Gestion centralisée d'affectation et d'ordonnancement des missions

Ce chapitre a pour objectif de proposer des algorithmes efficaces pour la gestion centralisée de l'affectation et l'ordonnancement des missions. Le problème a été modélisé mathématiquement dans le chapitre 5. En raison de la complexité du problème et des limitations de l'approche de programmation linéaire du chapitre 5, d'autres approches sont nécessaires.

Notre problème d'affectation et d'ordonnancement est similaire aux problèmes de tournées de véhicules (*Vehicle Routing Problems, VRP*) où les véhicules sont les robots et les clients sont les missions. Il est particulièrement proche des problèmes de tournées de véhicules avec contraintes de temps (*Vehicle Routing Problems with Time Windows*, VRPTW) tel que ceux décrits au chapitre 5. La nécessité de la gestion d'énergie ajoute une couche de complexité supplémentaire à notre problème.

Une étude bibliographique montre que les algorithmes génétiques sont particulièrement adaptés aux problèmes de tournées de véhicules. Dans la suite de ce chapitre, nous proposons un algorithme génétique pour la gestion centralisée d'affectation et d'ordonnancement des missions.

6.1. Algorithmes génétiques et évolutionnaires

L'algorithme génétique est une approche d'optimisation par évolution d'une population de solutions à travers des opérateurs comme le croisement et la mutation. La structure des algorithmes évolutionnaires utilisée dans ce chapitre est illustrée par le pseudo code suivant :

Algorithme génétique:

1. Générer une population initiale de μ solutions admissibles;
2. Coder en chromosomes les solutions de la population initiale;
3. Evaluer la fontion de fitness de chaque solution;
4. Sélectionner λ couples des chromosomes parents pour la reproduction;
5. Appliquer un croisement à chaque couple des parents pour obtenir des chromosomes enfants;
6. Appliquer une mutation à chaque chromosome enfant avec une probabilité de mutation P_m;
7. Décoder les chromosomes enfants;
8. Evaluer la fontion de fitness de chaque solution;
9. Sélectionner μ chromosomes enfants ou parents pour former la nouvelle génération;
10. Répéter les étapes 4-9 jusqu'à l'arrêt.

Algorithme 1 : Algorithmes Génétiques

Dans un algorithme génétique, chaque solution est représentée par un chromosome qui peut être par exemple une suite de lettres d'un alphabet donné. La représentation des solutions en chromosomes dépend fortement des problèmes étudiés et joue un rôle important dans l'efficacité de l'algorithme génétique.

L'algorithme génétique résout un problème d'optimisation à travers l'évolution d'une population de solutions afin d'identifier les caractéristiques communes des bonnes solutions et de diversifier la recherche. Il commence par une population initiale de solutions souvent générées de manière aléatoires avec différentes heuristiques. Après la génération de la population initiale, l'algorithme génétique fait évoluer la population à l'aide de différents opérateurs génétiques. Chaque population est donc appelée une génération.

A chaque itération / génération, chaque nouvelle solution est évaluée afin de déterminer sa qualité. La fonction d'évaluation connue aussi sous le nom *fitness function* ou fonction de fitness peut être soit la valeur du critère à optimiser soit une fonction de la valeur du critère. Nous associons à chaque solution une valeur de fitness *f*. La fonction de fitness est définie comme suit :

$$f = \begin{cases} Critère, & \text{si le gain par palier est utilisé,} \\ 1/Critère, & \text{si la pénalité continue est utilisée} \end{cases}$$

où *Critère* est défini par l'équation (5-1) pour le cas du modèle de gains par palier et par l'équation (5-20) pour le cas du modèle de la pénalité continue.

Dans ce chapitre, chaque solution est représentée de manière indirecte par un chromosome en utilisant un algorithme de codage pour la conversion. Avant l'évaluation d'un chromosome, un algorithme de décodage est alors nécessaire pour convertir un chromosome en une solution explicite et complète d'affectation et d'ordonnancement des missions. Plus précisément, chaque génération est donnée sous la forme de chromosomes qui peuvent être décodés pour représenter une solution qui est un ensemble de planning pour les robots. La fonction fitness d'un chromosome est la somme de la fonction objectif de cette solution qui peut être directement calculé en additionnant tous les retards et les latences selon les méthodes présentées dans le chapitre précédent.

L'évolution de la population commence par la sélection des chromosomes pour la reproduction de nouveaux chromosomes. Les chromosomes sélectionnés sont ensuite croisés selon un opérateur de croisement et ensuite mutés par un opérateur de mutation. Après la phase de mutation et d'évaluation, on applique les opérations de sélection pour le remplacement qui consiste à choisir un sous ensemble des solutions performantes nouvellement générés (Chromosome enfants) pour remplacer une partie des solutions de la génération parent.

Cette boucle générationnelle est relancée plusieurs fois jusqu'à ce que l'un des critères d'arrêt soit atteint qui peut être soit le nombre maximum d'itérations ou les générations, soit le temps de calcul maximal, ou même si une solution optimale est trouvée.

Nous donnons dans la suite de ce chapitre les différents composants de notre algorithme génétique pour la gestion centralisée d'affectation et d'ordonnancement des missions. La génération de la population initiale dépend fortement du problème d'optimisation concerné et ne sera pas détaillé dans ce chapitre. Dans notre étude, nous nous contentons d'une génération aléatoire de solutions.

6.2. Codage génétique des solutions par chromosomes

Comme nous avons indiqué dans le chapitre précédent, une solution de la gestion centralisée d'affectation et d'ordonnancement des missions est définie par (i) l'affectation des missions à l'ensemble des robots, (ii) l'ordre dans lequel chaque robot exécute ses missions, et (iii) la date de début de chaque mission.

Compte tenu des critères retenus dans le chapitre précédent, les dates de début des missions peuvent être déduites directement de (i) et (ii). Connaissant les missions à exécuter par un robot et l'ordre d'exécution, les dates de début des missions correspondent aux dates au plus tôt d'exécution des missions en respectant l'ordre donné et les dates de disponibilités des missions. Plus précisément,

$$s_{[i+1],n} = MAX\left(s_{[i],n} + T_{[i]} + A_{[i],[i+1]}, es_{[i+1]}\right) \qquad (6.1)$$

où [i] est la i-ième mission du robot n, $s_{[i],n}$ sa date de début, $T_{[i]}$ la durée de la mission [i], $A_{[i],[i+1]}$ le temps nécessaire pour aller de la position de fin de la mission [i] à la position de début de la mission [i+1], $es_{[i+1]}$ la date au plus tôt de la mission [i+1].

Pour résumer, une solution d'affectation et d'ordonnancement est complètement caractérisée par l'affectation des missions aux robots et l'ordre de passage des missions sur chaque robot. Elle est dite **admissible** si chaque mission *m* est affectée à un robot *n* équipé de module fonctionnel nécessaire, i.e. $f_m \in C_n$ et chaque robot dispose suffisamment d'autonomie énergétique pour ses missions, i.e. $\sum_{m \in M} x_{nm} E_m \leq E_{n,\max}$.

