Université de Technologie de Belfort Montbéliard

Département informatique

IT45

Problèmes d'Affectation, de Planification et de Routage des tournées des Employés

---

Analyse et Conception

Présenté par :
Maureen Grandidier
Simon Nemo

Enseignants :
Mira Bou-Saleh
Olivier Grunder

Table des matières

Présentation du problème

1 Objectif

Notre objectif est de créer un circuit hamiltonien pour chacun des employés en :

— Harmonisant la charge de travail et la distance parcourue pour chaque employé

— Minimisant le nombre d'affectation dont la spécialité est insatisfaite.

— Minimisant le nombre d'heures supplémentaires, heures perdues, et la distance totale parcourue.

2 Contraintes dures

Les contraintes dures sont les contraintes qui se doivent d'être respectées dans la partition en sortie de l'algorithme.

— Toutes les missions doivent être affectées

— Une mission ne peut être effectuée que par un intervenant ayant les bonnes compétences

3 Contraintes souples

Les contraintes souples sont les contraintes qui peuvent être ou ne pas être respectées dans la partition en sortie de l'algorithme (elles sont considérées comme des "préférences").

— Chaque intervenant doit avoir une pause d'au moins 1h entre midi et 14h

— Respecter la limite des heures supplémentaires autorisées (10h/semaine, 2h/jour)

— L'amplitude d'une journée de travail ne doit pas dépasser 12h

— Un intervenant ne peut réaliser qu'une mission à la fois

— Un intervenant doit avoir le temps de se déplacer d'une mission à une autre.

— Une mission est effectuée par un et un seul intervenant (une mission n'est affectée qu'une fois)

— Respecter les heures maximales de travail par jour (Temps plein = 8h, temps partiel = 6h)

Chapitre I : Méthodes de résolution

1 Introduction

Ce problème s'apparente à un VRP ou Vehicle Routing Problem. C'est une extension classique du problème du voyageur de commerce, et il fait partie de la classe des problèmes NP-complet. Ce qui signifie que tous les algorithmes connus pour résoudre ce problème ont un temps d'exécution exponentiel avec la taille des données d'entrée, et sont donc inexploitables en pratique même pour des instances de taille modérée. Pour ce type de problème, on distingue les méthodes exactes et les méthodes approchées. Les méthodes exactes fournissent la meilleure solution mais sont limitées à des petites instances de problèmes. En effet, le temps de résolution augmente rapidement lorsque le nombre de variable de décision augmente également. Or, pour un établissement SESSAD, le jeu de données est beaucoup trop grand. Il faut donc mettre en place un algorithme permettant d'obtenir une solution approchée.

2 Heuristiques

Des méthodes heuristiques ont été conçues pour explorer seulement des parties de l'espace de recherche, se concentrant dans ces parties qui semblent contenir une amélioration des solutions. De ce fait, le temps requis pour obtenir une solution est réduit mais la solution trouvée est rarement optimale. De plus, ces algorithmes ne permettent pas de générer plusieurs solutions, une optimisation en cascade est alors impossible.

3 Métaheuristiques

Une métaheuristique est une heuristique générique qui peut s'appliquer à n'importe quel problème d'optimisation mais qui nécessite une adaptation au problème concerné (contrairement aux heuristiques qui sont en général associées à un problème particulier).

Les métaheuristiques partagent en général trois particularités :

— Elles partent d'une ou plusieurs solutions initiales générées aléatoirement dans la plupart des cas, ou construites à partir d'une heuristique dans certains cas.

— Elles mémorisent des solutions ou des caractéristiques des solutions visitées durant le processus de recherche afin de ne pas tomber dans des solutions déjà visitées ou dans un optimum local.

— Elles utilisent une procédure qui permet de créer une nouvelle solution à partir de la solution courante et des informations mémorisées. Il s'agit d'un élément essentiel de la recherche car ceci permet une meilleure exploration de l'espace des solutions.

En cours, nous avons vu 2 algorithmes de ce type :

— La recherche tabou

— Les algorithmes génétiques

3.1 Tabou

Les avantages de cet algorithme sont qu'il est très efficace sur de nombreux problèmes d'optimisation et qu'il est simple à comprendre et donc à mettre en place. Cependant, les paramètres peuvent être difficiles à régler, donc difficile à le rendre efficace et il est gourmand en ressource (taille de la liste tabou). Cet algorithme n'assure aucune convergence.

3.2 Génétique

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisements, mutations, sélection, etc. Les avantages de cet algorithme sont qu'il permet de créer de nombreuses solutions (intéressant pour une optimisation en cascade) et qu'il permet de jouer sur de nombreux paramètres. Cependant, ce type d'algorithme est difficile à mettre en place et n'assure en aucun cas d'avoir la solution optimale malgré un nombre d'itération élevé. Le temps de calcul peut être élevé car la fonction d'évaluation est évaluée de nombreuses fois. Il existe également le risque d'être coincé dans un optima local mais cet inconvénient peut être contourné en lançant plusieurs fois l'algorithme et en jouant avec les paramètres (ex : taux de mutation) permettant par ailleurs d'augmenter le nombre de solutions trouvées que l'on pourra utilisée pour l'optimisation en cascade.

L'algorithme génétique semble plus adapté à notre problème car il permet de produire de nombreuses solutions permettant une optimisation en cascade et est plus flexible. C'est donc cet algorithme que nous allons choisir. Il faudra prêter une attention particulière à la fonction d'évaluation afin de limiter le temps de calcul.

4 Codage des solutions

Chaque individu de la population représente une solution au problème à optimiser. Un individu est représenté par un chromosome (un chromosome représente une solution). Ce chromosome est constitué de gènes (un gène représente un élément d'une solution). A chaque individu est associé une évaluation (appelée aussi fitness) qui mesure la qualité de la solution. L'évaluation représente la performance de l'individu vis-à-vis du problème. La pertinence du codage va dépendre du choix des opérateurs de reproduction et l'efficacité globale de l'algorithme.

