14} x_{14} + m_{15} x_{15} + m_{16} x_{16} <= p_{max5}$

$z_1 x_1 + z_2 x_2 + z_3 x_3 + z_4 x_4 + z_5 x_5 + z_6 x_6 + z_7 x_7 + z_8 x_8 + z_9 x_9 + z_{10} x_{10} + z_{11} x_{11} + z_{12} x_{12} + z_{13} x_{13} + z_{14} x_{14} + z_{15} x_{15} + z_{16} x_{16} <= p_{max6}$

Pour une solution donnée, chaque objet à son propre poids qui correspond aux différentes fonctions contraintes :
$h_1 \ldots\ldots h_{16}$ pour la première contrainte. $j_1 \ldots\ldots j_{16}$ pour la deuxième contrainte. $k_1 \ldots\ldots k_{16}$ pour la troisième contrainte. $I_1 \ldots\ldots I_{16}$ pour la quatrième contrainte. $m_1 \ldots\ldots m_{16}$ pour la cinquième contrainte. $z_1 \ldots\ldots z_{16}$ pour la sixième contrainte.

les contraintes sont les poids à ne pas dépasser: p_{max1} , p_{max2} , p_{max3}, p_{max4} , p_{max5} , p_{max6}
Leurs valeurs sont exigées, dans notre exemple on prendra les valeurs suivantes :
$p_{max1}= 54, p_{max2}=64, p_{max3}=55, p_{max4}=46, p_{max5}=57, p_{max6}=50$

3.2.1/Recherche globale ; algorithme génétique pour une itération

3.2.1.1/ Population initiale :

Calcul de la matrice des profits :

Avant de calculer les valeurs des profits il faut d'abord générer une population initiale au hasard constituée de 20 individus :

1	1	0	1	1	]	0	1	1	1	0	1	1	]	1	0
0	1	1	1	1	]	1	1	1	1	1	0	1	]	1	0
0	0	0	1	1	]	1	0	0	0	1	1	0	]	0	1
1	1	0	1	1	]	0	0	1	0	1	0	0	]	1	0
1	1	1	0	0	]	1	0	1	0	1	0	0	]	1	0
0	1	1	1	1	]	1	1	0	1	0	0	1	1	1	0
1	0	0	0	0	1	0	1	0	1	0	1	1	1	0	0
0	1	0	1	0	]	0	0	1	1	1	0	1	1	0	1
0	1	0	1	0	]	0	0	1	1	1	0	1	1	0	1
0	0	0	0	0	]	0	0	1	1	0	0	0	1	0	0
1	0	0	0	0	]	1	1	1	0	0	0	1	]	0	0
1	1	1	0	1	]	1	0	1	0	0	1	1	]	1	1
0	0	0	0	1	1	1	1	1	0	1	1	0	1	1	1
1	1	0	0	0	]	1	0	0	0	0	0	0	1	0	0
1	1	0	0	0	]	1	0	1	1	0	0	0	]	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	]	0	1
1	0	0	1	0	1	1	1	0	0	0	1	0	1	0	1
0	1	1	1	1	]	1	0	0	1	1	0	1	1	1	1
1	1	0	0	1	]	0	0	1	1	1	0	0	1	0	0
1	0	1	1	1	]	1	1	1	1	1	0	0	1	0	0

Pour chaque individu on calcul sa valeur de profit pour les **06 objectifs** donnés :

59	48	31	33	37	39
66	53	34	41	44	33
36	29	25	20	33	24
41	32	27	34	31	34
46	29	22	29	32	29
50	42	26	32	34	25
42	36	27	20	29	32
53	48	31	37	35	29
53	48	31	37	35	29
21	16	11	10	13	10
34	20	15	15	23	23
57	38	27	29	39	35
58	46	40	33	50	36
24	15	12	18	20	19
46	32	12	19	22	20
48	43	33	27	35	30
45	34	27	20	35	32
59	51	34	40	42	32
49	40	30	36	34	36
63	46	36	35	44	42

Ensuite on calcul les valeurs d'évaluation de chaque individu de la population initiale, c'est les valeurs **d'adaptation « fitness »** de chaque individu (configuration) selon la fonction d'évaluation suivante :

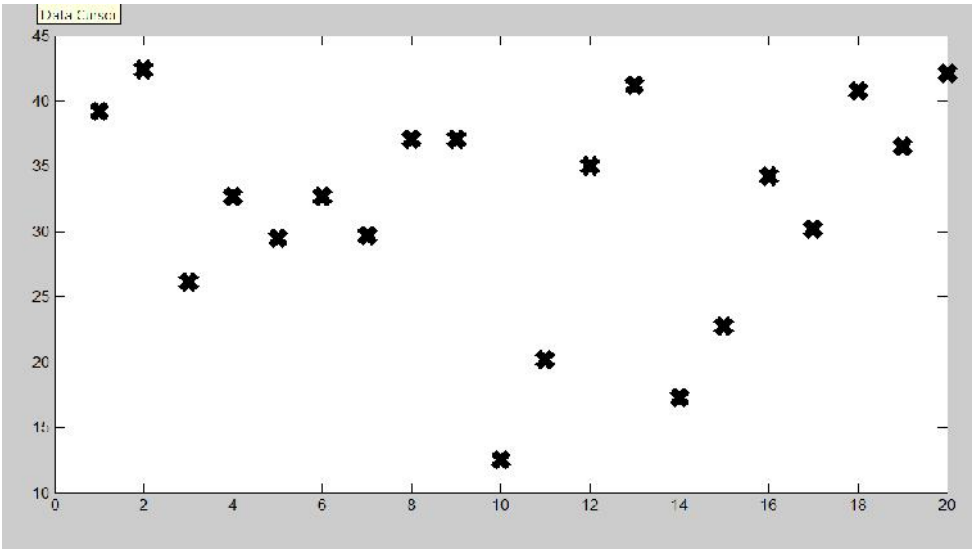
$F = \sum \lambda_i F_i = \lambda_1 F1 + \lambda_2 F2 + \lambda_3 F3 + \lambda_4 F4 + \lambda_5 F5 + \lambda_6 F6$ tel que $\sum \lambda = 1$, λ est le vecteur de poids λ , il prend des valeurs choisit aléatoirement dans l'intervalle de **0 à 1** puisqu'on connait pas au **préalablement** l'importance des objectifs. cette fonction d'évaluation sert a évaluer tous les individus de la génération pour les permetrre participer à la phase de croisement en vue de formation des prochaines générations.

Le vecteur λ choisit aléatoirement est

0.1054 0.1501 0.2186 0.2198 0.0853 0.2208

Voici le vecteur qui correspond aux valeurs d'évaluations de chaque individus de la population initiale:

39.2207
42.3955
26.1224
32.6515
29.5175
32.7116
29.6679
37.0889
37.0009
12.5345
20.2019
35.0427
41.2290
17.2619
22.7436
34.2714
30.1957
40.7463
36.4884
42.1342



Calcul de la matrice de poids

Pour chaque individu on calcul son poids à l'égard des **06 contraintes** (fonctions poids de sac à dos sus exigés) :

63	58	62	47	45	42
61	49	77	47	47	45
31	33	36	30	34	31
44	32	45	24	35	28
45	31	50	29	25	26
51	50	62	44	41	37
41	43	30	41	22	36
50	48	63	38	30	28
50	48	63	38	30	28
19	18	20	24	10	18
28	19	29	24	19	21
59	54	67	46	36	44
48	49	53	48	33	61
26	26	24	21	14	13
38	34	50	37	24	25
40	38	46	32	24	42
38	44	34	40	33	40
58	56	75	49	45	40
48	39	48	38	28	32
54	41	56	51	48	44

On remarque que ces valeurs dépassent les valeurs de poids maximum exigées dans le contexte du problème, c'est pour cela on a introduit une **séquence de correction** pour les individus qui ne respectent pas les contraintes de départ.

3.2.1.2/Correction

Le principe est de commencer à enlever du sac à dos les objets qui ont de faibles rapports **profits/poids** et à chaque fois qu'on enlève un objet de sac à dos, en recalculera le poids global de l'individu et sa valeur d'évaluation , la séquence s'itère tanqu 'il reste des viols sur des contraintes de poids . à la fin de ce processus on aura une générations d'individus correctes ,valable à participer comme une base pour **l'approche évolutionnaire d'optimisation** pour retrouver des solutions (individus) beaucoup plus meilleur . Pour notre exemple on obtiendra une nouvelle **génération d'individus corrects** :

1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0
0	1	1	1	1	0	0	1	1	0	1	0	1	0	1	0
0	0	1	1	1	0	1	0	0	0	1	1	0	0	0	1
1	1	1	1	1	0	0	0	1	0	1	0	0	0	1	0
1	1	1	1	0	0	1	0	1	0	1	0	0	0	1	0
0	1	1	1	1	0	0	1	0	1	0	0	1	1	0	0
1	0	1	1	0	1	0	1	0	1	0	1	1	1	0	0
0	1	1	1	0	0	0	0	1	0	1	0	1	1	0	1
0	1	1	1	0	0	0	0	1	0	1	0	1	1	0	1
0	0	1	1	0	0	0	0	1	1	0	0	0	1	0	0
1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0
1	1	1	1	1	0	0	0	1	0	0	1	1	0	1	1
0	0	1	1	1	1	0	1	1	0	1	1	0	1	0	0
1	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0
1	1	1	1	0	0	1	0	1	1	0	0	0	0	0	1
0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	1
1	0	1	1	0	1	1	1	0	0	0	1	0	1	0	1
0	1	1	1	1	0	1	0	0	0	1	0	1	1	0	1
1	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0
1	0	1	1	1	0	0	1	1	1	1	0	0	1	0	0

Ainsi on calcul Pour chaque individu de la nouvelle génération trouvée sa valeur de profit pour les **06 objectifs** donnés et en suite en calcule leurs valeurs d'évaluation :

49	38	29	31	35	37
46	41	31	38	35	30
36	29	25	20	33	24
41	32	27	34	31	34
46	29	22	29	32	29
43	40	24	30	26	23
42	36	27	20	29	32
44	39	30	36	34	28
44	39	30	36	34	28
21	16	11	10	13	10
34	20	15	15	23	23
46	35	25	27	31	33
44	37	37	30	41	33
24	15	12	18	20	19
46	32	12	19	22	20
48	43	33	27	35	30
45	34	27	20	35	32
44	39	31	37	39	29
49	40	30	36	34	36
57	45	35	34	37	41

35.1767
35.7410
26.1224
32.6515
29.5175
29.6728
29.6679
34.0449
34.0449
12.5345
20.2019
31.4322
35.6572
17.2619
22.7436
34.2714
30.1957
35.1306
36.4884
40.0954

Il est clair que les nouveaux individus ci-trouvés respectent tous les contraintes de poids de sac à dos sus-exigés :