Afin de simplifier la représentation génétique, dans ce chapitre, nous utilisons une représentation indirecte d'une solution d'affectation et d'ordonnancement. Cette représentation est faite avec un seul chromosome qui est une suite de symboles appartenant à un alphabet L. L'alphabet L contient l'ensemble des missions à affecter plus des symboles particuliers à préciser. Les positions dans le chromosome des missions d'un même robot doivent respecter l'ordre d'exécution de ces missions.

Le processus de transformation d'une solution d'affectation et d'ordonnancement en un chromosome est appelé le codage. A l'inverse, la transformation d'un chromosome en une solution complète d'affectation et d'ordonnancement est appelé décodage. Trois codages et décodages sont utilisés dans ce chapitre.

Codage parallèle avec séparateurs

Cette représentation utilise un symbole spécial "**f**", appelé séparateur, pour indiquer la fin d'un planning d'un robot. Ce symbole est ajouté à la fin de chaque séquence d'exécution. Le chromosome est obtenu par concaténation des séquences d'exécution des différents robots, chacune terminant par le symbole "**f**". Voir la Figure 27 pour une illustration où le chromosome est {2, 5, **f**, 3, 1, 6, **f**, 4, **f**}.

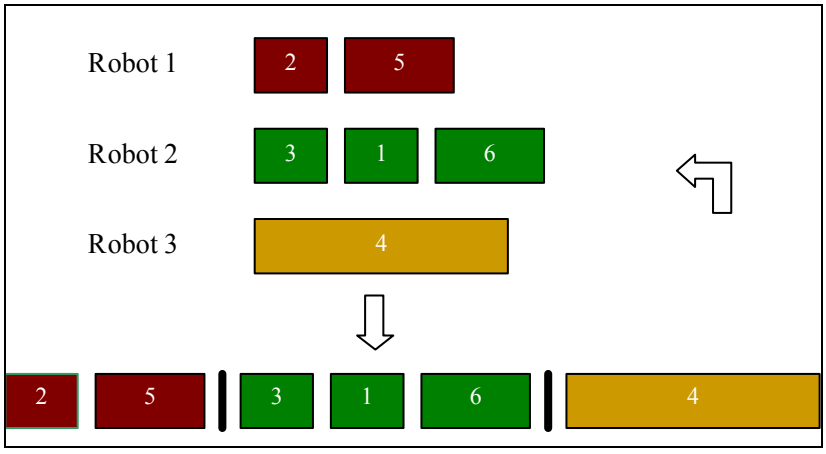

Figure 27 : Chromosome avec séparateurs

Le principal atout de cette représentation est sa simplicité. Un autre avantage de cette représentation est la simplicité de décodage car la fin d'une séquence d'exécution de missions est marquée par un séparateur. Mais d'après la littérature, l'utilisation des séparateurs conduit souvent à des chromosomes de mauvaise qualité.

Codage séquentiel

Dans cette représentation, l'ensemble L de symboles est simplement l'ensemble des missions à planifier. Le codage est très simple, c'est la séquence des missions dans l'ordre de leur date s_m de début et en cas d'égalité dans l'ordre alphabétique du robot l'exécutant. Ce codage est donc une représentation séquentielle dans le temps. La Figure 28 est un exemple de cette représentation que nous appelons codage séquentiel.

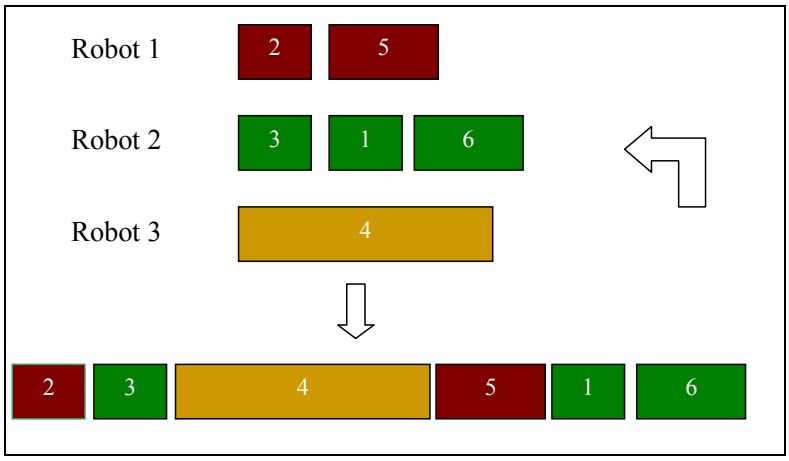

Figure 28 : Codage séquentiel

Le décodage d'un chromosome s'effectue par une heuristique gloutonne. Elle place les missions les unes après les autres dans l'ordre du chromosome. Chaque nouvelle mission est placée sur un robot équipé de module nécessaire qui permet de démarrer la mission au plus tôt tout en respectant les contraintes d'autonomie énergétique. L'inconvénient de cette représentation est que la procédure de décodage ne garantit pas la génération de la même solution initiale du chromosome codé.

Codage séquentiel avec missions fictives

Cette représentation est similaire au codage séquentiel mais permet de préserver l'identité de la solution obtenue par le décodage du chromosome d'une solution initiale. L'ensemble de symboles *L* est l'ensemble des missions à planifier, complété par un symbole "ϕ" représentant des missions fictives. La séquence des missions de chaque robot est complétée par des missions fictives de manière à avoir le même nombre de missions sur tous les robots. Le chromosome est obtenu en triant les missions dans l'ordre: d'abord les premières missions des différents robots dans l'ordre alphabétique des robots, ensuite les deuxièmes missions des différents robots, ainsi de suite. La Figure 29 est un exemple. Les séquences des différents robots sont {2,5,ϕ} pour le robot 1, {3,1,6} pour le robot 2 et {4, ϕ, ϕ} pour le robot 3. Le chromosome final est {2,3,4; 5,1, ϕ; ϕ,6, ϕ}.