Il existe plusieurs types de codage pour les valeurs attribuées aux gènes : binaire, réelle, par structure arborescente ou encore par permutation.

Dans la permutation, les chromosomes contiennent une séquence de gênes, dans laquelle l'ordre est significatif, et où les gênes sont des nombres entiers.

Un chromosome ici est une suite de nœuds. Il y a autant de noeud que de missions, les chiffres représentent l'id des missions et ils sont classés dans l'ordre dans lequel les intervenants doivent réaliser les missions. Ce type de codage est appelé codage par liste de permutation.

Nous utilisons également un codage direct : toute l'information de la solution est présente dans le chromosome. Chaque permutation contient à la fois les missions et des séparateurs de tournées. Nous utilisons -1 comme séparateur de tournées. Le reste des chiffres représentent l'id des missions. Au départ, nous séparions chaque jour de la semaine pour chaque intervenant. Cependant, les missions doivent être exécutées à un certain horaire (et ont donc un ordre chronologique), il n'y avait donc pas besoin d'utiliser des séparateurs pour chaque journée comme nous l'avions prévu au départ. A la fin, il y a autant de tournées que d'intervenants.

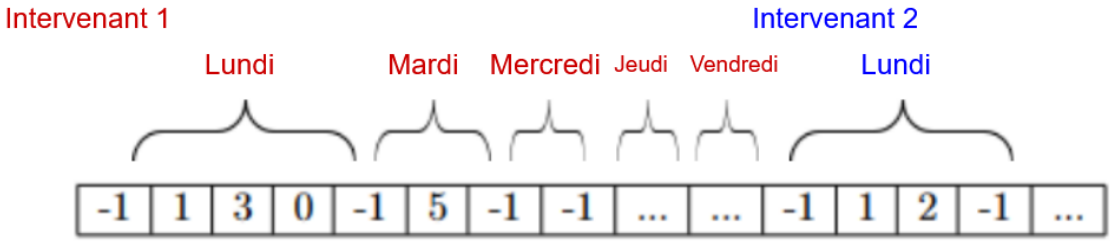

FIGURE 1 – Codage direct ancienne version

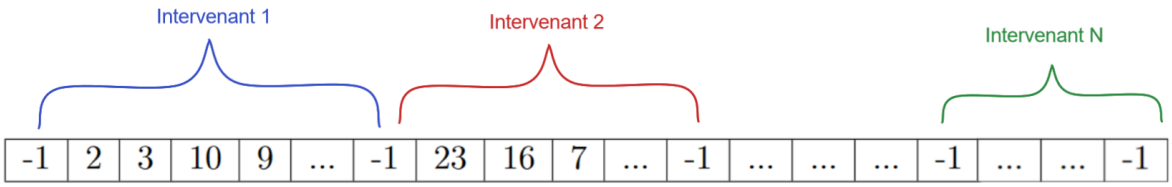

FIGURE 2 – Codage direct nouvelle version

Nous traitons les séparateurs comme les autres gênes lors des croisements ou autre, et nous appliquons une réparation sur les enfants si besoin.

5 GÉNÉRATION DE POPULATION

Au niveau de la génération de la population, on a le choix entre plusieurs possibilités :

— Initialisation aléatoire
— utilisation d'heuristiques

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Étant donné que notre optimum dans l'espace d'état du problème actuel nous est totalement inconnu, il est naturel de faire une initialisation aléatoire la plus uniforme possible afin de favoriser une exploration de l'espace de recherche maximum.

6 Fonction d'adaptation (fitness)

Notre problème est de type multi-objectifs, puisque nous cherchons par exemple à minimiser le nombre d'affectations dont la spécialité est insatisfaite, à atteindre un équilibre entre les employés, entre les heures supplémentaires de chacun, les heures non-travaillées, la distance parcourue etc.

L'algorithme optimise à partir de la 1ère fitness donnée et on applique des pénalités pour chaque contrainte souple non respectée. L'algorithme générera plusieurs solutions considérées comme "bonne" puis on optimisera en cascade avec le second critère, on retiendra de nouveau plusieurs solutions considérées comme les meilleures, puis on finira par optimiser avec le troisième critère pour obtenir la solution optimale qui convient aux trois fitness.

Voici les trois fitness que nous allons utiliser :

$$f_{employees} = \frac{\zeta \cdot \sigma_{WH}(s) + \gamma \cdot \sigma_{OH}(s) + \kappa \cdot \sigma_D(s)}{3}$$

$$f_{students} = \alpha \cdot penalties(s)$$

$$f_{SESSAD} = \frac{\beta \cdot sumWOH(s) + \kappa \cdot moyD(s) + \kappa \cdot maxD(s)}{3}$$

avec :

— $\sigma_{WH}(s)$ = écart type des heures non-travaillées (wasted hours) des employés pour s
— $\sigma_{OH}(s)$ = écart type des heures supplémentaires des employés pour s
— $\sigma_D(s)$ = écart type des distances des employés pour s
— $penalties(s)$ = nombre d'affectations dont la spécialité est insatisfaite pour s
— $sumWOH(s)$ = somme des heures non-travaillées et des heures supplémentaires de tous les employés pour s
— $moyD(s)$ = distance moyenne parcourue par les employés pour s
— $maxD(s)$ = distance maximale parcourue par les employés pour s

— $\alpha, \beta, \gamma, \zeta$ et κ sont des facteurs de corrélation.
$\alpha = 100$ / nombre total de missions dans le problème
$\beta = 100$ / nombre total d'heures qu'un employé peut travailler en semaine (45h)
$\gamma = 100$ / nombre total des heures supplémentaires tolérées (10h)
$\zeta = 100$ / moyenne des heures du quota du travail des employés
$\kappa = 100$ / moyenne de toutes les distances ($\frac{\sum_{m \in M}(d_{(center,m)}+d_{(m,center)}}{nombred'intervenants}$)

7 Sélection des individus

Le principe des algorithmes génétiques repose sur le principe de la sélection naturelle.