52	47	49	33	36	34
46	37	55	24	33	31
31	33	36	30	34	31
44	32	45	24	35	28
45	31	50	29	25	26
44	43	52	34	32	27
41	43	30	41	22	36
43	41	54	26	24	23
43	41	54	26	24	23
19	18	20	24	10	18
28	19	29	24	19	21
51	49	54	35	28	35
38	36	33	34	22	49
26	26	24	21	14	13
38	34	50	37	24	25
40	38	46	32	24	42
38	44	34	40	33	40
42	43	55	32	34	30
48	39	48	38	28	32
51	38	50	43	42	37

3.2.1.3/Ordonnancement

Après la phase de correction on procède à **l'ordonnancement (classement)** des individus de la génération selon leurs valeurs de profits (**évaluation** de chaque individu), et pour cela on introduit une séquence qui joue ce rôle, on aura la génération suivante :

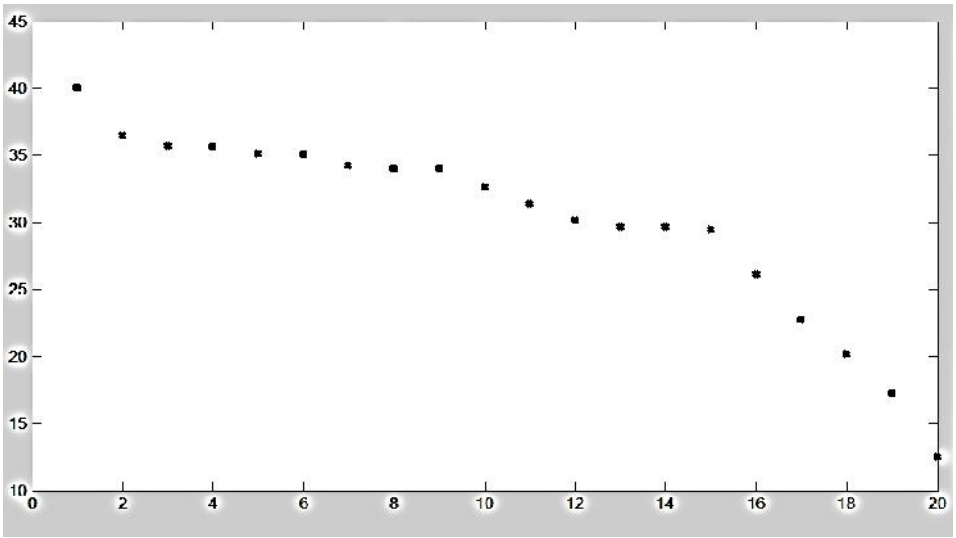
1	0	1	1	1	0	0	1	1	1	1	1	0	1	0	0
1	1	0	0	1	0	0	0	1	1	1	1	0	1	0	0
0	1	0	1	1	0	0	1	1	0	1	1	1	0	1	0
0	0	0	0	1	1	0	1	1	0	1	1	0	1	0	0
1	1	0	1	1	0	0	1	1	0	0	1	1	0	0	0
0	1	0	1	1	0	1	0	0	0	1	1	1	1	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1
0	1	0	1	0	0	0	0	1	0	1	1	1	1	0	1
0	1	0	1	0	0	0	0	1	0	1	1	1	1	0	1
1	1	0	1	1	0	0	0	1	0	1	1	0	0	1	0
1	1	0	0	1	0	0	0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	1	0	0	0	1	0	1	0	1
0	1	1	1	1	0	0	1	0	1	0	1	1	1	0	0
1	0	0	0	0	1	0	1	0	1	0	1	1	1	0	0
1	1	1	0	0	0	1	0	1	0	1	1	0	0	1	0
0	0	0	1	1	0	1	0	0	0	1	1	0	0	0	1
1	1	0	0	0	0	1	0	1	1	0	1	0	0	0	1
1	0	0	0	0	0	1	1	1	0	0	1	1	0	0	0
1	1	0	0	0	0	1	0	0	0	0	1	0	1	0	0
0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0

La matrice de profits et les valeurs d'évaluation après l'ordonnancement seront donc :

57	45	35	34	37	41
49	40	30	36	34	36
46	41	31	38	35	30
44	37	37	30	41	33
49	38	29	31	35	37
44	39	31	37	39	29
48	43	33	27	35	30
44	39	30	36	34	28
44	39	30	36	34	28
41	32	27	34	31	34
46	35	25	27	31	33
45	34	27	20	35	32
43	40	24	30	26	23
42	36	27	20	29	32
46	29	22	29	32	29
36	29	25	20	33	24
46	32	12	19	22	20
34	20	15	15	23	23
24	15	12	18	20	19
21	16	11	10	13	10

40.0954
36.4884
35.7410
35.6572
35.1767
35.1306
34.2714
34.0449
34.0449
32.6515
31.4322
30.1957
29.6728
29.6679
29.5175
26.1224
22.7436
20.2019
17.2619
12.5345

L'image ci-après représente les valeurs d'évaluation des individus après leurs mises en ordre :



La matrice de poids de tous les individus de la nouvelle génération après leurs mises en ordre à l'égard des 6 contraintes sus-exigés :

51	38	50	43	42	37
48	39	48	38	28	32
46	37	55	24	33	31
38	36	33	34	22	49
52	47	49	33	36	34
42	43	55	32	34	30
40	38	46	32	24	42
43	41	54	26	24	23
43	41	54	26	24	23
44	32	45	24	35	28
51	49	54	35	28	35
38	44	34	40	33	40
44	43	52	34	32	27
41	43	30	41	22	36
45	31	50	29	25	26
31	33	36	30	34	31
38	34	50	37	24	25
28	19	29	24	19	21
26	26	24	21	14	13
19	18	20	24	10	18

Ensuite, on calcule l'individu **de référence** c'est une approximation du **meilleur individu** dans toutes les générations ; et pour cela on introduit une correction sur le meilleur individu en négligeant tous les critères de poids :

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Après sa correction, on aura une estimation du meilleur individu qui respecte les critères de poids :

1 1 0 0 1 1 0 1 1 0 1 0 1 1 0 0

Cet individu sert comme référence pour juger les autres individus de la population. Sa valeur d'évaluation sera donc :

45.0605

3.2.1.4/La sélection

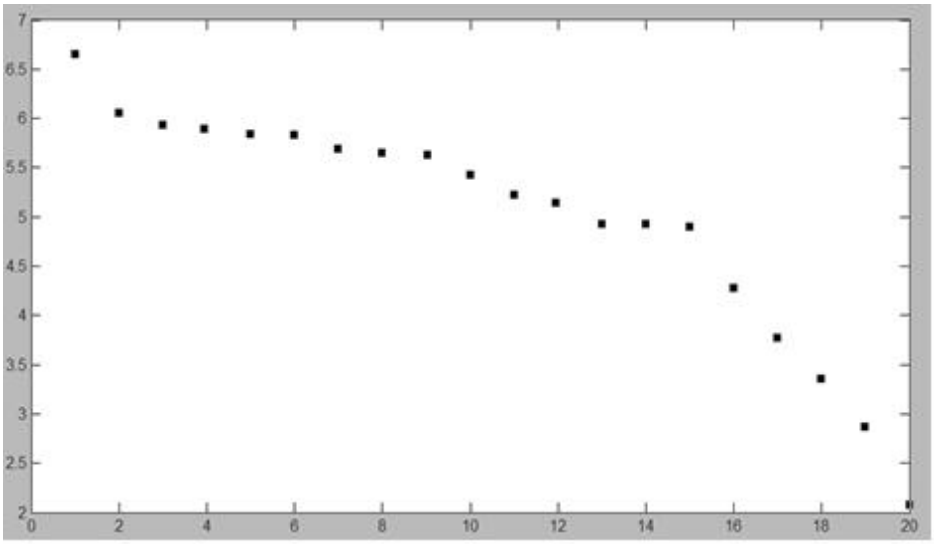
On procède à la phase de sélection de 10 (meilleurs) individus, et pour cela on utilisera le principe de la **roue de fortune** ; ce principe repose sur la création d'une population intermédiaire constituée de 10 individus (moitié de la population initiale), en utilisant une roue de fortune pour les choisir au hasard.

D'abord on calcule le **taux de fitness** de chaque individu de la génération **Y1**……jusqu'à **Y20** selon :

$$Y_i = F_i \, 100 / \sum F_i \qquad i = 1 \ldots\ldots\ldots 20$$

Voici les taux de fitness des individus trouvés ainsi qu'une image qui les représente :

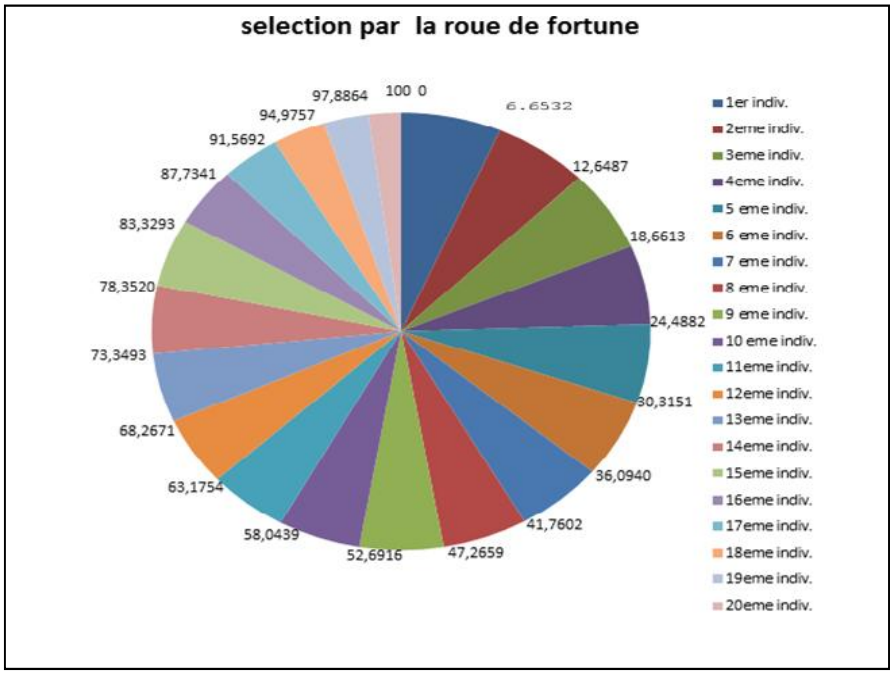
6.6532
6.0546
5.9306
5.9167
5.8370
5.8293
5.6868
5.6492
5.6492
5.4180
5.2156
5.0105
4.9237
4.9229
4.8979
4.3346
3.7739
3.3522
2.8643
2.0799



Le principe de la roue de fortune, est de représenter les taux de fitness trouvés sur une roue de fortune ,celle-ci est subdivisée en 20 partie ,chaque parties correspond à un individu, la surface de chaque partie est proportionnel à la valeur d'évaluation de l'individu qui le représente (chaque individu occupe un pourcentage de la roue comme montre la figure ci-après) :

Voici les intervalles qui délimitent les taux d'évaluation de chaque individu dans la population :

0 [le1^{er} individu]**6.6532 [**le 2ème individu]**12.6487 [**le 3ème individu**] 18.6613 [**le 4ème individu**] 24.4882 [**le 5èmeindividu**] 30.3151 [**le 6ème individu]**36.0940 [**le7^{ème}individu**] 41.7602 [**le 8ème individu**] 47.2659 [**le 9ème individu**] 52.6916 [**le 10ème individu**] 58.0439 [**le 11ème individu**] 63.1754 [**le 12ème individu**] 68.2671 [**le 13ème individu**] 73.3493 [**le 14ème individu**] 78.3520 [**le 15ème individu**] 83.3293 [**le 16ème individu**] 87.7341 [**le 17ème individu**] 91.5692 [**le 18ème individu**] 94.9757 [**le 19ème individu**] 97.8864 [**le 20 ^{ème} individu**] 100.0000**



Ensuite on fait tourner cette roue 10 fois (un nombre équivalent à la moitié du nombre d'individus de la génération) et à chaque fois on prendra la valeur sur laquelle la roue s'est arrêté et donc l'individu correspondant sera évidemment choisi :

Après 10 tirages on aura par exemples les résultats suivants :

92 43 18 90 98 44 11 26 41 59

à chaque valeur retrouvée on va prendre l'individu qui lui correspond.