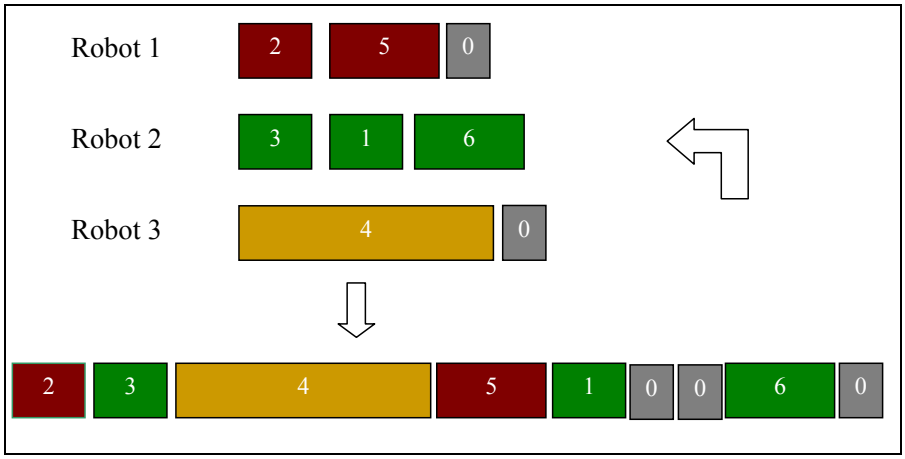

Figure 29 : Codage séquentiel avec missions fictives

Le décodage d'un chromosome de cette représentation est simple. Il suffit d'affecter les missions dans l'ordre du chromosome et chaque fois à un robot différent de manière cyclique. La solution finale est déduite facilement en ignorant les missions fictives.

Notons que, par construction, une solution obtenue par le décodage d'un chromosome séquentiel est nécessairement admissible. Par contre, les solutions obtenues par décodage des chromosomes parallèles avec délimiteurs ou séquentiels avec missions fictives ne sont pas nécessairement admissibles. Il faut pour cela vérifier si chaque mission est affectée à un robot correctement équipé et si les autonomies des robots sont respectées.

6.3. Opérateurs de sélection pour la reproduction

Nous distinguons deux familles d'opérateurs de sélection. Les opérateurs de sélection pour la reproduction sont utilisés pour déterminer les chromosomes parents à croiser pour reproduire la nouvelle génération. Les opérateurs de sélection pour le remplacement servent à éliminer les chromosomes pour former une nouvelle génération.

Parmi les opérateurs de sélection pour la production on distingue deux familles : les sélections proportionnelles et les sélections par tournois.

La roulette

La roulette (*Roulette Wheel Selection* RWS) est une méthode de sélection proportionnelle inspirée du jeu de casino. Cet opérateur a été proposé par Hollande pour les premiers

algorithmes génétiques. Cette méthode consiste à décomposer un disque en μ portions, où chaque portion représente un chromosome parent, en plus la taille de chaque portion est proportionnelle à la performance des individus. Si on choisi un disque dont la circonférence est de taille λ, la circonférence λ_i de la portion d'un individu i de fitness f_i ($i=1..\mu$) peut être donnée par la formule suivante :

$$\lambda_i = \frac{\lambda}{\sum_{i=1..\mu} f_i} f_i$$

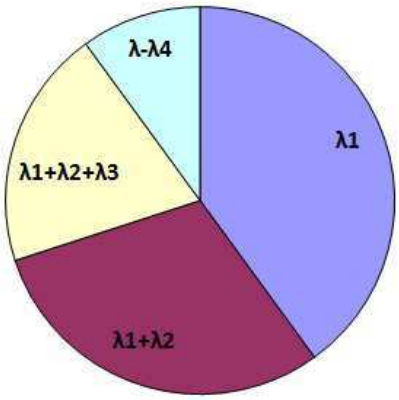

Figure 30 : Roulette

Ensuite, pour effectuer les sélections, on effectue λ tirages de valeurs aléatoires comprises entre 0 et λ. Pour chacun des tirages, la portion pointée est sélectionnée, et ainsi l'individu correspondant est sélectionné pour la reproduction.

Echantillonnage stochastique universel

La méthode RWS nécessite λ tirages, ce qui augmente la probabilité que certains individus soient sélectionnés plusieurs fois au détriment d'autres individus de la population. Pour remédier à cet effet de la roulette, Baker et al. [Baker 87] ont proposé la méthode d'échantillonnage stochastique universel (*Stochastic Universal Sampling SUS*) qui nécessite un seul tirage. Avant de lancer le tirage, la roulette est pointée par λ pointeurs qui sont équidistants les uns des autres, où la distance entre deux pointeurs successifs est de valeur unitaire si la circonférence de la roulette est de valeur λ. Une seule valeur aléatoire λ_0 comprise entre 0 et λ est générée. Cette valeur de λ_0 déterminera la l'angle avec lequel le système de pointeurs pivotera. Après la rotation des pointeurs les λ portions pointées seront sélectionnés et les λ parents correspondants seront sélectionnés pour la reproduction.

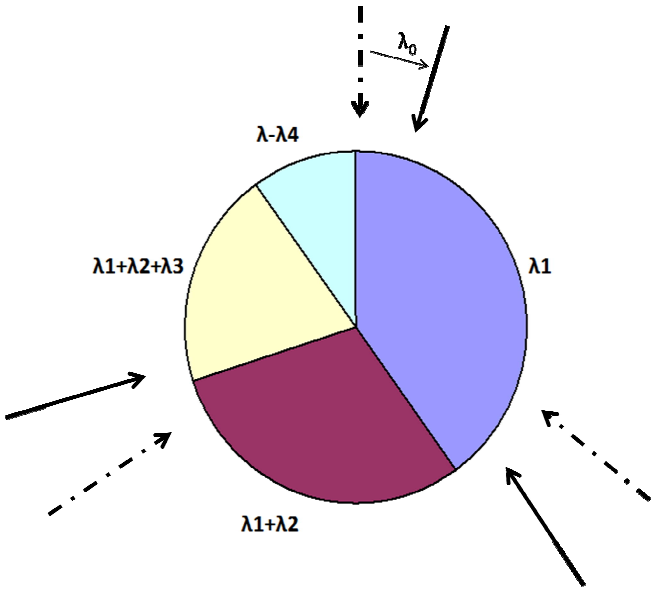

Figure 31 : SUS

Sélection de Boltzmann

Les méthodes RWS et SUS se basent sur les performances de chaque individu pour déterminer la taille de la portion associée, ce qui peut conduire à une convergence prématurée. De la Maza et al. proposent de remplacer les critères de performance f_i utilisés dans les opérateurs de sélection proportionnelle par des critères de performance ajustés selon l'évolution de l'algorithme génétique [de la Maza 93]. Ils proposent un critère de performance ajusté f_i' inspiré par la distribution dite "soft max" de Boltzmann.

$$f_i' = \exp(\frac{f_i}{T})$$

Où T représente la « Température » qui décroit graduellement, passant ainsi d'une sélection équiprobable à une sélection du meilleur individu (avec le plus grand fitness f_i).

Sélection selon le rang des individus

Une autre variante des méthodes de sélection proportionnelle consiste à ranger les individus par ordre croissant de performance, et la fonction de performance ajustée est donnée par la formule suivante :

$$f_i' = \left(1 - \frac{r_i}{\mu}\right)^p$$

où r_i est le rang du chromosome parent i, μ est le nombre d'individus de la population parent et p désigne un paramètre de pression qui détermine la convergence de la population vers les individus ayant les meilleurs performances.