Les individus sont sélectionnés pour la phase de reproduction avec une probabilité proportionnelle à leur adaptation.

Il existe de nombreuses techniques de sélection, nous allons présenter celles que nous avons vu en cours et bien entendu notre choix final :

— La sélection aléatoire : comme son nom l'indique, ce type de sélection choisit le chromosome selon une distribution uniforme.

— La sélection par roulette : elle consiste à associer à chaque chromosome une probabilité d'être sélectionné proportionnelle à son fitness. Avec cette technique, les individus avec les meilleures fitness ont plus de chance d'être sélectionné. Cependant, avec cette méthode on risque d'obtenir un "super-héros", un individu avec une probabilité de sélection très élevée qui sera presque systématiquement toujours choisi (et fera perdre en diversité), ce qui empêche l'exploration de possible meilleures solutions.

— Sélection par tournoi : Cette technique tire au hasard deux ou plusieurs individus de la population et le plus fort est sélectionné, c'est-à-dire celui ayant le fitness le plus intéressant. Cette technique a comme avantage de ne pas avoir à trier la population et d'éviter le risque du "super-héros" mais le meilleur individu peut ne pas être sélectionné (champ d'exploration réduit)

Pour ce projet, nous avons choisi la sélection roulette.

8 Reproduction

Les 2 méthodes les plus courantes sont :

— Croisement : produire 1 ou plusieurs enfants à partir d'un ou plusieurs parents

— Mutation : un individu mute et donne un nouvel individu

D'autres méthodes existent mais sont beaucoup plus complexes : recherche locales, algorithmes de colonies de fourmis, règles basées sur l'indicateurs de similarité entre individus...

Il faudra que l'on fasse attention lors des croisements et mutations que les enfants produits respectent bien les contraintes. Pour cela, nous pourrons toujours déplacer les séparateurs jusqu'à trouver une solution. Et s'il est impossible de les "réparer" nous les exclurons.

Au niveau des croisements, il en existe plus de 22. Dans un premier temps, nous utiliserons le croisement en un point, et selon les résultats que nous obtiendrons, nous changerons de stratégie ou non. (Nous pensons utiliser sinon le Partially Mapped Crossover).

8.1 Croisement en un point

La première étape consiste à choisir aléatoirement un point de coupure pour partager chaque parent en deux parties. Puis le premier enfant est construit en utilisant la première partie du premier parent et la deuxième partie du deuxième parent. A l'inverse, le deuxième enfant est une concaténation de la seconde partie du premier parent et de la première partie du second parent.

Mais cet opérateur s'il est appliqué aux problèmes représentés sous forme de permutations, comme le VRP, a de fortes chances de produire des enfants invalides, par duplication et/ou omission de certains éléments de la permutation. On doit donc adapté cet opérateur.

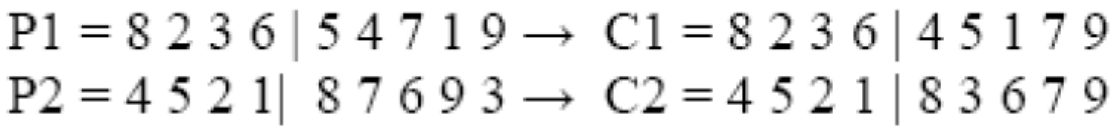

FIGURE 3 – Croisement en un point pour codage par permutation

8.2 Partially Mapped Crossover (PMX)

Le but de cet opérateur de croisement est de construire un enfant par le choix de sous séquences ordonnancées de l'un des parents et de préserver l'ordre et la position d'autant de sous séquences que possible des autres parents. La sous-séquence de l'ordonnancement est sélectionnée par le choix de deux points de coupure aléatoire, lesquels servent de frontière pour l'opération de substitution.

Considérons par exemple les deux parents :

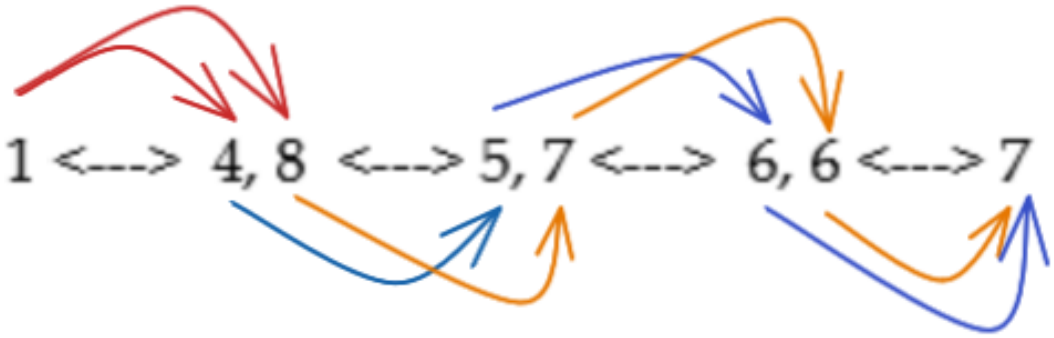

FIGURE 4 – Table de correspondance

P1 = 123 | 4567 | 89
P2 = 452 | 1876 | 93

Étape 1 : Ces deux parents vont produire deux enfants. Dans cette première étape, les segments compris entre les points de coupures sont échangés :

E1 = xxx | 1876 | xx
E2 = xxx | 4567 | xx

Notons qu'en vue de procéder à l'étape 3 de résolution des conflits, il est possible d'établir entre les deux points de coupures une table de correspondances entre les allèles. Cette table est définie comme suit dans notre exemple :

(ainsi après l'allèle 1 il y a aura soit l'allèle 4 soit 8, après le 4 il y aura le 5, après le 8 il y aura le 7, après le 5 ou 7 il y aura le 6...)