Après la sélection de 10 individus par le principe de la roue de fortune on aura la population intermédiaire suivante (l'ordre n'est pas nécessaire ici) :

1	1	0	0	1	0	0	0	1	0	1	1	1	0	0	1
1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	1
1	0	0	1	1	0	0	1	1	1	1	0	0	1	0	0
0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0
0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1
0	1	0	1	0	0	0	0	1	1	1	0	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0
0	0	0	0	1	1	0	1	1	0	1	1	0	1	0	0
1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0

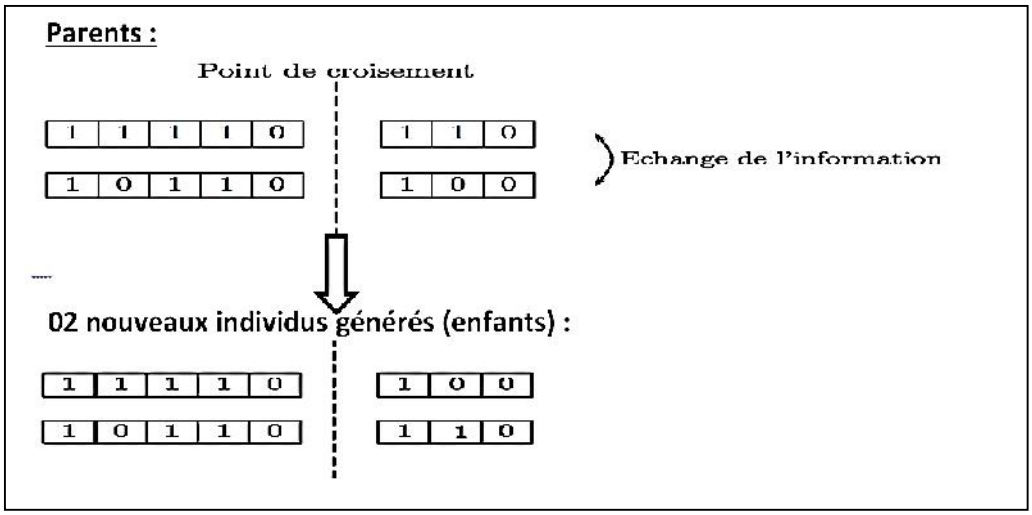
Ce sont les individus qui vont participer à la phase de croisement pour générer 10 nouveaux individus.

3.2.1.5/Croisement (reproduction)

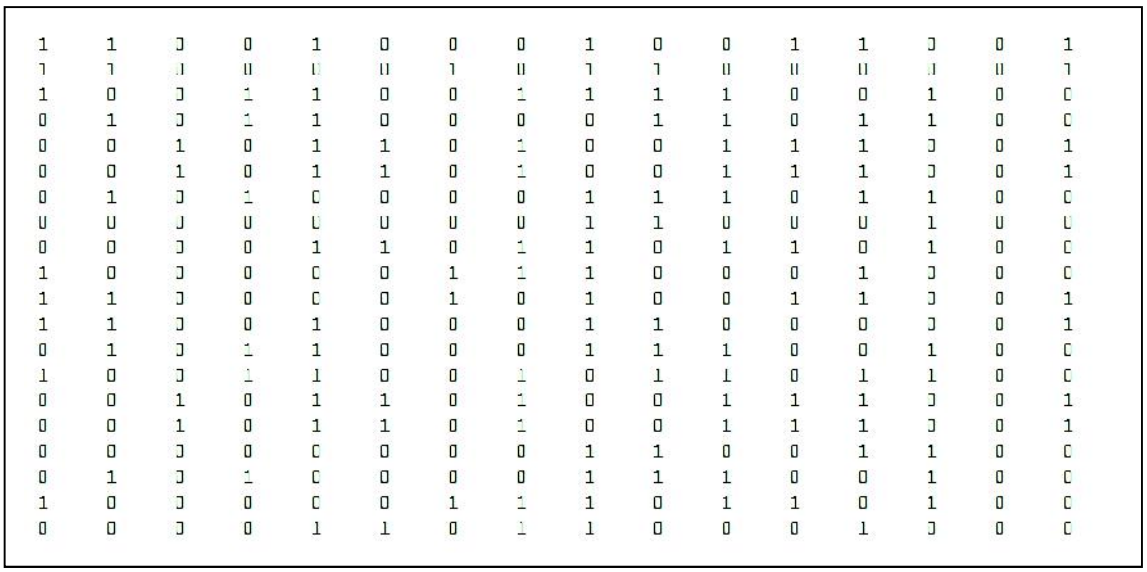
Les individus de la population intermédiaire vont se coupler 2 à 2 pour générer 10 nouveaux individus et comme sa on aura une nouvelle génération de 20 individus.

Le principe est de grouper les individus de la population intermédiaire 2 à 2 et d'appliquer un opérateur de croisement mono-point sur chaque couple (parents) pour créer deux nouveaux individus (enfants).

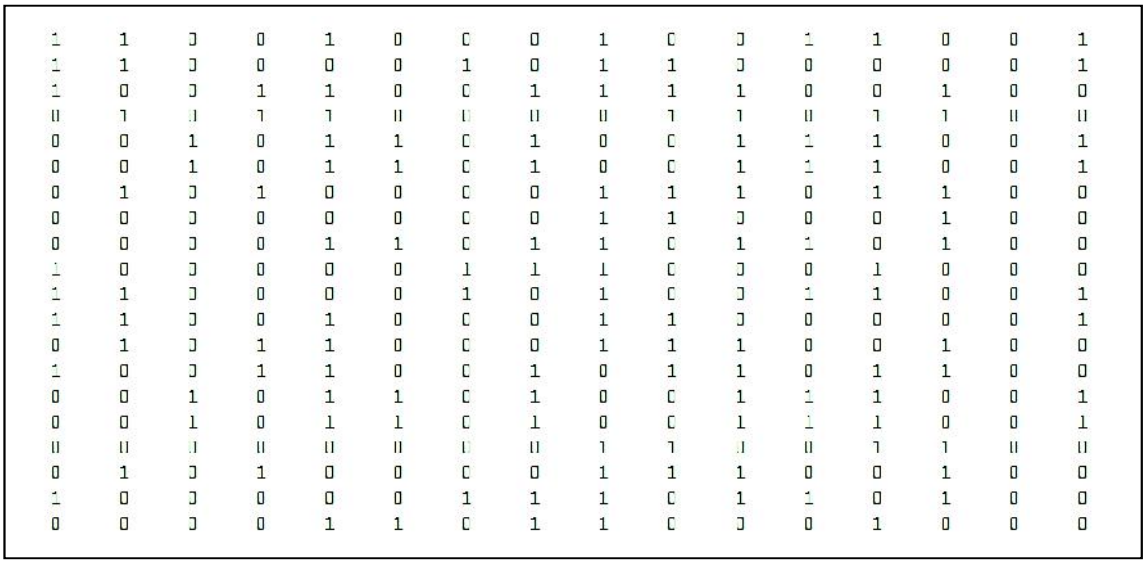
Pour cela on va choisir à chaque fois par hasard un site de croisement et de faire changer l'information entre les couples (parents), pour générer 02 nouveaux individus (enfants).



A la fin de cette phase on aura une nouvelle génération composée de 20 individus :



Après on fait intervenir la séquence de correction de poids s'il y on a des dépassements et on aura une génération correcte suivante :



la matrice de profits sera donc :

45	34	24	26	30	32
46	32	12	19	22	20
52	43	34	33	36	40
40	40	30	36	32	28
48	43	33	27	35	30
48	43	33	27	35	30
46	41	30	36	34	28
21	16	11	10	13	10
44	37	37	30	41	33
34	20	15	15	23	23
48	32	21	23	33	29
43	34	15	22	19	23
45	40	30	36	34	28
47	43	34	33	34	40
48	43	33	27	35	30
48	43	33	27	35	30
25	20	15	14	17	14
42	37	26	32	30	24
50	34	33	27	43	39
28	23	19	18	21	17

la matrice de poids:

47	45	50	33	25	32
38	34	50	37	24	25
46	36	43	40	40	35
43	41	48	32	32	26
40	38	46	32	24	42
40	38	46	32	24	42
47	42	53	34	28	26
19	18	20	24	10	18
38	36	33	34	22	49
28	19	29	24	19	21
47	46	52	37	24	30
38	33	48	33	25	27
44	39	50	35	33	33
45	38	41	37	39	28
40	38	46	32	24	42
40	38	46	32	24	42
25	23	27	27	12	20
41	37	46	31	26	24
45	40	36	39	26	34
21	15	26	19	15	36

valeurs d'évaluation :

30.4322
22.7436
38.5237
33.6028
34.2714
34.2714
34.5559
12.5345
35.6572
20.2019
28.7265
24.4493
34.3004
37.8261
34.2714
34.2714
16.5345
30.5559
35.8009
20.0582

Puis on fait intervenir la séquence d'ordonnancement ou de classification ; on aura une génération d'individus ordonnés selon leurs valeurs d'évaluation, On recalcule la matrice :

1	0	0	1	1	0	0	1	1	1	1	0	0	1	0	0
1	0	0	1	1	0	0	1	0	1	1	0	1	1	0	0
1	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0
0	0	0	0	1	1	0	1	1	0	1	1	0	1	0	0
0	1	0	1	0	0	0	1	1	1	1	0	1	1	0	0
0	1	0	1	1	0	0	1	1	1	1	0	0	1	0	0
0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1
0	1	0	1	1	0	0	1	0	1	1	0	1	1	0	0
0	1	0	1	0	0	0	1	1	1	1	0	0	1	0	0
1	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1
1	1	0	0	0	0	1	1	1	0	0	1	1	0	0	1
1	1	0	0	1	0	0	1	1	1	0	0	0	0	0	1
1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	1
1	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0
0	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0