L'avantage de cette méthode est qu'elle ne nécessite pas de calculer la fonction objectif de chaque individu, il suffit de les classer selon des règles prédéfinies, ou bien de les comparer.

Sélection par tournois

Une autre famille des méthodes de sélection, populaire pour sa simplicité d'implémentation, se base sur le principe de tournois. Cette méthode consiste à (i) sélectionner aléatoirement *k* chromosomes dans la population des parents, (ii) sélectionner le meilleur des *k* chromosomes pour la reproduction, et (iii) répéter ce processus jusqu'à la sélection de λ individus pour la reproduction. La sélection par tournois peut être avec remise ou sans remise. Dans la sélection sans remise, un individu sélectionné pour la reproduction est retiré de la population des parents et donc ne peut plus être sélectionné. Dans la sélection avec remise, les individus sélectionnés peuvent toujours être sélectionnés plus tard.

Les tournois stochastiques sont une évolution de ce système. On choisit chaque fois deux chromosomes et on retient aléatoirement l'un des deux individus avec une probabilité *p* plus grande ($0.5 < p < 1$) de sélectionner le meilleur des deux.

6.4. Opérateurs de sélection pour le remplacement

Cet opérateur a pour but de remplacer les μ chromosomes parents de la génération G par un autre ensemble de μ chromosomes enfants. La génération G+1 est constituée d'une part des parents de la génération G et d'autre part des chromosomes enfants générés à la génération G, soit issues de l'ensemble de π individus générés par des heuristiques à l'issue de la boucle générationnelle de l'algorithme génétique.

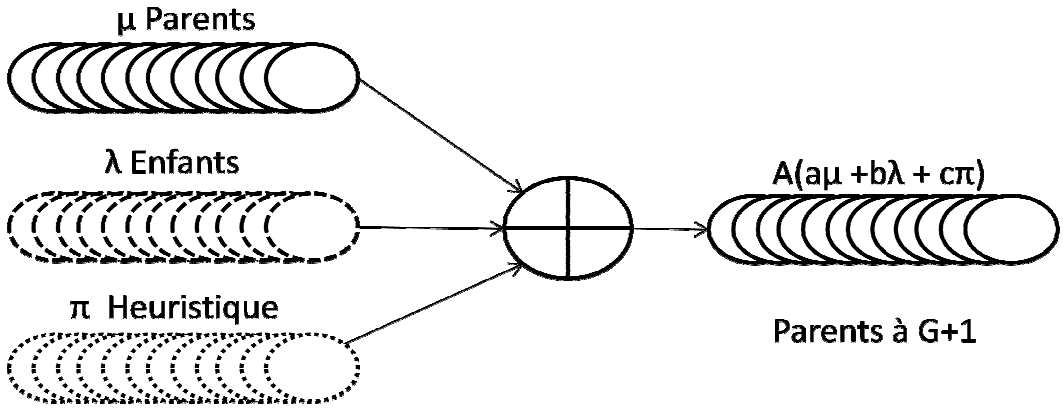

Figure 32 : Sélection pour le remplacement

Remplacement générationnel

Cet opérateur a été utilisé par Holland dans les premiers algorithmes génétiques. Cet opérateur écarte la totalité de la génération des parents, cette dernière est remplacée par la génération des enfants ; en limitant pour cela le nombre λ des chromosomes enfants générés à la taille de la population choisie.

Remplacement déterministe

Pour cette sélection, la population des enfants générés est plus grande que la population des parents ($\lambda > \mu$). Pour cela, les μ meilleurs enfants remplaceront la génération des parents pour former la génération G+1.

Remplacement stationnaire (Steady State)

Cette méthode est utilisée dans le cas où la génération des enfants n'est pas nécessairement meilleure que la génération précédente. Dans cette méthode, la majorité des parents de la

génération *G* est conservée après la reproduction et seulement une minorité des enfants est injectée à la population suivante G+1, ce qui permettra d'atteindre une convergence plus lente sans risquer de perdre la spécificité et la diversité de la génération des parents.

Remplacement élitiste
Dans cet opérateur, la nouvelle génération G+1 est constituée de meilleurs chromosomes parmi l'ensemble des chromosomes enfants et des chromosomes parents. Ainsi, le meilleur individu rencontré dans le passé reste dans la population jusqu'à la détection d'autres solutions strictement meilleures.

Remplacement hybride
Dans cette sélection, la nouvelle génération G+1 est constituée de trois parties : une partie de la population des parents à la génération G, une portion de la population enfants, et des solutions générées par des heuristiques pour compléter la population.

6.5. Opérateurs de Croisements de chromosomes

Les opérateurs de variation consistent à engendrer de nouveaux chromosomes à partir d'un ou de plusieurs chromosomes. L'opérateur de croisement est une variante des opérateurs de variation qui consiste à appliquer une série de transformation à deux chromosomes qu'on appellera chromosomes parents pour engendrer un ou deux chromosomes enfants ayant un mélange des caractéristiques des deux parents.

Ces mécanismes de croisement ont pour but de prendre un sous ensemble de gènes d'un chromosome père et de construire le reste des gènes avec le deuxième chromosome parent de façon à ce que le nouveau chromosome représente une solution réalisable dans notre cas. Le nouveau chromosome devrait contenir toutes les missions et chaque mission peut apparaître une fois et une seule fois dans le chromosome. Différentes techniques de croisement ont été mises en œuvre et la comparaison entre ces techniques peut être trouvée dans les sections suivantes.
Dans notre étude, le croisement dépend du codage utilisé. Pour chaque codage, chaque chromosome est transformé en une permutation d'un ensemble de symboles différents et les opérateurs de croisement pour les problèmes de voyageurs de commerce comme LOX et PMX sont utilisés pour générer de nouvelles permutations et donc de nouvelles solutions.

Pour le codage parallèle avec séparateurs, nous commençons par transformer un chromosome en une permutation d'un ensemble de symboles différents pour différencier les séparateurs. Ainsi le chromosome {2, 5, **f**, 3, 1, 6, **f**, 4, **f**} de la Figure 27 devient {2, 5, $\mathbf{f_1}$, 3, 1, 6, $\mathbf{f_2}$, 4, $\mathbf{f_3}$}.

Algorithme : croisement des chromosomes parallèles avec séparateurs
1. Sélectionner deux parents pour la reproduction;
2. Transformer chaque chromosome en différenciant les séparateurs associés à chaque robot;
3. Appliquer un opérateur de croisement pour générer deux nouvelles permutations;
4. Remplacer les symboles "$\mathbf{f_i}$" par "**f**" pour obtenir deux nouveaux chromosomes;
5. Décoder les nouveaux chromosomes et vérifier l'admissibilité des solutions correspondantes;
6. Ajouter les chromosomes admissibles dans l'ensemble d'enfants.