Étape 2 : les chromosomes enfants sont complétés en transmettant les allèles non conflictuelles issus du second parent.

E1 = x23 | 1876 | x9
E2 = xx2 | 4567 | 93

Étape 3 : cette dernière étape consiste à terminer l'élaboration des descendants en résolvant les conflits à l'aide de la table de correspondance :

E1 = 423 | 1876 | 59
E2 = 182 | 4567 | 93

Au niveau des mutations, il en existe en tout 6 types. Nous commencerons notre

programmation du problème avec l'opérateur de mutation par permutation qui est un des opérateurs de mutation les plus répandus. Bien entendu, nous changerons si besoin.

8.3 Mutation par permutation en codage réel

Lors de cette mutation, 2 positions sont sélectionnées au hasard et les gènes situés dans ces positions sont permutés.

Chromosome de départ : 1 **2** 3 4 5 6 7 **8** 9
Après permutation : 1 **8** 3 4 5 6 7 2 **9**

9 Remplacement

Après l'application des opérateurs de reproduction, les enfants sont évalués. Il faut ensuite déterminer la génération suivante (appelé remplacement) en déterminant quel individus devront disparaître de la population à chaque génération et quels individus fort survivent dans la population de la prochaine génération. Les quatre paragraphes suivants présentent quatre types de stratégies de remplacement.

9.1 Remplacement des stratégies d'évolution

Dans les stratégies d'évolution, la taille de la population de la prochaine génération est plus petite que la taille de la population des enfants. Soit on choisi les meilleurs individus de la population des enfants, soit on les choisit de la population des enfants et la population courante conjointes.

9.2 Remplacement de type Steady-state

A chaque génération, un (ou deux) enfant(s) seulement est (sont) généré(s). Ils remplacent le plus mauvais individu de la population courante. En d'autre termes, la population de la prochaine génération est la population courante, excepté le plus mauvais individu qui est remplacé par une ou deux solutions de la population des enfants. Ainsi la population de la prochaine génération est est peut-être plus grande que la population courante. Cependant, la plupart des algorithmes évolutionnaires sont appliqués sur des populations avec des tailles fixes en gardant au moins le meilleur individu dans la population courante.

9.3 Remplacement élitiste

L'élitisme est une façon de protéger la rémanence de bonnes solutions et d'assurer leur survie tout au long de la recherche. Ici, la population de la prochaine génération est choisie à partir de la population des enfants et de la population courante. Cette sélection à l'avantage de permettre une convergence plus rapide des solutions, mais au détriment de la diversité des individus.

9.4 Remplacement générationnel

La population des enfants remplace systématiquement la population courante. Ainsi, la population de prochaine génération est égale à la population des enfants.

9.5 Remplacement roulette

La probabilité qu'un individu soit remplacé est proportionnel à sa fitness. Ainsi, les individu ayant la meilleur adaptation ont une forte probabilité de ne pas être remplacés et les individus peu performant ont le plus de chance d'être remplacés par le fils. Dans ce type de remplacement, le fils est forcément dans la population suivante.

10 Conditions d'arrêt

Nous avons décidé mettre en place des conditions d'arrêts qui seront définies en argument dans la console.

Il y aura :

— Un critère d'arrêt sur le nombre d'itération (soit le nombre de génération)

— Un critère d'arrêt lorsque le temps maximal de calcul choisi est dépassé

Au critère d'arrêt, l'algorithme retourne la meilleure solution qu'il a obtenu.

Chapitre 2 : Implémentation

1 Définition des structures de données

1.1 Random

Nous permet simplement d'initialiser le générateur de variable aléatoire. En effet, l'aléatoire est un concept qu'on a du mal à traiter en informatique, étant données que nos machines sont des systèmes déterministes. Actuellement, l'aléa est simulé via des algorithmes mathématiques qui renvoient des suites de valeurs. Si ces algorithmes sont utilisés plusieurs fois à la suite, ils ont le défaut de toujours renvoyer la même suite de valeurs. Pour contourner ce problème, on initialise souvent ces algorithmes avec une valeur imprédictible : dans notre code, on utilise le temps système de la machine.

1.2 Distance

La classe distance permet d'encapsuler les données du fichier des Distances.csv dans une matrice. Elle contient 3 fonctions permettant de retourner la distance entre 2 missions, la distance entre une mission et le SESSAD et la distance entre le SESSAD et une mission.

1.3 Intervenant

La classe intervenant permet d'encapsuler les données d'un intervenant. La classe contient les 4 attributs :

- son ID
- sa compétence
- sa spécialité
- son temps de travail hebdomadaire

La classe contient également les getters et setters de ces attributs

1.4 Mission

La classe mission permet d'encapsuler les données d'une mission. Elle contient 6 attributs :

- son ID
- le jour de la mission
- l'heure du début de la mission (en minutes)
- l'heure de fin de la mission (en minutes)
- la compétence requise pour la mission
- la spécialité recommandée pour la mission

La classe contient également les getters et setters de ces attributs

1.5 Liste

La classe Liste est une classe développée lors des TP d'AP4A. Elle est plus simple à utiliser que la classe List de la STL et suffit largement pour ce projet.

1.6 Chromosome

Cette classe représente un chromosome. Elle génère le chromosome de manière plus ou moins aléatoire :

— le gène à une probabilité d'être une mission ou un séparateur en fonction du nombre de missions restant à affecter et le nombre de séparateur restant à affecter.

— Dans le cas ou le gène est une mission, celle-ci est choisie aléatoirement. Si la mission tirée est déjà affectée, on tire une nouvelle mission aléatoirement tant qu'on ne trouve pas une mission pas encore affectée.