Les profits de chaque individu seront donc :

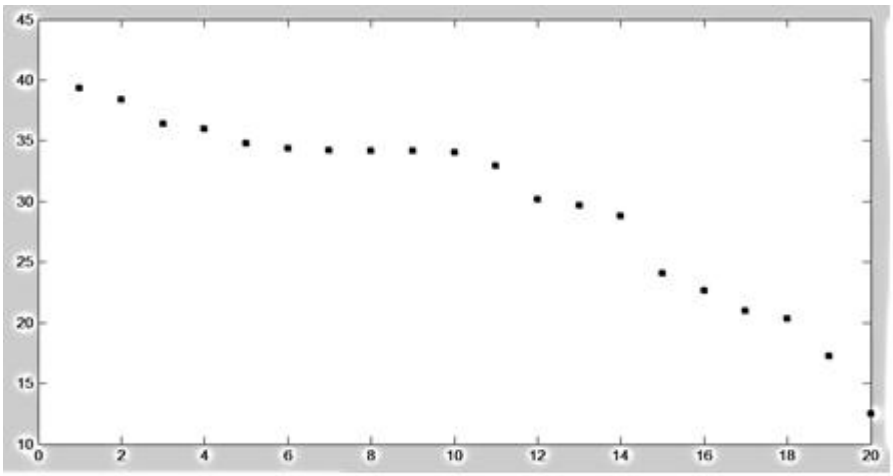
52	43	34	33	36	40
47	43	34	33	34	40
50	34	33	27	43	39
44	37	37	30	41	33
46	41	30	36	34	28
45	40	30	36	34	28
48	43	33	27	35	30
48	43	33	27	35	30
48	43	33	27	35	30
48	43	33	27	35	30
40	40	30	36	32	28
42	37	26	32	30	24
45	34	24	26	30	32
48	32	21	23	33	29
43	34	15	22	19	23
46	32	12	19	22	20
34	20	15	15	23	23
28	23	19	18	21	17
25	20	15	14	17	14
21	16	11	10	13	10

les poids de chaque individu seront :

46	36	43	40	40	35
45	38	41	37	39	28
45	40	36	39	26	34
38	36	33	34	22	49
47	42	53	34	28	26
44	39	50	35	33	33
40	38	46	32	24	42
40	38	46	32	24	42
40	38	46	32	24	42
40	38	46	32	24	42
43	41	48	32	32	26
41	37	46	31	26	24
47	45	50	33	25	32
47	46	52	37	24	30
38	33	48	33	25	27
38	34	50	37	24	25
28	19	29	24	19	21
21	15	26	19	15	16
25	23	27	27	12	20
19	18	20	24	10	18

le vecteur qui correspond aux valeurs d'évaluations (fitness) de chaque individu de la génération (population issue de la premiere itération de l'algorithme génétique) par ordre décroissant est :

38.5237
37.8261
35.8009
35.6572
34.5559
34.3004
34.2714
34.2714
34.2714
34.2714
33.6028
30.5559
30.4322
28.7265
24.4493
22.7436
20.2019
20.0582
16.5345
12.5345



Représentation des individus par leurs valeurs d'évaluations

Et comme sa on arrivera presque à terminer la première **itération** de l'algorithme génétique (reste l'opérateur de mutation), cette algorithme forme la première partie de notre algorithme globale (**GAS** $^{\textbf{MOKP}}$) c'est **la recherche globale.**

Après la recherche globale on fait appel à l'algorithme de colonie de fourmis pour réaliser la recherche locale (on effectue une recherche autour de chaque individu pour l'améliorer) c'est **la mutation.**

3.2.2/Recherche locale ; Mutation par algorithme de colonie de fourmis

Pour réaliser cette phase on va exploiter la stratégie de dépôt de traces de phéromone sur chaque objet sélectionné dans **S** (le sac à dos). Dans ce cas, lors de la construction des solutions suivantes, la probabilité de sélectionner chaque objet de **S** sera augmentée.

Pour trouver 20 nouvelles solutions dans une itération on utilisera 20 fourmis (chaque fourmi propose une solution pour développer un individu donné).
Pour chaque individu une fourmi va construire sa propre solution pour l'améliorer et pour cela elle va citer les objets qui ne sont pas pris en considération lors de la formation de ce dernier… C'est les **candidats ,** elle va sélectionner toujours les **candidats (objets)** qui vérifient les contraintes.

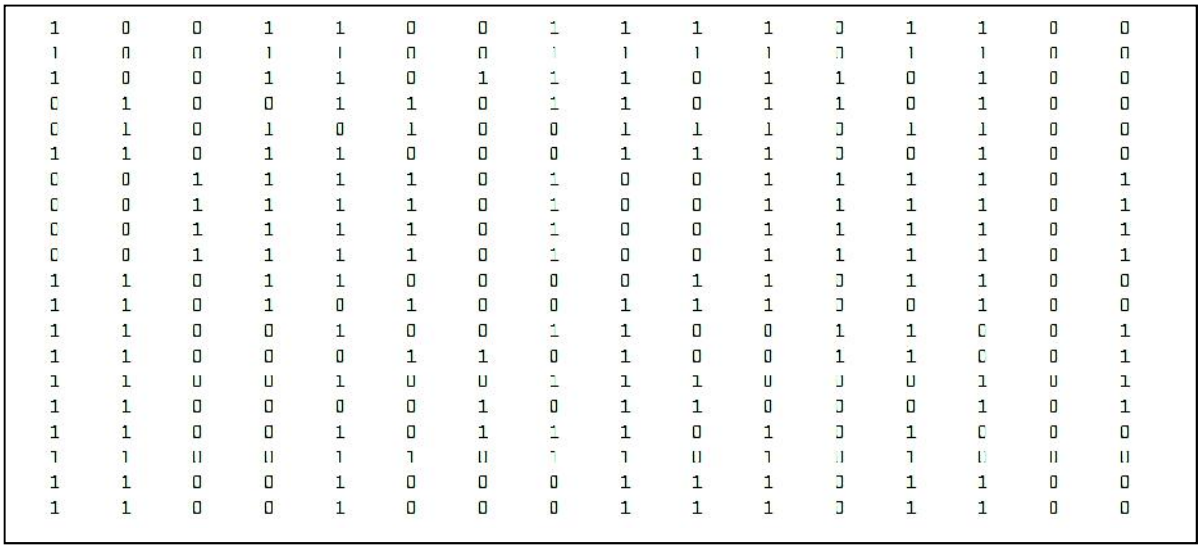
Pour cela on fait appel à un jeu d'instruction (une séquence de programme), pour qu'une fourmi puisse faire correspondre à chaque individu de la population ses candidats appropriés, On aura la liste des candidats pour chaque individu de la population (la liste étant un nouvel individu formé d'objets qui respectent bien les contraintes de poids de sac à dos).

Pour former les candidats d'un individu :

A une configuration donnée (solution),on fait correspondre les places vides dans le sac à dos (pour les faire compléter par des objets non encore choisit dans le sac) c'est-à-dire on identifie les indices des 0 dans cette configuration (individu) , et on commencera d'abord par l'ajout des objets qui ont des rapports **profit/poids** plus importants (**meilleurs candidats**) , et à chaque fois on recalcule

le poids du nouveau individu trouvé à l'égard des 06 contraintes de poids de sac à dos sus-exigées, et on rejet les objets dont leurs poids provoquent un dépassement dans le poids global de l'individu.

On aura tout de suite une complémentation des individus par leurs candidats appropriés (les objets possibles à intégrer dans la population tout en respectant les contraintes de poids),c'est la matrice vers laquelle va se converger l'algorithme de colonie de fourmis pour la première itération de l'algorithme génétique.



Cette population est complète parce qu'on a ajouté à tous les individus de la population issue de l'algorithme génétique, tous les objets (candidats) possibles **(sans violer les contraintes de poids de sac à dos).**

Voici la matrice de poids qui correspond (tous les individus respectent les contraintes de poids) :

52	41	50	43	42	37
52	41	50	43	42	37
52	46	44	45	44	45
48	47	47	36	23	50
48	43	54	38	29	40
52	43	52	40	39	34
49	50	52	40	36	48
49	50	52	40	36	48
49	50	52	40	36	48
49	50	52	40	36	48
51	45	50	37	38	27
50	42	49	40	33	39
51	49	55	35	27	34
48	47	53	41	25	44
47	45	55	41	28	33
43	42	52	43	25	29
49	36	55	33	31	34
47	34	50	29	26	41
54	44	55	41	30	34
54	44	55	41	30	34

La matrice de profits (elle s'est nettement améliorée bien sûr) :

56	47	38	37	40	44
56	47	38	37	40	44
56	40	40	34	50	46
52	45	39	39	45	35
50	45	34	39	38	30
52	43	33	39	37	39
54	49	42	35	44	38
54	49	42	35	44	38
54	49	42	35	44	38
54	49	42	35	44	38
47	43	33	39	35	39
53	44	33	38	37	37
53	42	27	29	33	35
52	36	25	26	37	31
54	45	24	30	28	31
49	35	18	24	28	25
55	41	31	38	41	38
53	44	34	40	38	39
53	44	34	40	38	40
53	44	34	40	38	40

le vecteur qui correspond aux valeurs d'évaluations (fitness) de chaque individus de la génération (population souhaité) issue de l'algorithme génétique :

```
40.0954
42.2936
43.3513
36.2329
40.8452
37.3835
37.3835
41.7266
33.0114
39.4471
33.7950
32.3421
40.3457
43.4069
37.9801
22.8278
27.5365
27.5365
43.4017
45.4017
```

Probabilités de choix

Le principe de recherche par le biais de **la stratégie phéromonale**, c'est de faire correspondre à chaque individu de la population actuel une fourmi pour lui améliorer, pour la première itération chaque fourmi va choisir **1 candidat** pour chaque individu de la population actuel selon la probabilité de choix calculés selon la formule ci-après:

$$p_{S_k}(o_i) = \frac{[\tau_{S_k}(o_i)]^{\alpha}.[\eta_{S_k}(o_i)]^{\beta}}{\sum_{o_j \subset Candidats}[\tau_{S_k}(o_j)]^{\alpha}.[\eta_{S_k}(o_j)]^{\beta}}$$

$$S_k \leftarrow S_k \cup \{o_i\}$$

à chaque candidat **(objet)** on fait correspondre une probabilité de choix selon la formule susmentionné tels que :

$o_{i=1.....16}$ représente les candidats qui vont être choisit par la fourmi **k** pour compléter une solution.

$S_{k=1.....20}$ représente la solution proposée par la fourmi k donc $p_{S_k}(o_i)$ c'est la probabilité d'un objet o_i pour qu'il soit choisit par la fourmi k pour former sa propre solution S_k en une itération de l'algorithme.