Algorithme 2 : croisement de chromosomes parallèles avec séparateurs

Pour le codage séquentiel, chaque chromosome est déjà une permutation des symboles différents correspondants à différentes missions. Le croisement est donc plus aisé.

Algorithme : croisement des chromosomes séquentiels

1. Sélectionner deux parents pour la reproduction;
2. Appliquer un opérateur de croisement pour générer deux nouvelles permutations;
3. Décoder les nouveaux chromosomes et retirer ceux n'ayant pas de solution décodée;
4. Ajouter les nouveaux chromosomes restant dans l'ensemble d'enfants.

Algorithme 3 : croisement de chromosomes séquentiels

Pour le codage séquentiel avec missions fictives, les différents chromosomes n'ont pas nécessairement le même nombre de symboles. Le nombre de symboles dans chaque chromosome est multiple de M où M est le nombre de robots. Nous devons ainsi ajouter d'autres missions fictives afin d'avoir des chromosomes de la même longueur. Pour le chromosome {2,3,4; 5,1, ϕ; ϕ,6, ϕ} de la Figure 29, on peut ajouter trois missions fictives, chacune à un robot, pour obtenir un chromosome de longueur 9, i.e. {2,3,4; 5,1, ϕ; ϕ,6,ϕ; ϕ,ϕ,ϕ}. En différentiant les notations des missions fictives dans l'ordre de leur apparition, nous obtenons des permutations d'un même ensemble de symboles. Le chromosome de la Figure 29 devient alors {2,3,4; 5,1, ϕ_1; ϕ_2,6,ϕ_3; ϕ_4,ϕ_4,ϕ_6}.

Algorithme : croisement des chromosomes séquentiels avec missions fictives

1. Sélectionner deux chromosomes parents pour la reproduction;
2. Ajouter des missions fictives au chromosome le plus court pour obtenir deux chromosomes de la même longueur;
3. Transformer les symboles "ϕ" en "ϕ_i" en les numérotant dans l'ordre de leur apparition;
4. Appliquer un opérateur de croisement pour générer deux nouvelles permutations;
5. Remplacer les symboles "ϕ_i" par "ϕ" pour obtenir deux nouveaux chromosomes;
6. Décoder les nouveaux chromosomes en affecter les missions (fictives ou non) aux robots de manière cyclique et vérifier l'admissibilité des solutions correspondantes;
7. Coder en chromosome les solutions admissibles pour supprimer les missions fictives inutiles et pour normaliser les représentations avec les missions fictives à la fin de chaque séquence d'un robot;
8. Ajouter les chromosomes obtenus dans l'ensemble d'enfants.

Algorithme 4 : croisement de chromosomes séquentiels avec missions fictives

Dans la suite, nous présentons les opérateurs que nous utilisons pour le croisement des deux permutations d'un même ensemble de symboles.

Croisement LOX

Le *linéaire Ordre Crossover* (LOX) est l'une des techniques de croisement classiques pour les problèmes de tournées de véhicules. Elle a été introduite par Davis pour le croisement des deux permutations d'un même ensemble de symboles. [Davis 85]

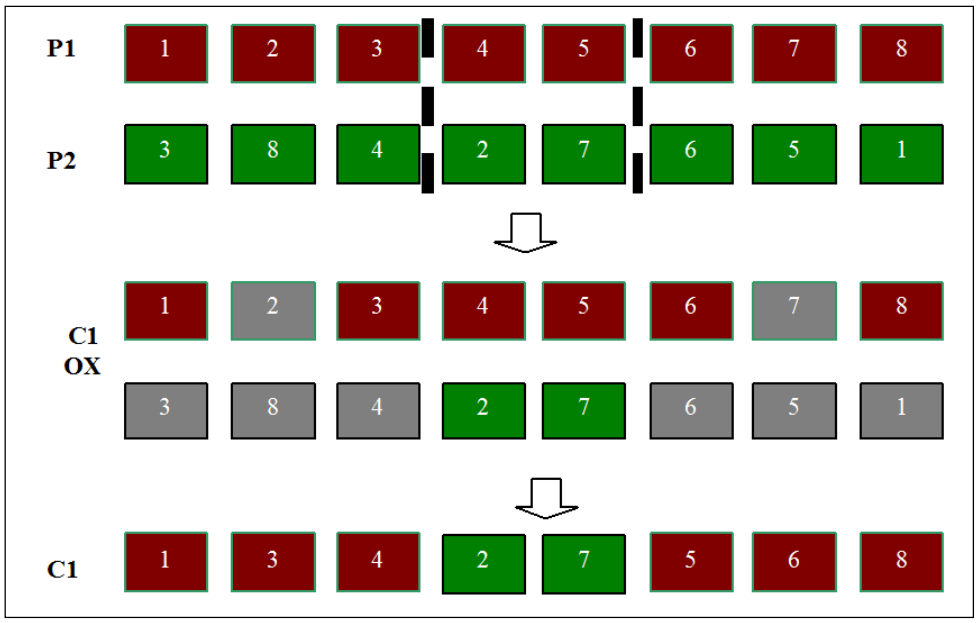

Figure 33 : Croisement LOX

Cette technique illustrée par Figure 33 commence par choisir aléatoirement deux points de coupe. Les symboles figurant de la partie centrale du deuxième parent sont supprimés dans le premier parent. Un nouveau chromosome enfant est obtenu en insérant la partie centrale du parent 2 dans le reste du parent 1. En inversant les deux parents, un autre chromosome enfant peut être généré de manière similaire.

Croisement PMX

La technique de croisement PMX (*Partially Mapped Crossover*) illustrée à la Figure 34, a été présentée par Goldberg et Lingle [Goldberg 85].

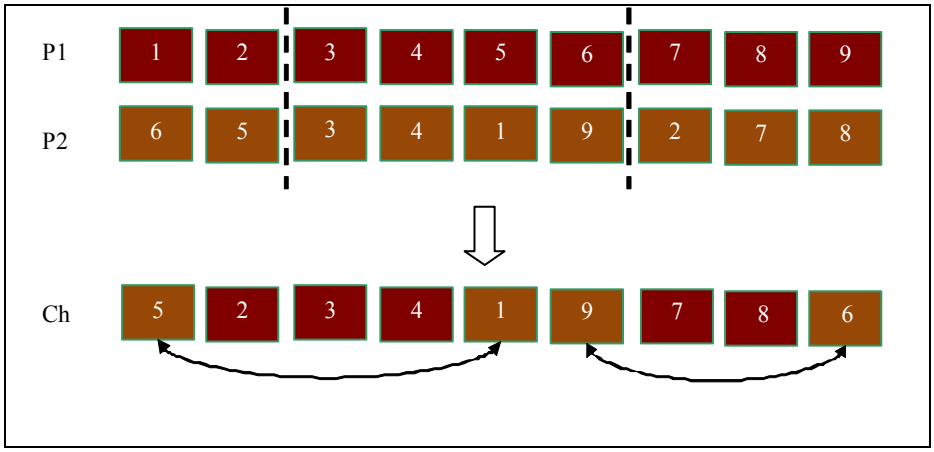

Figure 34 : Croisement PMX

Ce croisement est également défini par deux points de coupe choisis aléatoirement. La partie centrale du deuxième parent est utilisée dans le nouveau chromosome enfant. Le reste du nouveau chromosome vient du chromosome du parent 1 en remplaçant chaque symbole figurant dans la partie centrale du parent 2 par le symbole de la même position du parent 1. Ainsi, dans la Figure 34, le symbole "1" du parent 1 figurant dans la partie centrale du parent 2 à la cinquième position est remplacé par le symbole "5" à la cinquième position du parent 1. De même, en inversant les deux parents, un autre chromosome enfant peut être obtenu.