Cette classe contient également toutes les fonctions associées à un chromosome :

— Fonctions d'évaluation du chromosome pour les 3 fitness du problème

— 2 fonctions permettant de vérifier que les contraintes dures soit respectées (toutes les missions sont affectées et l'intervenant doit avoir la bonne compétence)

— Une fonction permettant de réparer le chromosome lorsqu'une contrainte dure n'est pas respectée

— Une fonction qui permet de compter les pénalités à ajouter à la fitness 1 pour non respect des contraintes souples (cette fonction existe également avec l'affichage des contraintes non respectées)

— Une fonction qui retourne la distance parcourue par chaque intervenant

— Une fonction permettant d'afficher le chromosome et les informations associées à ce dernier

Enfin, cette classe contient les deux opérateurs de mutation :

— l'échange entre deux gènes quelconques

— l'échange de deux gènes entre deux intervenant ayant la même compétence

1.7 Population

Cette classe contient un ensemble de chromosomes et est de taille réglable. Elle permet d'obtenir des statistiques sur l'ensemble des chromosomes qu'elle contient. Elle permet également de les trier en fonction de leur fitness. Cette classe contient les fonctions de sélection des individus qui seront utilisée par la classe Ae afin d'effectuer des croisements et le remplacement dans la population de la génération suivante.

1.8 Ae

La classe Ae permet de déroulement de l'algorithme génétique :

— Lors de son initialisation, elle créer un population initiale.

— Pour chaque génération, l'algorithme sélectionne 2 parents par la sélection roulette puis de manière aléatoire effectue le croisement et/ou effectue une mutation puis met à jour la population.

La classe Ae contient l'opérateur de croisement que nous utilisons. Cet opérateur sélectionne aléatoirement une compétence. Le fils 1 est constitué des gènes du père 1 des intervenants n'ayant pas la compétence choisie et des gènes du père 2 des intervenant ayant la compétence choisie. Le fils 2 est constitué des gènes du père 2 des intervenants n'ayant pas la compétence choisie et des gènes du père 1 des intervenant ayant la compétence choisie.

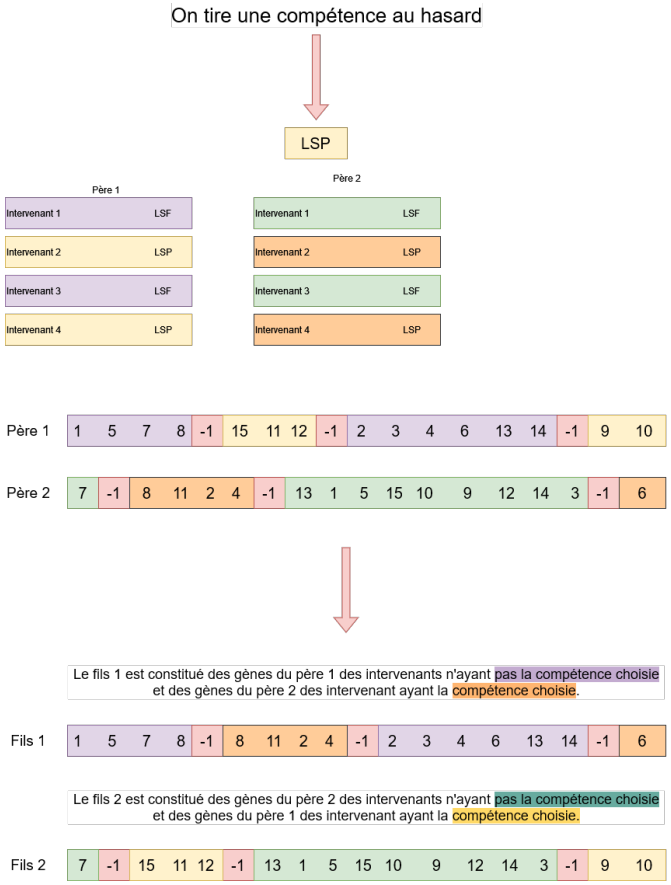

FIGURE 5 – Nombre générations - Temps d'execution

1.9 Main

Le main lance autant de fois possible l'algorithme durant le temps imparti. A la fin de chacun des lancements, les 10% meilleurs individus sont retenu dans une liste de chromosomes. A la fin du temps imparti, les meilleurs individus sur la fitness 1 sont contenu dans cette liste. On trie ensuite cette liste avec comme critère la fitness 2. On garde de nouveau les 10% meilleurs individus sur la fitness 2 parmi les individus de la liste. Puis enfin parmi les individus restant, on garde celui qui à la meilleure fitness 3.

Chapitre 3 : Étude expérimentale sur le paramétrage

Les algorithmes évolutionnistes dont les algorithmes génétiques sont des algorithmes fortement paramétrables : taille de la population, nombre de générations limite, taux de croisement et de mutation, type d'opérateur de sélection... L'objectif de cette étude expérimentale est d'analyser l'impact des différents paramètres sur nos résultats.

1 Configuration de l'ordinateur utilisé

— Système d'exploitation : Windows 11
— Type : PC à base de x64
— Processeur : AMD Ryzen 7 5800H with Radeon Graphics 3.20GHz, 8 coeurs
— Mémoire RAM installée : 16Go

2 45 missions et 4 intervenants

2.1 Nombre de générations

Les paramètres utilisés :
— Taille population : 20
— Taux de croisement : 0.8
— Taux de mutation : 0.5