$[\tau_{S_k}(o_i)]$:l'intensité de trace de phéromone déposée sur l'objet o_i ,son importance relative est déterminé par le paramètre α ,il a été affecté à α =2.cette intensité est fixée en début de l'algorithme à $[\tau_{S_k}(o_i)] = \tau_{max}$ =5.

$[\eta_{S_k}(o_i)]$: un facteur heuristique ; son importance relative est déterminé par le paramètre β , il a été affecté à β =5.

$$\eta_{S_k}(j) = \frac{p_j}{h_{S_k}(j)}$$

Tel que p_j est le profit de l'objet **j** , $h_{S_k}(j)$ représente la dureté de l'objet j par rapport à toutes les contraintes $i \in 1..6$ relativement à la solution construite S_k, plus ce ratio est faible plus l'objet est intéressant.il est donné par :

$$h_{S_k}(j) = \sum_{i=1}^{m} \frac{r_{ij}}{d_{S_k}(i)}$$

$r_{ij\ i=1.....6}$ c'est le ressource (poids) de l'objet **j** à l'égard des 6 contraintes de poids sus-exigés. $d_{S_k}(i)$ est la quantité restante de la ressource lorsque la fourmi va choisir l'objet j en une solution s_k, à l'égard des 6 contraintes elle sera donnée par : $d_{S_k}(i) = b_i - \sum_{g \subset S_k} r_{ig}$ tel que b_i : est le poids à ne pas dépasser pour les 6 contraintes **(i=1……6)** et $\sum_{g \subset S_k} r_{ig}$ c'est la somme des poids des objets contenants la solution S_k.

Le nombre d'itération de l'algorithme est ainsi fixé **à LE =5 ,(** le nombre d'itération est faible puisque dans notre cas on a étudier un simple exemple de sac à dos ,un individu contient 16 objets au maximum….si on augmentera la taille de l'individu on aura due d'augmenter le nombre d'itération de l'algorithme pour que ce dernier converge vers la population souhaitée **,** dans notre cas notre algorithme converge très rapidement en environs de 5 itérations …c'est le nombre maximum de candidats sollicités par les fourmis dans leurs solutions…..

Mise à jours des traces de phéromone et Calcul de $[\tau_{S_k}(o_i)]$

$[\tau_{S_k}(o_i)]$ étant la quantité de trace de phéromone trouvé sur l'objet o_i , en début de l'algorithme on la calcule ,tous en considérant que **l'ancienne quantité** de phéromone déposés sur tous les objets de l'individu est fixée à $\tau_{t-1}(o_i) = \tau_{max}$ =5.

$[\tau_{S_k}(o_i)] = \tau_{S_k\ t}(o_i) = (1-\rho) * \tau_{t-1}(o_i) + \tau`_{S_k\ t}(o_i)$ c'est **la quantité actuel** trouvée sur l'objet o_i, tel que $0 \leq \rho \leq 1$ avec ρ est un facteur constant sert à réduire les traces de phéromone (c'est l'évaporation de phéromone),il est pris au hasard entre 0 et 1.

$\tau`_{S_k\ t}(o_i)$ c'est la quantité de phéromone actuellement déposée par la fourmi k sur l'objet o_i. cette quantité est définit comme suit :

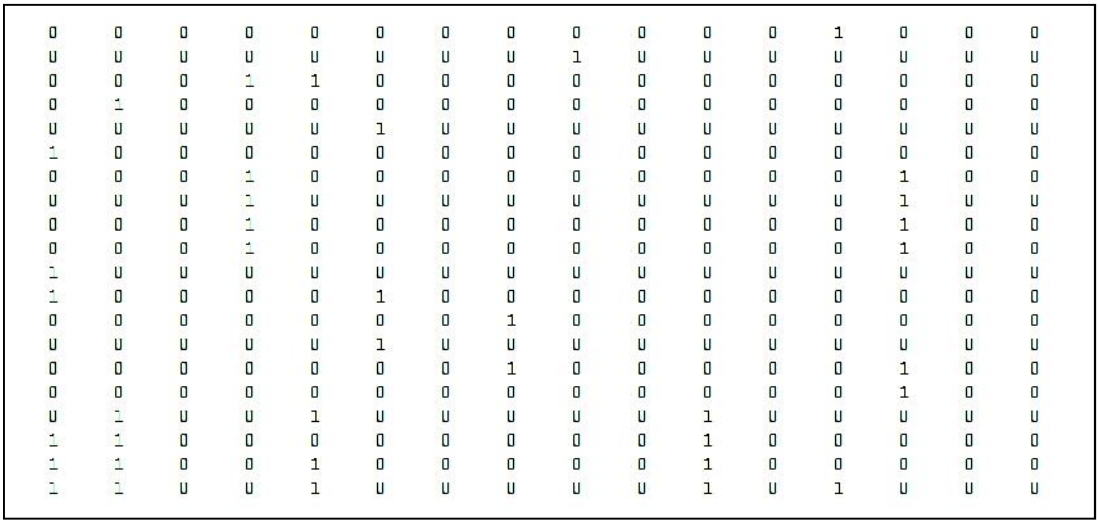
$\tau`_{S_k\ t}(o_i) = 1/(1 + \text{profit}(S_{best}) - \text{profit}(S_k))$. avec :

$\text{profit}(S_{best})$ c'est la valeur maximal des profits des individus de la population c'est le profit du meilleur individu .

$\text{profit}(S_k)$ c'est le profit de l'individu qui contient l'objet o_i sur lequel on va déposer de la nouvelle quantité de phéromone.

Identification des candidats pour chaque individu de la population pour la première itération

Si on fait la différence entre la population issue de l'algorithme génétique et la population complète obtenue on ajoutant tous les candidats possibles on aura la matrice des candidats suivante :



Donc on aura un candidat o_{13} pour la première fourmi , un candidat o_9 pour la deuxième fourmi ,deux candidats o_4,o_5 pour la troisième fourmi .....ainsi de suite.

Chaque fourmi construit une solution pour un individu pondant une itération (elle va choisir un candidat) donc il faut que le nombre d'itération soit supérieur ou égale le nombre de candidat d'une fourmi , dans notre cas l'algorithme converge rapidement puisque on a pris un simple exemple (la taille de l'individu est de 16 objets au maximum sa implique que le nombre de candidats sera très limité pour un individu donné)

3.2.2.1/Résultats pour une itération de l'algorithme de colonie de fourmis

On a 20 fourmis :

Pour chaque fourmi :

Commençant par la première fourmi :

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

Cette combinaison est la déférence entre la meilleur solution pouvant être obtenu par la fourmi et l'individu actuel ,il est claire que cette solution sera obtenu après une seule itération .le candidat ici est o_{13}.

Après on calcule par la formule suscitée les probabilités de chaque candidat pour qu'il soit choisit ,il faut d'abord calculer la quantité de trace de phéromone sur cet objet [$\mathbf{\tau s_1(o_{13})}$] (voir la phase de la mise à jour de phéromone sus-cités), et le

facteur heuristique $[\eta s_1$ **(o_{13})]** (voir la phase de probabilités de choix sus-cités) .il est claire qu'ici on va trouver $ps_1(o_{13})$=1 puisqu'ona un seul candidat **o_{13} .**

Dans le cas où on a plusieurs candidats on calcule pour chaque candidat sa probabilité pour qu'il soit choisit dans la solution et en suite on utilisera la roue de fortune pour prendre **un seul candidat** ;

le principe est de diviser une roue en parties dont chaque partie correspond a une probabilité de choisir un candidat , et après on fait tourner la roue et on prendra la valeur sur laquelle la roue s'est arrêté ,donc on va choisir le candidat qui correspond à cette valeur de probabilité.

Une foi un candidat a été choisi on modifie l'indice qui lui correspond dans l'individu actuel de 0 à 1.Et ainsi on répète l'opération avec tous les fourmis (20 fourmis) ; pour qu'elles puissent former leurs solutions, à la fin de **la première itération de l'algorithme** on aura la **population :**

1	0	0	1	1	0	0	1	1	1	1	0	1	1	0	0
1	0	0	1	1	0	0	1	1	1	1	0	1	1	0	0
1	0	0	0	1	0	1	1	1	0	1	1	0	1	0	0
0	1	0	0	1	1	0	1	1	0	1	1	0	1	0	0
0	1	0	1	0	1	0	0	1	1	1	0	1	1	0	0
1	1	0	1	1	0	0	0	1	1	1	0	0	1	0	0
0	0	1	0	1	1	0	1	0	0	1	1	1	1	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	1	0	1
0	0	1	0	1	1	0	1	0	0	1	1	1	1	0	1
0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	1
1	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0
0	1	0	1	0	1	0	0	0	1	1	1	0	1	0	0
1	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1
1	1	0	0	0	1	1	0	1	0	0	1	1	0	0	1
1	1	0	0	1	0	0	1	1	1	0	0	0	0	0	1
1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	1
1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0
0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

Une amélioration claire des individus de la population issue de l'algorithme génétique est obtenue après **la première itération de l'algorithme de colonie de fourmis** mais elle est proportionnellement médiocre si on la compare avec la population finale souhaitée comme le représente La **matrice de profits** ci-aprés :

56	47	38	37	40	44
56	47	38	37	40	44
53	37	37	31	47	43
52	45	39	39	45	35
50	45	34	39	38	30
52	43	33	39	37	39
51	46	39	32	41	35
51	46	39	32	41	35
51	46	39	32	41	35
51	46	36	30	38	33
47	43	33	39	35	39
46	41	30	35	34	26
53	42	27	29	33	35
52	36	25	26	37	31
51	42	18	25	22	26
49	35	18	24	28	25
44	30	25	25	33	32
38	33	29	28	31	26
35	30	25	24	27	23
31	26	21	20	23	19

Les poids de cette populations :

Tous les individus issues de différents itérations de l'algorithme de colonie de fourmis vérifient les contraintes de poids de sac à dos (les 06 contraintes sus exigés) , la matrice de poids issue de la **première itération de l'algorithme** est représentée ci-dessous :

52	41	50	43	42	37
52	41	50	43	42	37
48	42	40	43	33	43
48	47	47	36	23	50
48	43	54	38	29	40
52	43	52	40	39	34
45	46	48	38	25	46
45	46	48	38	25	46
45	46	48	38	25	46
44	42	50	34	35	44
51	45	50	37	38	27
42	38	47	35	27	38
51	49	55	35	27	34
48	47	53	41	25	44
42	37	53	35	27	29
43	42	52	43	25	29
36	23	37	27	23	24
29	19	34	22	19	39
33	27	35	30	16	23
27	22	28	27	14	21

le vecteur qui correspond aux valeurs d'évaluations (fitness) de chaque individus de la population issue de la première itération de l'algorithme de colonie de fourmis :

```
42.5237
42.5237
39.5454
40.8994
37.8945
39.4884
39.0643
39.0643
37.2714
39.0643
38.7908
33.8945
34.7097
32.0651
28.7268
27.5365
25.4441
29.8374
26.3137
22.3137
```