Croisement Cyclique

Le croisement cyclique [Oliver 87], illustré par la Figure 35, a été proposé par Oliver et al. . Le principe de base de ce croisement est de trouver une série de sous-séquences de gènes, qui seront permutés entre les deux chromosomes parents de façon à ce que les chromosomes enfants ne présentent pas de gènes doublons ni de gènes absents.

Pour cela on détermine un vecteur référence, le vecteur binaire permettant d'identifier les gènes à échanger entre les deux chromosomes.

Afin de déterminer ce vecteur référence, on choisit d'abord aléatoirement un point de départ qui indique un gène quelconque du premier chromosome parent. Ensuite on cherche dans le premier chromosome parent le gène ayant la même valeur que le gène du second parent correspondant au gène de départ. On désigne par gène correspondant, le gène de l'autre chromosome parent, ayant la même position que le gène en question. Chaque fois que l'on effectue un échange, le vecteur référence vaut 1 au gène correspondant. Et le nouveau gène avec qui on vient de faire l'échange est fixé comme le nouveau point de départ. Ce processus est répété jusqu'à retrouver le point de départ.

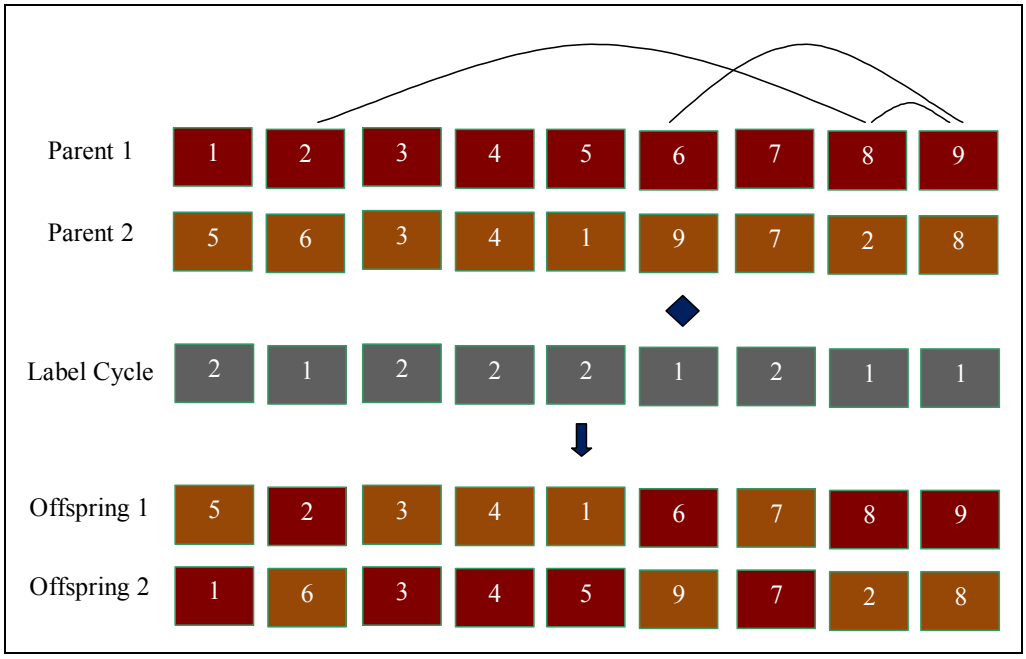

Figure 35 : Croisement cyclique

Par exemple, dans la Figure 35, le point de départ est le 6ème élément ayant la valeur 6 dans le premier chromosome parent et le gène correspondant du second chromosome ait la valeur 9. Donc on cherche le gène du premier chromosome qui a la valeur 9. Dans notre exemple, c'est le 9ème gène et on effectue une permutation entre les $6^{èmes}$ gènes et les $9^{èmes}$ gènes En suivant le même principe on effectue un échange entre les $9^{èmes}$ et les $8^{èmes}$, puis entre les $8^{èmes}$ et les $2^{èmes}$ où on trouve le gène correspondant ayant la valeur de départ 6 ce qui marque le fin du processus d'échange.

Deux chromosomes enfants sont ensuite construits à l'aide du vecteur de référence. Le premier chromosome est obtenu en choisissant pour chaque position le gène du parent 1 (resp. 2) si le vecteur de référence vaut 1. Le deuxième chromosome est obtenu par le choix inverse.

Croisement intelligent

Cette famille de croisements combine les méthodes de croisement précédentes. La première méthode de combinaisons, appelée croisement aléatoire, choisit au hasard à chaque génération un des opérateurs de croisement. Cette opération s'est avérée efficace pour certaines instances et donne des solutions plus performantes.

Une amélioration de cette combinaison aléatoire consiste à attribuer à chaque opérateur de croisement un indicateur de performance et à choisir les opérateurs de croisement avec des probabilités proportionnelles à leur indicateur de performance. Ainsi, plus l'indicateur de performance d'un croisement est élevé, plus il a de chance d'être choisi à la prochaine génération. A la fin de chaque itération, l'indicateur de performance de l'opérateur choisi est ajusté en fonction des résultats de la nouvelle génération par rapport à l'ancienne génération. L'indicateur de performance est borné dans un intervalle donné afin de permettre à tous les opérateurs d'être choisis.

6.6. Opérateur de Mutations

L'opérateur de mutation fait partie des opérateurs de variation, comme l'opérateur de croissement. Ces opérateurs effectuent des modifications sur les chromosomes et donne naissance à de nouveaux chromosomes. L'opérateur de mutation effectue des variations sur un seul chromosome et nécessite la connaissance d'un seul chromosome pour effectuer ces variations. Les variations classiques comme *bit flop* pour les chromosomes binaires et la mutation *2-opt* avec les représentations ordinales du problème de voyageurs de commerce ne semblent pas pertinentes pour notre représentation. Nous proposons les mutations suivantes.

Mutation aléatoire

La mutation aléatoire effectue un échange entre deux gènes choisis aléatoirement du chromosome considéré. Après la mutation, le chromosome est décodé et l'admissibilité vérifiée avant l'introduction dans la population des chromosomes enfants.