On dit qu'un algorithme converge lorsque, au fur et à mesure des itérations, la sortie se rapproche de plus en plus d'une valeur spécifique. On peut voir que plus les générations sont nombreuses, plus les résultats tendent à être précis et à converger. Ce phénomène apparaît dès 1000 générations (on passe d'une moyenne de 59.6 à une moyenne de 3.6, ce qui n'est pas négligeable). Mais si on prend en compte le ratio meilleur résultat/temps d'execution, on peut voir que 50 000 générations semble être le meilleur paramètre de

nb génération	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
50	762,407	659,241	959,015	760,823	1361,61	900,6192	279,65676
100	604,256	462,078	657,739	1013,02	1115,33	770,4846	279,82134
500	6,61328	107,697	57,2412	57,5472	59,6651	57,752756	35,754841
1000	6,21647	6,7439	5,22654	3,57626	3,68707	5,090048	1,438974
10 000	2,72811	3,53522	5,87061	2,97607	3,53522	3,729046	1,2480264
50 000	4,15192	2,72811	3,53522	3,91258	3,15116	3,495798	0,5731225
100 000	2,72811	2,97607	3,53522	4,15192	3,53522	3,385308	0,5549834

FIGURE 6 – Nombre générations - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
0,002195	0,002255	0,003134	0,002841	0,002508	0,0025866	0,0003981
0,005202	0,004095	0,004254	0,004211	0,005852	0,0047228	0,0007715
0,012152	0,012342	0,012913	0,013462	0,015975	0,0133688	0,0015446
0,025714	0,02554	0,02691	0,025701	0,027256	0,0262242	0,0007964
0,222322	0,2245	0,218037	0,221207	0,221185	0,2214502	0,0023367
1,1711863	1,153935	1,140654	1,127592	1,132221	1,1451177	0,0176827
2,373971	2,326047	2,261835	2,362157	2,252991	2,3154002	0,0558888

FIGURE 7 – Nombre générations - Temps d'execution

génération pour cette instance. En effet, il propose en moyenne un des meilleurs résultats avec un des écart-type les plus faibles (donc avec un des dispersements autour de la moyenne la plus faible). Par ailleurs, son temps d'exécution est le plus faible.

2.2 Taille de population

Les paramètres utilisés :

— Nombre de générations : 50 000

— Taux de croisement : 0.8

— Taux de mutation : 0.5

Taille population	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
5	3,53522	3,91258	3,91258	3,53522	2,72811	3,524742	0,4836524
25	3,53522	2,72811	2,72811	3,53522	3,53522	3,212376	0,4420724
50	4,41669	3,53522	3,53522	2,72811	3,53522	3,550092	0,5973504
100	2,72811	2,72811	2,72811	2,72811	2,92654	2,767796	0,0887406
200	3,53522	2,97499	2,72811	2,72811	2,98197	2,98968	0,3296691

FIGURE 8 – Taille population - Meilleur résultat

Ici on peut voir que 100 se démarque des autres tailles de population. Cette taille offre le meilleur résultat (aussi bien niveau moyenne que écart-type) tout en gardant un temps d'exécution raisonnable. On peut voir ici que rien ne sert de prendre une taille de population trop élevé comme 200 car le résultat, en plus d'être moins précis, fait

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
1,320128	1,311004	1,320764	1,32702	1,325671	1,3209174	0,0063
1,420274	1,38871	1,411141	1,416522	1,405176	1,4083646	0,0123749
1,653367	1,689167	1,666435	1,688062	1,665931	1,6725924	0,0155397
2,305	2,385226	2,361267	2,376706	2,362097	2,3580592	0,0313353
4,757377	4,831076	4,872615	4,795492	4,815997	4,8145114	0,0426576

FIGURE 9 – Taille population - Temps d'exécution

augmenter le temps d'execution de près de 2s (nous avons une instance de 45 missions donc le temps d'exécution est assez faible, mais pour des instances plus élevées, cela peut être problématique)

3 96 MISSIONS ET 6 INTERVENANTS

3.1 Nombre de générations

Les paramètres utilisés :

— Taille population : 20

— Taux de croisement : 0.8

— Taux de mutation : 0.5

nb génération	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
50	4318.19	3911,35	4468,13	5224,57	4266,09	4467,535	554,68435
100	3416.51	4018	4274,04	3315,27	4215,08	3955,5975	440,70027
500	661.794	1966,16	1265,25	1459,64	1611,31	1575,59	296,40972
1000	208.639	360,677	208,862	1011,6	360,411	485,3875	358,02131
10 000	205.063	203,817	204,918	203,632	203,99	204,08925	0,5715111
50 000	203.25	203,375	203,373	203,573	203,886	203,55175	0,2417759
100 000	203.504	203,568	203,488	203,083	203,692	203,45775	0,2635531

FIGURE 10 – Nombre générations - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
0,011345	0,010616	0,010599	0,011045	0,010786	0,0108782	0,0003166
0,016059	0,018578	0,017888	0,016322	0,015287	0,0168268	0,0013609
0,059347	0,065766	0,058124	0,057856	0,057571	0,0597328	0,0034399
0,112143	0,113455	0,112595	0,112586	0,110619	0,1122796	0,001043
0,657565	0,612443	0,613777	0,626034	0,629511	0,627866	0,0181966
3,622334	3,099726	3,074655	3,099427	3,105005	3,2002294	0,2362578
6,347275	6,68334	6,261127	6,454932	6,271067	6,4035482	0,1745892

FIGURE 11 – Nombre générations - Temps d'execution

Ici on peut voir que 50 000 et 100 000 ont des résultats assez similaires (différence de 0.094) avec des écarts-type très proches également. La différence ici se trouve au niveau du temps d'execution (respectivement 3.2 et 6.4). Ainsi le meilleur nombre de générations pour cette instance est 50 000.