3.2.2.2/Résultats pour 05 itérations de l'algorithme de colonie de fourmis et comparaison avec ceux obtenus par un algorithme génétique classique

Si maintenant on relance l'algorithme pour le nombre maximum d'itérations que nécessite l'algorithme pour la convergence vers la population souhaité **(5 itérations dans notre exemple)** on aura les résultats suivants :

```
1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0
1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0
1 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0
0 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0
0 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0
1 1 0 1 1 0 0 1 1 1 1 0 0 1 0 0
0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1
0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1
0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1
0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1
1 1 0 1 1 0 0 1 0 1 1 0 1 1 0 0
1 1 0 1 1 1 0 1 1 1 1 0 0 1 0 0
1 1 0 0 1 0 0 1 1 0 0 1 1 0 0 1
1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1
1 1 0 0 1 0 0 1 1 1 0 0 0 1 0 1
1 1 0 0 1 0 1 1 1 1 0 0 0 1 0 1
1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0
1 1 0 0 1 1 0 1 1 0 1 0 1 0 0 0
1 1 0 0 1 0 0 1 1 1 1 0 1 1 0 0
1 1 0 0 1 0 0 1 1 1 1 0 1 1 0 0
```

On remarque que l'algorithme s'est convergé vers la population souhaitée après les 05 itérations, la matrice des profits des individus à l'égard des 6 fonctions objectifs est la même que celle-ci calculée précédemment :

56	47	38	37	40	44
56	47	38	37	40	44
56	40	40	34	50	46
52	45	39	39	45	35
50	45	34	39	38	30
52	43	33	39	37	39
54	49	42	35	44	38
54	49	42	35	44	38
54	49	42	35	44	38
54	49	42	35	44	38
47	43	33	39	35	39
53	44	33	38	37	37
53	42	27	29	33	35
52	36	25	26	37	31
54	45	24	30	28	31
49	35	18	24	28	25
55	41	31	38	41	38
53	44	34	40	38	39
53	44	34	40	38	40
53	44	34	40	38	40

Voici le vecteur qui correspond aux valeurs d'évaluations (fitness) de chaque individus de la population après 05 itérations de l'algorithme de colonie de fourmis, c'est le même que celui calculé précedement pour la population souhaitée :

40.0954
42.2936
43.3513
36.2329
40.8452
37.3835
37.3835
41.7266
33.0114
39.4471
33.7950
32.3421
40.3457
43.4069
37.9801
22.8278
27.5365
27.5365
43.4017
45.4017

La même chose pour la matrice de poids :

52	41	50	43	42	37
52	41	50	43	42	37
52	46	44	45	44	45
48	47	47	36	23	50
48	43	54	38	29	40
52	43	52	40	39	34
49	50	52	40	36	48
49	50	52	40	36	48
49	50	52	40	36	48
49	50	52	40	36	48
51	45	50	37	38	27
50	42	49	40	33	39
51	49	55	35	27	34
48	47	53	41	25	44
47	45	55	41	28	33
43	42	52	43	25	29
49	36	55	33	31	34
47	34	50	29	26	41
54	44	55	41	30	34
54	44	55	41	30	34

Après la mise en ordre de la population obtenue après 05 itérations de l'algorithme de colonie de fourmis (c'est la population souhaitée) on aura :

1	0	0	1	1	0	1	1	1	0	1	1	0	1	0	0
1	0	0	1	1	0	0	1	1	1	1	0	1	1	0	0
1	0	0	1	1	0	0	1	1	1	1	0	1	1	0	0
0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	1
0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	1
0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	1
0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	1	1	0	1	1	0	1	0	0
1	1	0	0	1	0	0	1	1	1	1	0	1	1	0	0
1	1	0	0	1	0	0	1	1	1	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	0	0	0
1	1	0	1	1	0	0	1	1	1	1	0	0	1	0	0
1	1	0	1	0	1	0	1	1	1	1	0	0	1	0	0
1	1	0	0	1	0	1	1	1	0	1	0	1	0	0	0
1	1	0	1	1	0	0	1	0	1	1	0	1	1	0	0
0	1	0	1	0	1	0	1	1	1	1	0	1	1	0	0
1	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1
1	1	0	0	1	0	0	1	1	1	0	0	0	1	0	1
1	1	0	0	0	1	1	1	1	0	0	1	1	0	0	1
1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	1

Voici les valeurs de profits de chaque individu après la convergence (en ordre du meilleur au mauvais individu) :

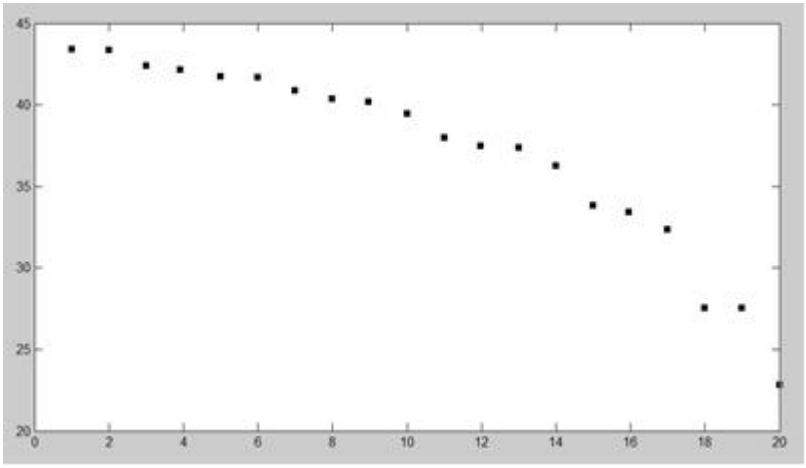
56	40	40	34	50	46
56	47	38	37	40	44
56	47	38	37	40	44
54	49	42	35	44	38
54	49	42	35	44	38
54	49	42	35	44	38
54	49	42	35	44	38
52	45	39	39	45	35
53	44	34	40	38	40
53	44	34	40	38	40
53	44	34	40	38	39
52	43	33	39	37	39
53	44	33	38	37	37
55	41	31	38	41	38
47	43	33	39	35	39
50	45	34	39	38	30
53	42	27	29	33	35
54	45	24	30	28	31
52	36	25	26	37	31
49	35	18	24	28	25

L’ordre apparaisse clairement dans le vecteur de valeurs d’évaluations (fitness) de chaque individus :

Les valeurs d’évaluation des individus sont nettement améliorées lors de 05 itérations de l’algorithme de colonie de fourmis .les résultats à gauche sont donc pour la première itération du GAS^{MOKP}, si on les compare avec ceux (à droite) obtenus par un algorithme génétique simple Classique pour sa 1ère itération : on constate une nette amélioration :

43.4069
43.3513
42.4017
42.2936
41.7266
41.6572
40.8452
40.3457
40.0954
39.4471
37.9801
37.3835
37.3835
36.2329
33.7950
33.0114
32.3421
27.5365
27.5365
22.8278

42.2467
40.7219
40.0954
40.0954
40.0954
36.4884
35.7410
35.7410
35.1767
34.3156
34.2714
33.2585
32.7439
31.5062
29.6728
29.5175
28.1011
27.5044
24.0064
17.2619



Représentation des valeurs d'évaluation des individus de la population après la première itération du GAS^{MOKP}

Voilà ci-dessous La matrice de poids (à l'égard des 6 contraintes de poids de sac à dos sus exigés) des individus -en ordre- qui sont issues de 05 itérations de l'algorithme de colonie de fourmis pour une itération de l'algorithme génétique **(c'est la même que celle de la population souhaitée calculée précédemment)** :

52	46	44	45	44	45
52	41	50	43	42	37
52	41	50	43	42	37
49	50	52	40	36	48
49	50	52	40	36	48
49	50	52	40	36	48
49	50	52	40	36	48
48	47	47	36	23	50
54	44	55	41	30	34
54	44	55	41	30	34
47	34	50	29	26	41
52	43	52	40	39	34
50	42	49	40	33	39
49	36	55	33	31	34
51	45	50	37	38	27
48	43	54	38	29	40
51	49	55	35	27	34
47	45	55	41	28	33
48	47	53	41	25	44
43	42	52	43	25	29

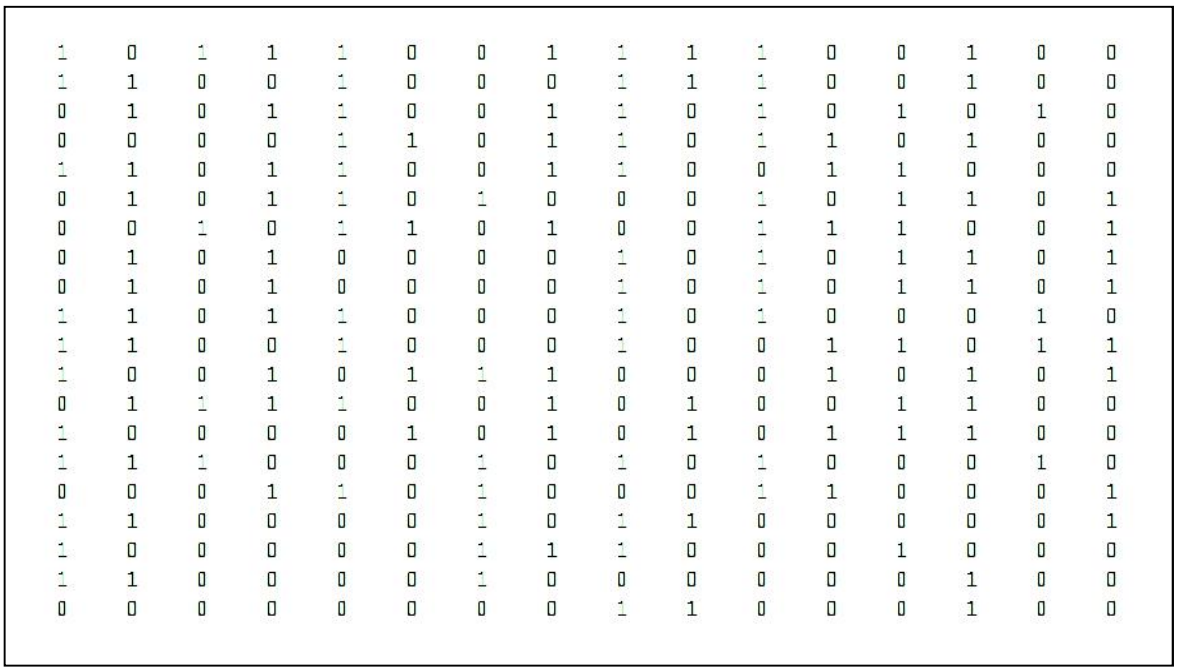
Les résultats exposés précédemment sont obtenus pour une itération de l'algorithme génétique , tous en fixant les configurations de la population initiales sur des valeurs prises au hasard , dans le but de comparer les résultats obtenus après plusieurs lancements du programme , on a même fixer le vecteur de poids λ, la population intermédiaire obtenu de la phase de sélection ,les sites de croisement , ….ext