Optimisation locale

Une autre variante de cet opérateur consiste à décoder le chromosome et ensuite effectuer une optimisation locale de la solution obtenue. Cette optimisation locale peut s'avérer gourmande en temps de calcul ce qui affecte le temps de calcul d'une boucle génératrice.

Dans notre étude, l'optimisation locale tente d'améliorer le planning d'un robot en appliquant une combinaison des opérations suivantes :

- Déplacer une mission sélectionnée aléatoirement à une position choisie aléatoirement,
- Echanger les positions des deux missions sélectionnées aléatoirement.

Une amélioration serait d'appliquer les heuristiques proposées dans le chapitre précédent afin d'améliorer le planning de chaque robot, dans le cas où seulement quelques missions existent dans le planning du robot (moins de quatre missions), une énumération de toutes les combinaisons possibles peut être envisageable.

6.7. Comparaisons et Résultats Numériques.

6.7.1. Performances des opérateurs de croisement

Nous considérons une instance composée de 7 robots, 49 missions et 4 modules de fonctionnalité. L'algorithme génétique est limité à 1000 générations et à 5 min de temps de calcul.

Quatre approches de croisement sont comparées (LOX, PMX, Cyclique, et le croisement aléatoire). La Figure 36 montre l'évolution de la meilleure solution de la population d'une génération à une autre.