3.2 Taille de population

Les paramètres utilisés :

— Nombre de génération : 50 000

— Taux de croisement : 0.8

— Taux de mutation : 0.5

Taille population	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
5	203,521	203,321	204,376	203,222	204,409	203,7698	0,5786758
25	203,142	203,449	203,935	203,404	203,07	203,4	0,3406266
50	203,564	203,391	203,242	203,45	203,605	203,4504	0,1447111
100	203,654	203,588	204,055	203,201	203,192	203,538	0,3593501
200	204,394	205,339	205,093	204,273	205,415	204,9028	0,5348647

FIGURE 12 – Taille population - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
3,87378	3,959379	3,920628	3,848368	3,899855	3,900402	0,0427316
4,051811	4,052021	4,037327	4,086391	4,014693	4,0484486	0,0261095
4,307517	4,266472	4,299752	4,362991	4,296627	4,3066718	0,0351332
5,037158	5,035798	5,012041	5,023675	5,04031	5,0297964	0,011763
7,444246	7,395632	7,469622	7,45733	7,403493	7,4340646	0,032867

FIGURE 13 – Taille population - Temps d'exécution

Le meilleur résultat se trouve dans ce cas-ci entre 25 et 50. Le meilleur écart type se retrouve avec 50 mais on peut remarquer ici que si la taille de population est faible, la convergence se fait plus rapidement.

4 100 MISSIONS ET 10 INTERVENANTS

4.1 Nombre de générations

Les paramètres utilisés :

— Taille population : 20

— Taux de croisement : 0.8

— Taux de mutation : 0.5

nb génération	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
50	8978,27	10337,9	9486,33	8477,03	10245,9	9505,086	802,67842
100	10085,3	7633,41	9228,46	8094,88	9284,94	8865,398	988,3677
500	5675,57	6741,44	5168,62	5433,43	5337,6	5671,332	625,66821
1000	3218,76	3461,55	3967,26	2970,48	3334,1	3390,43	369,72712
10 000	1,54956	2,09949	52,8985	2,30854	2,33525	12,238268	22,731955
50 000	1,03271	0,971403	0,884487	1,76376	1,85839	1,30215	0,4687538
100 000	1,90609	1,02807	0,687623	1,2121	1,37981	1,2427386	0,4511093

FIGURE 14 – Nombre générations - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
0,013913	0,012361	0,013553	0,013181	0,013069	0,0132154	0,0005816
0,019329	0,018244	0,019168	0,017123	0,018908	0,0185544	0,0009009
0,06522	0,064245	0,064974	0,062951	0,067837	0,0650454	0,0017927
0,135894	0,116837	0,118057	0,118939	0,117344	0,1214142	0,0081329
1,109134	1,120839	1,10975	1,102	1,102335	1,1088116	0,0076483
5,710431	5,690489	3,258195	5,738691	5,83789	5,2471392	1,1132955
10,007149	10,020754	10,570089	10,064768	10,209794	10,174511	0,2352773

FIGURE 15 – Nombre générations - Temps d'exécution

Ici on peut voir que les résultats deviennent intéressants entre 50 000 et 100 000 (on passe de 12 à 10 000 à environ 1 dès 50 000). Le temps d'exécution est clairement inférieur pour 50 000 c'est donc le nombre de générations que nous retiendrons.

4.2 Taille de population

Les paramètres utilisés :

— Nombre de génération : 50 000
— Taux de croisement : 0.8
— Taux de mutation : 0.5

Taille population	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
5	1,69333	0,855434	0,74376	0,960952	1,76064	1,2028232	0,4852001
25	1,76502	1,05707	1,20505	1,97678	1,76053	1,55289	0,3983172
50	1,85724	2,0535	1,06803	0,736611	2,06639	1,5563542	0,6140566
100	1,681	1,87674	1,92995	1,5331	0,716406	1,5474392	0,4907682
200	2,48842	53,3039	2,08891	2,53836	105,091	33,102118	45,890282

FIGURE 16 – Taille population - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
3,921972	3,927714	3,927255	3,931462	3,920207	3,925722	0,0045756
3,938934	4,040329	4,022897	4,024647	4,005088	4,006379	0,0397167
4,225773	4,226477	4,220962	4,238185	4,227416	4,2277626	0,0063365
4,816767	4,923154	4,909183	4,936624	4,905798	4,8983052	0,0471931
7,443415	7,493914	7,275557	7,382423	7,307926	7,380647	0,0909734

FIGURE 17 – Taille population - Temps d'exécution

Le meilleur résultat se trouve ici plus vers 5 même si 25 reste un résultat plus que correct. Leur temps d'exécution est également semblable.

5 Conclusion sur le nombre de générations et la taille de population

En conclusion, le meilleur nombre de génération pour 45 missions et 100 missions est 50 000. Pour 96 missions, on avait retenu 50 000 et 100 000. Nous choisissons donc 50 000 comme nombre de générations définitif.

Pour la taille de population, elle est respectivement pour 45, 96 et 100 missions de 100, 25/50 et 5/25. Nous décidons de prendre 25 qui est la taille de population respectant au mieux le ratio performance / temps de calcul.

6 Opérateur de mutation

Les paramètres utilisés :

— Nombre de générations : 50 000

— Taille population : 100

— Taux de croisement : 0.8

Taux de mutation	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
0	1508,74	908,036	807,341	1211,27	1157,18	1118,5134	275,4563
0,25	2,97607	3,63037	2,72811	5,30765	2,72811	3,474062	1,0894221
0,5	2,72811	3,53522	2,72811	3,53522	2,72811	3,050954	0,4420724
0,75	2,72811	2,72811	2,72811	2,72811	3,53522	2,889532	0,3609506
1	2,72811	2,72811	2,72811	3,53522	2,72811	2,889532	0,3609506

Figure 18 – Taux de mutation - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
1,69571	1,662673	1,730958	1,613734	1,809066	1,7024282	0,073617
1,733126	1,708791	1,879925	1,732229	1,742924	1,759399	0,0685339
1,846655	1,866329	1,846462	1,849447	1,842889	1,8503564	0,0092274
1,968706	1,963274	2,272926	1,947345	1,966448	2,0237398	0,1395503
2,094513	2,075694	2,208373	2,035221	2,070086	2,0967774	0,0659637

Figure 19 – Taux de mutation - Temps d'execution

Les résultats entre 0.75 et 1 sont identiques 5 chiffres après la virgule. Leurs écart-types sont également identiques. La différenciation se fait sur le temps d'exécution, on choisit 0.75.