3.2.3/Résultats de lancement du GAS^{MOKP} pour 05 itérations

Dans ce qui suit on présente les résultats obtenus lors de lancement de l'algorithme global (algorithme génétique pour la recherche globale + l'algorithme de colonie de fourmis pour la recherche locale) pour 05 itérations ;

On va rendre d'abord à notre algorithme son aspect heuristique, aléatoire tout en gardant la même population initiale, le même vecteur de poids λ :

Population initiale



Les matrices de valeurs de profits, de poids, et les valeurs d'évaluations ont été calculés précédemment ….,

Le meilleur individu de cette population c'est bien :

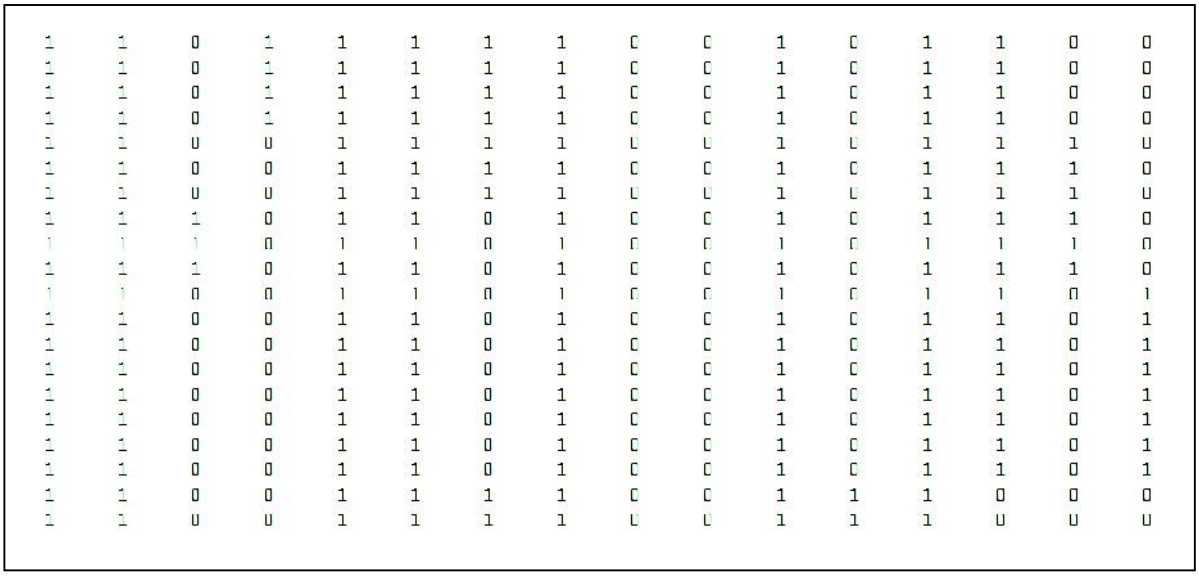
1 0 1 1 1 0 0 1 1 1 1 0 0 1 0 0

Sa valeur d'évaluation est de :

40.0954

Après 05 itérations on aura les résultats suivants

Population finale



Après le calcul de la matrice de valeurs de profits (pour les 06 fonctions objectifs) de différents individus de la population finale, nous avions les résultats ci-dessous :

56	47	40	45	48	44
56	47	40	45	48	44
56	47	40	45	48	44
56	47	40	45	48	44
54	45	38	43	46	42
54	45	38	43	46	42
54	45	38	43	46	42
53	46	38	43	40	42
53	46	38	43	40	42
53	46	38	43	40	42
54	50	37	42	39	41
54	50	37	42	39	41
54	50	37	42	39	41
54	50	37	42	39	41
54	50	37	42	39	41
54	50	37	42	39	41
54	50	37	42	39	41
54	50	37	42	39	41
57	46	37	38	47	42
57	46	37	38	47	42

La matrice de poids (pour les 06 contraintes de poids) de différents individus de la population finale est ainsi calculée ci-après :

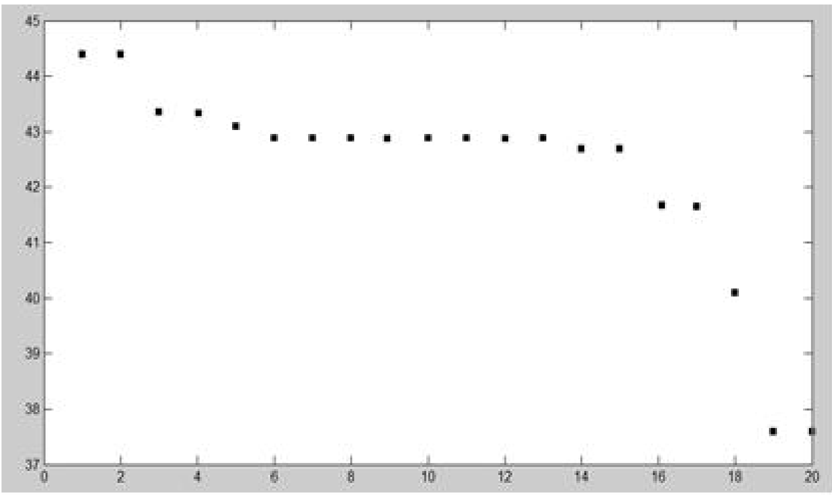
52	46	53	39	41	45
52	46	53	39	41	45
52	46	53	39	41	45
52	46	53	39	41	45
52	46	53	39	33	46
52	46	53	39	33	46
52	46	53	39	33	46
54	45	54	34	29	41
54	45	54	34	29	41
54	45	54	34	29	41
48	45	53	33	26	38
48	45	53	33	26	38
48	45	53	33	26	38
48	45	53	33	26	38
48	45	53	33	26	38
48	45	53	33	26	38
48	45	53	33	26	38
48	45	53	33	26	38
53	48	51	40	33	47
53	48	51	40	33	47

Les valeurs d'évaluation de chaque individu de la population finale sont :

44.3900
44.3900
43.3548
43.3548
43.0902
42.8877
42.8877
42.8877
42.8877
42.8877
42.8877
42.8877
42.8877
42.6843
42.6843
41.6455
41.6455
40.0954
37.5879
37.5879

Donc ces résultats sont obtenus lors de 05 itérations du GAS^{MOKP} , si on les compare avec ceux obtenus par un algorithme génétique simple (le tableau à droite) on constate une nette amélioration :

37.4418
35.9302
35.1819
34.7816
34.7816
32.8255
32.6921
31.8661
30.6273
30.6273
30.4322
29.9887
29.9887
27.0732
27.0440
27.0440
27.0440
26.3558
24.8228
22.2511



Représentation des valeurs d'évaluation des individus de la population après 05 itérations du GAS^{MOKP}

L'évolution rapide des individus de la population à travers les itérations successives de l'algorithme revient aux performances de cet algorithme d'un côté et au taille des individus qui est de l'ordre de 16 objets d'un autre côté. L'algorithme **GAS^{MOKP}** converge moins vite pour les populations qui appartiennent des individus de tailles beaucoup plus grandes.

3.2.4/Résultats du lancement du GAS^{MOKP} pour 10 itérations

Population finale

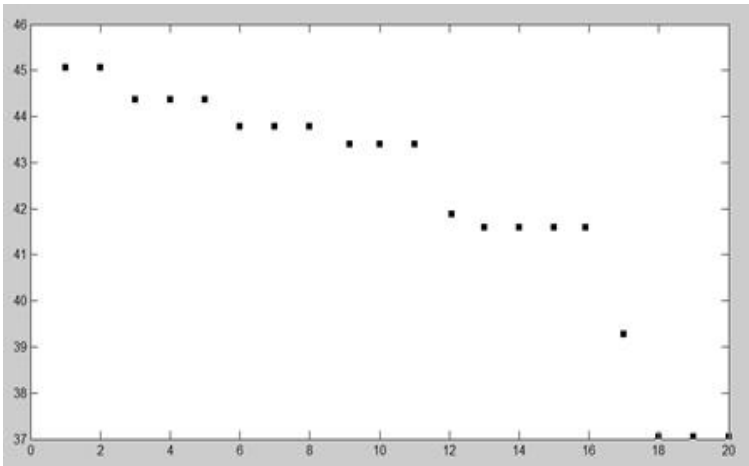
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0

Les valeurs d'évaluation de chaque individu de la population finale sont :

45.0605
45.0605
44.3629
44.3629
44.3629
43.7830
43.7830
43.7830
43.4069
43.4069
43.4069
41.6453
41.5848
41.5848
41.5848
41.5848
39.2813
37.0478
37.0478
37.0478

Les résultats à gauche sont pour le GAS^{MOKP} et ceux à droite sont obtenus par un algorithme génétique simple. On remarque que les individus de la population sont nettement améliorés :

37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
37.4444
36.6999
36.6999
33.4444
33.4444
31.9427
27.6652



Représentation de valeurs d'évaluation des individus de la population après 10 itérations du GAS^{MOKP}

3.2.5/Résultats du lancement du GAS^{MOKP} pour 30 itérations

Population finale

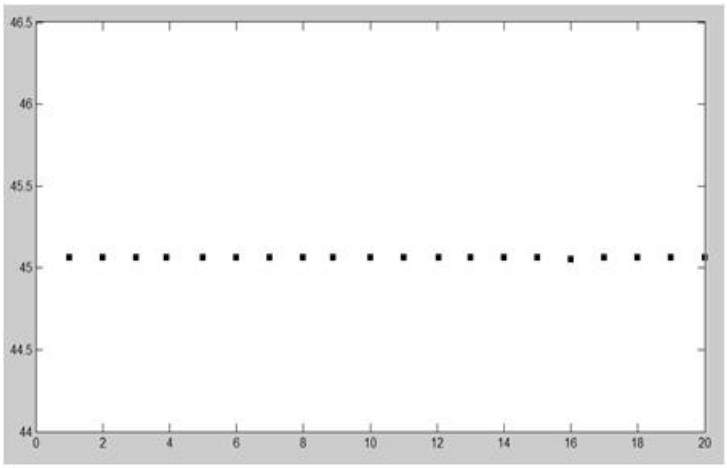
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0
1	1	0	1	1	1	1	1	0	0	1	0	1	1	]	0

On remarque que Tous les individus de la population ont convergé vers l'optimum trouvé, sa valeur d'évaluation est de 45.0605 il s'agit du meilleur **individu estimé :**

Si on compare ces résultats avec les résultats dans le tableau à droite obtenus par un algorithme génétique classique simple on verra clairement la différence:

45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605
45.0605

42.7488
41.5066
41.5066
41.5066
41.5066
41.5066
41.5066
41.5066
41.5066
41.5066
41.5066
37.5066
37.5066
37.5066
37.5066
37.5066
37.5066
37.5066
34.0254
34.0254



Représentation de valeurs d'évaluation des individus de la population après 30 itérations du GAS^{MOKP}