Au niveau de nos opérateurs de mutations, il y a une chance sur 2 que la mutation effectuée soit un échange entre 2 gênes quelconques (donc complètement du hasard) ou alors un échange entre 2 gênes d'intervenants ayant la même compétence. Nous avons décidé de modifier la probabilité de tomber sur l'échange entre 2 gênes quelconques pour voir les changements au niveau des résultats.

Les paramètres utilisés :

— Nombre de génération : 50 000

— Taille population : 100

— Taux de croisement : 0.8

— Taux de mutation : 0.75

proba aléatoire	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
0	56,1726	4,15192	3,15116	5,25334	5,25334	14,796472	23,146525
0,2	3,53522	3,53522	3,53522	2,72811	2,72811	3,212376	0,4420724
0,4	3,53522	2,178734	2,72811	2,72811	2,72811	2,7796568	0,4847564
0,6	2,72811	2,72811	3,53522	3,53522	2,72811	3,050954	0,4420724
0,8	4,85396	2,72811	2,472811	2,72811	2,72811	3,1022202	0,9854724
1	2,72811	2,72811	3,53522	3,53522	2,87624	3,08058	0,4194103

FIGURE 20 – Taux de mutation - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
1,691111	1,693551	1,69528	1,687543	1,73574	1,700645	0,0198326
1,849539	1,825487	1,8197	1,806921	1,813165	1,8229624	0,0164062
1,936251	2,72811	1,948542	1,889371	1,911075	2,0826698	0,3615353
2,002672	2,029836	2,280213	1,987781	2,034763	2,067053	0,1207206
2,105296	2,109348	2,138401	2,106421	2,174678	2,1268288	0,0300382
2,188104	2,190397	2,280335	2,184539	2,216934	2,2120618	0,0402673

FIGURE 21 – Taux de mutation - Temps d'execution

On peut voir que quand l'échange au hasard est à 0 (et donc qu'il n'est jamais choisi), la moyenne et l'écart-type sont les plus élevés par rapport aux autres résultats. A l'inverse, une probabilité de 1 (donc cet échange est toujours choisi), nous amène à un résultat proche du résultat optimal mais dans tous les cas bien inférieur à si on ne l'utilisait jamais. Le meilleur résultat en terme de moyenne se trouve à 0.4. Nous pouvons conclure qu'un algorithme sans échange aléatoire perdrait en terme de convergence et que pour obtenir des résultats optimaux il vaut mieux utiliser les deux de manière plus ou moins égale comme nous avons pu le faire.

7 OPÉRATEUR DE CROISEMENT

L'opérateur de croisement recompose les gènes d'individus existant dans la population. Les paramètres utilisés :

— Nombre de génération : 50 000

— Taux de croisement : 0.8

— Taux de mutation : 0.75

Taux de croisement	Meilleur résultat					Moyenne	Ecart-type
	ex1	ex2	ex3	ex4	ex5		
0	2,72811	3,53522	2,72811	3,53522	3,53522	3,212376	0,4420724
0,2	3,53522	2,72811	3,53522	3,53522	3,53522	3,373798	0,3609506
0,4	2,72811	2,72811	2,72811	2,72811	3,53522	2,889532	0,3609506
0,6	2,72811	2,72811	3,53522	2,72811	2,72811	2,889532	0,3609506
0,8	2,72811	3,53522	3,53522	3,53522	3,53522	3,373798	0,3609506
1	3,53522	2,72811	2,72811	2,72811	4,41669	3,227248	0,7511716

FIGURE 22 – Taux de croisement - Meilleur résultat

Temps d'execution (en s)					Moyenne	Ecart-type
ex1	ex2	ex3	ex4	ex5		
2,043718	1,956286	2,076158	1,918122	1,957133	1,9902834	0,0665052
2,074181	2,010951	1,919324	1,9291	1,940298	1,9747708	0,0662192
1,963558	1,980883	1,936669	1,941971	1,963346	1,9572854	0,0179756
2,054853	1,942567	1,948999	1,936855	1,974543	1,9715634	0,0487356
2,051646	1,990027	1,948495	1,99159	1,958852	1,988122	0,040243
1,942693	1,94492	1,959393	1,96804	1,986528	1,9603148	0,0179944

FIGURE 23 – Taux de mutation - Temps d'execution

Nous avons un seul opérateur de croisement qui croise aléatoirement une compétence. Nous pouvons voir que le meilleur résultat se trouve entre un taux de croisement de 0.4 et de 0.6. Ces deux taux de croisement ont les mêmes moyennes et les mêmes écart-types. Même leur temps d'exécution est extrêmement proche. Nous prendrons donc un taux de croisement à 0.5.

8 CONCLUSION SUR NOS PARAMÈTRES

A la fin, nos paramètres définitifs sont :

— Nombre de génération : 50 000

— Taille de population : 25

— Taux de croisement : 0.75

— Taux de mutation : 0.5

Conclusion

En conclusion, nous sommes satisfaits du travail effectué. Nous réussissons à avoir des résultats satisfaisants dans un temps raisonnable.

Les apports de l'optimisation et de la recherche opérationnelle sont présents dans beaucoup d'aspect de notre société : l'organisation des lignes de production d'automobiles ou la planification des missions spatiales par exemple. Mais aussi dans nos vies de tous les jours pour le recyclage des déchets, l'organisation des ramassages scolaires, les emplois du temps des employés ou la couverture satellite des téléphones portables... Ce projet nous a permis de mettre un pied dans le domaine de l'optimisation et de la recherche opérationnelle, et plus particulièrement dans l'optimisation de plusieurs objectifs et le respect de nombreuses contraintes.

Références

Table des figures