3.2.6/Comparaison du résultats de notre algorithme GAS^{MOKP} avec d'autre algorithme cités en références

m.n	rd	Vertex-AK		Path-AK		Edge-AK	
		Moyenne	Ecart	Moyenne	Ecart	Moyenne	Ecart
5.100	0,25	24192,0	33,13	24139,3	149,52	**24197,2**	37,52
5.100	0,50	43243,3	24,89	43137,9	137,11	**43246,4**	35,87
5.100	0,75	60470,5	25,37	60435,7	68,86	**60471,0**	28,35
5.250	0,25	**60335,9**	56,09	59110,6	911,77	60334,9	50,16
5.250	0,50	109203,4	39,09	107818	554,38	**109231,6**	47,97
5.250	0,75	151536,6	36,93	150578,6	152,68	**151536,9**	37,86
10.100	0,25	22553,0	60,97	22418,2	193,32	**22572,2**	61,81
10.100	0,50	42641,0	50,60	42383,0	192,37	**42658,5**	60,97
10.100	0,75	59540,0	34,71	59464,9	89,42	**59555,6**	44,70
10.250	0,25	**58847,6**	124,66	57367,3	593,35	58826,0	93,55
10.250	0,50	**108600,7**	109,50	106834,2	506,34	108594,1	78,03
30.100	0,25	**21618,1**	73,33	21411,4	213,22	21608,4	75,68
30.100	0,50	**41406,1**	69,05	41013,0	228,18	41403,3	68,53
30.100	0,75	59172,5	40,56	59049,9	111,09	**59173,2**	36,67
Moyenne		67642,2	55,33	67015,26	284,8	**67645,7**	54,21

Commentaire

Ces résultats sont pris avec α =1, β =5, ρ = 0.01 , le nombre de fourmis est fixé à **30** et les bornes de phéromone **zt max**=5, **zt min**=0.01, le nombre de cycles est de 2000.

Ces algorithmes sont appliqués sur 10 instances (individus), les résultats sont groupées en 5 classes **m.n** avec **m** $\in$**{5, 10,30} et n** $\in$**{100,250}, m :** est le

nombre de contraintes de poids de sac à dos, et **n :** le nombre d'objets contenants dans les instances. Les résultats sont obtenus pour des algorithmes avec un **rd =0.25** Et **rd= 0.50**, **rd=0.75**. Avec **rd** est le ratio de dureté.

Le tableau montre que les deux algorithmes **Vertex-AK** et **Edge-AK** trouvent des résultats beaucoup mieux que ceux trouvés par **Path-AK**.

Les résultats obtenus par notre algorithme GAS^{MOKP} sont de qualité supérieure surtout en durée de convergence qui est de l'ordre de quelques secondes, pour un exemple de sac à dos de 16 objets. Des exemples pris avec un nombre d'objets aussi important convergent moins vite que notre exemple.

Le tableau si dessus compare les temps d'exécution (en secondes) de différents algorithmes **Vertex-AK** et **Edge-AK** et **Path-AK**, pour les différentes classes .on constate que **Path-AK** est un peu plus rapide que les deux autres versions.

Conclusion

Les résultats obtenus lors de la simulation étaient très satisfaisantes, indiquant que l'algorithme **GAS^{MOKP}** à de très hautes performances , à savoir sa convergence vers l'optimum dans des délais très brefs .et sa grâces aux propriétés heuristiques caractérisant les approches approchés sur les quelles se base le **GAS^{MOKP} ; l'algorithme génétique et l'algorithme de colonie de fourmis.**

CONCLUSION GENERALE

Dans ce mémoire, nous avons résolus le problème de sac à dos multidimensionnel multiobjectif par le **GAS^{MOKP}**; un algorithme basé sur l'hybridation d'un **algorithme génétique** et un **algorithme de colonie de fourmis**.

Après une présentation de l'état de l'art des problèmes d'optimisation, et plus particulièrement les problèmes d'optimisation multiobjectif, en donnant les différentes définitions relatives à ces problèmes, on a mis l'accent sur les principales approches utilisées pour la résolution de ces problèmes et on a distingué 03 catégories d'approches : **les approches exactes** ,**les approches approchées** et **les approches hybrides** .

Dans le cadre de ce mémoire nous nous sommes intéressés par les approches approchées constituants des métaheuristiques, bien que ces approches donnent des résultats approximatifs, ils sont de qualités supérieurs néanmoins leurs temps de calculs qui est très satisfaisant, et pour cela nous avons présentés d'une manière plus explicite **les algorithmes génétiques** et **les algorithmes de colonie de fourmis** et nous avons expliqués les différents phases de développement de ces algorithmes.

Ensuite, on a abordé la résolution du problème de **sac à dos multidimensionnel multiobjectif**, nous avons définis les problèmes d'optimisation combinatoires et leur complexité théorique en premier lieu et on a pris le sac à dos multidimensionnel multiobjectif comme exemple de ces problèmes et après avoir définir ce problème, nous avons proposés pour sa résolution notre algorithme **GAS^{MOKP}**.

Le GAS^{MOKP} est basé sur l'hybridation de deux approches approchées ; l'algorithme génétique pour assurer la recherche globale donc **diversifier** la recherche et l'algorithme de colonie de fourmis exploité comme un opérateur de **mutation** de l'algorithme génétique globale donc ce dernier assure la recherche locale donc **intensifier** la recherche.

On a exploiter Le **GAS^{MOKP}** pour un exemple de sac à dos de dimensions de l'ordre de 16 objets et de **six fonctions objectif**s et **six contraintes de poids**, avec une population initiale de vingt individus .Les résultats obtenus lors de la simulation étaient très satisfaisants ,non seulement les solutions sont de qualités supérieurs ,le temps de calcul est de l'ordre de quelques secondes et cela grâce aux propriétés et avantages que nous apportent les métaheuristique et leurs hybridation .

Les résultats obtenus traduisent l'importance de **GAS^{MOKP}** pour être considérer comme un outil servant à la résolution d'autre problèmes d'optimisation combinatoires rencontrés dans différents domaines d'activités.et pourra être dans les futures travaux de recherches , un objet d' amélioration pour résoudre d'autre problèmes d'optimisation continus.

REFERENCES BIBLIOGRAPHIQUES

1-« **Optimisation multiobjectif par colonies de fourmis Cas des problèmes de sac à dos** » mémoire Présenté par Inès Alaya en vue de l'obtention du diplôme de Docteur en informatique –université **CLAUDE BERNARD LYON1.** 5 mai 2009

2-« Approches hybrides pour les problèmes multiobjectif » mémoire présenté par **Vincent BARICHARD** en vue de l'obtention du diplôme de Doctorat en informatique- **Laboratoire d'Etude et de Recherche en Informatique université d'Angers.** 24 Novembre 2003

3-« Application d'un algorithme hybride à colonies de fourmis au problème d'affectation quadratique » mémoire présentée par **ADIL KAMIL** dans le cadre du programme de maitrise en ingénierie de l'université du **QUEBEC A CHICOUTIMI.** février 2008

4-« **Approche Multi-Agents basée sur la Recherche Tabou pour le Job Shop flexible »** Article publié par **Meriem Ennigrou** de l'Institut Préparatoire aux Etudes d'Ingénieurs – El Manar-Tunisie et **Khaled Ghédira** de l'Ecole Nationale des Sciences de l'Informatique Tunisie.

5-« **Métaheuristiques populationnelles Algorithmes évolutionnaires** » et **« Ant Colony Optimisation »** présentation de cours sur les algorithmes génétiques et l'optimisation par colonie de fourmis (cours de **C. Pellegrini**) .

6-« **variantes des algorithmes de colonie de fourmis** »présenté par **HAO Wang –** session d'automne 2006 UQAM.

7-« **Optimisation par colonies de fourmis »** cours présenté par **COSTANZO Andrea** , **LUONG Thé Van et MARILL Guillaume -**19 mai 2006.

8-« **Application d'un algorithme de colonie de fourmis au problème du voyageur de commerce** » article publier le 1er juillet 2008 par **Pierre Schwartz** .

9-« **Algorithmes évolutionnaires parallèles pour l'optimisation multiobjectif de réseaux de télécommunications mobiles** » mémoire présenté par **HERVÉ MEUNIER** en vue de l'obtention du diplôme de Doctorat en informatique- université des sciences et technologies de **LILLE -** le 12/06/2002

10-« optimisation combinatoire multiobjectif : apport des méthodes coopératives et contribution à l'extraction de connaissances » mémoire pour obtenir le grade de habilitation à diriger des recherches de L'USTL-discipline : informatique – présenté par **CLARISSE DHAENENS- FLIPO** le 05 octobre 2005- université des sciences et technologies de **LILLE**.

11-« **optimisation et algorithme** » support de cours présenté par **VINCENT MAGNIN** Université de Lille 1 - 13 janvier 2001.

12-« **Algorithmes génétiques hybrides pour l'optimisation combinatoire** » article publié par **Charles Fleurent et Jacques A. Ferland –**Département d'informatique et de recherche opérationnelle Université de Montréal, Canada .08 novembre 1994.

13-« **Application des Algorithmes Evolutionnaires aux Problèmes d'Optimisation Multiobjectif avec Contraintes** » mémoire présenté par **Olga Roudenko** pour l'obtention du grade de docteur en mathématiques appliquée -05 mars 2004N

14-« Algorithmes à estimation de distribution et colonies de fourmis, mécanismes communs et hybridation avec une recherche locale » présentation de **Johann Dréo & Patrick Siarry** -Université Paris 12 -le 12/03/2004.

15-« partitionnement de territoire à l'aide d'un algorithme génétique » mémoire présenté par **Harold Waterkeyn** pour obtenir le diplôme du master en sciences informatiques /Année académique 2012-2013/université libre de Bruxelles, université d'Europe-faculté des sciences Département d'informatique.

16- « contribution à la résolution de problèmes d'optimisation combinatoire : méthodes séquentielles et parallèles » présenté et soutenue par **Mohamed Esseghir LAMALI** ,le 5 octobre 2012 pour obtenir le diplôme de doctorat de l'université de TOULOUSE.

17-« Métaheuristique Hybrides et Leurs Applications » présenté par **Lhassane IDOUMGHAR** le 21 novembre 2012 ; pour obtenir le diplôme d'Habilitation à Diriger des Recherches /Université de Haute Alsace.

18-« Méthodes de Résolution de Problèmes Difficiles Académiques » présenté par **AMIRA Gherboudj,** pour l'obtention du diplôme de doctorat 3ème cycle LMD, spécialité Informatique /Année universitaire 2012-2013.

19-« Conception de métaheuristique d'optimisation pour la segmentation des images de télédétection» présenté par **HOCINI Lotfi, pour** l'obtention du diplôme de Magister en Electronique ,le 14/11/2012.

20-« Optimisation Multiobjectif de l'allocation stratégique par un algorithme génétique » présenté par **Claire MONIN** le 7 janvier 2014 pour l'obtention du diplôme d'Actuaire de l'université de Lyon.