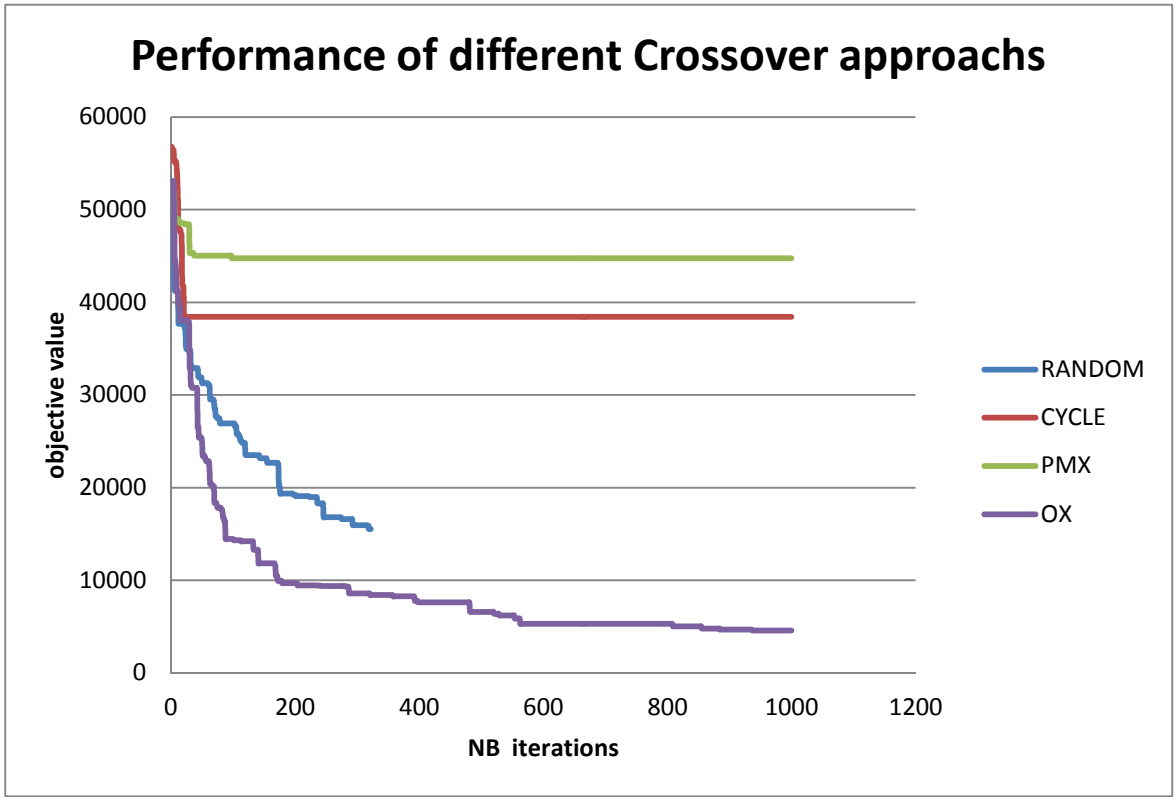

Figure 36 : **Performance des opérateurs de croisement**

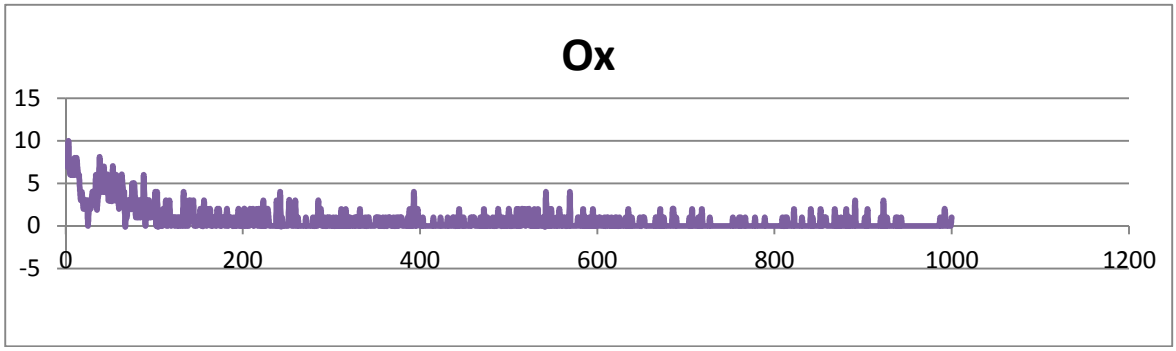

Figure 37 : Nombre de chromosome remplacés par croisement LOX

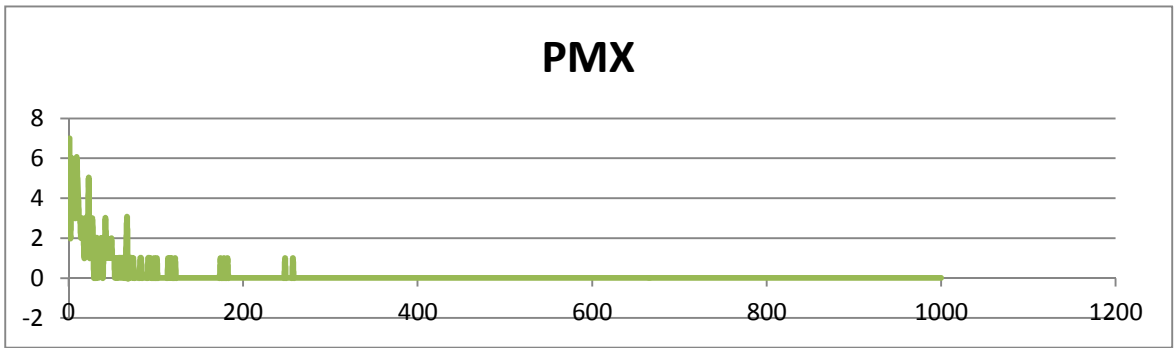

Figure 38 : Nombre de chromosome remplacés par croisement PMX

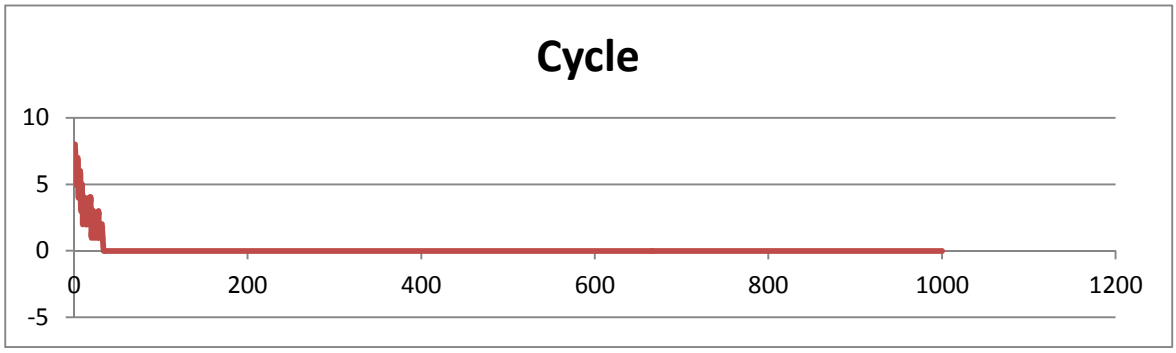

Figure 39 : Nombre de chromosome remplacés par croisement cyclique

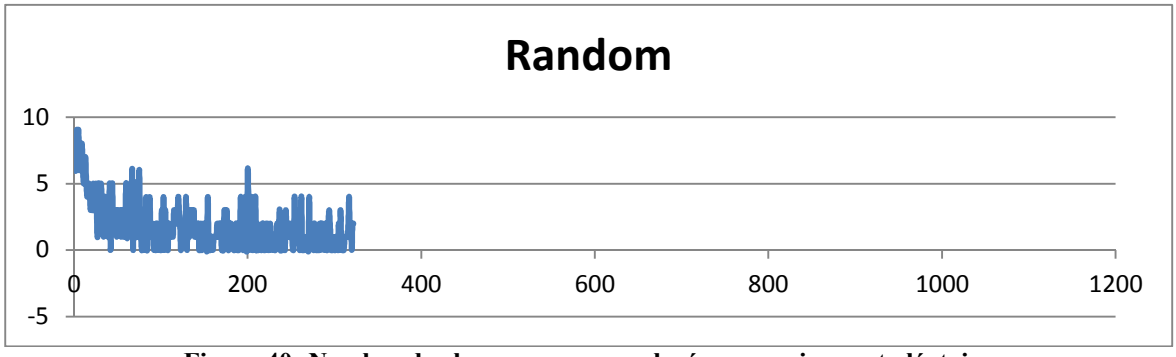

Figure 40: Nombre de chromosome remplacés par croisement aléatoire

La comparaison, illustrée par la Figure 36, entre les performances des différentes techniques de croisement appliquées sur la même instance, montre clairement la prépondérance de l'opérateur de croissement LOX par rapport aux autres opérateurs de croissement. Les graphes (Figure 37 ;Figure 38 ; Figure 39 ; Figure 40) présentent le nombre des descendants générés à chaque itération et qui sont plus performants que certains individus parents.

6.7.2. Performances des opérateurs de mutation optimisés

L'instance à examiner dans cette section se compose de 6 robots, 78 missions et 3 modules fonctionnalité. Cet algorithme à été limité à 1000 itérations au maximum, soit à un temps de calcul de 5 minutes. Les quatre approches de croisement présentées à la section précédente (LOX, PMX, Cyclique, et le croisement aléatoire qui est un mélange des trois précédents) sont testées une fois avec amélioration locale et une fois avec mutation génétique simple.
Le tableau suivant illustre les performances de chaque approche avec la performance de la meilleure solution trouvée pour notre problème de minimisation de coût /retards, le nombre des descendants améliorés et le critère d'arrêt qui a été abouti en premier

Approche	Meilleur Solution	# Iterations	CPU	Nombre Descendants.
LOX	26424	1000	47	1029
LOX + opt	6314	47	304	214
PMX	124392	1000	52	211
PMX+opt	5582	297	300	356
Cycle	148407	1000	53	240
Cycle + opt	8316	398	300	201
Mix	69475	1000	47	1035
Mix + opt	5396	85	303	355

Tableau 15 : Performances des différents opérateurs

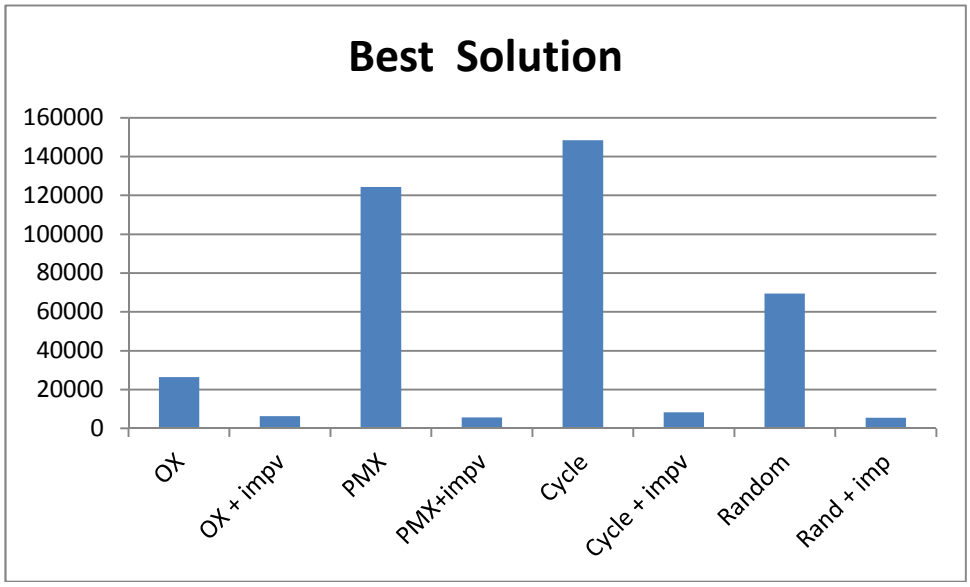

Figure 41 : Meilleure solution par approche

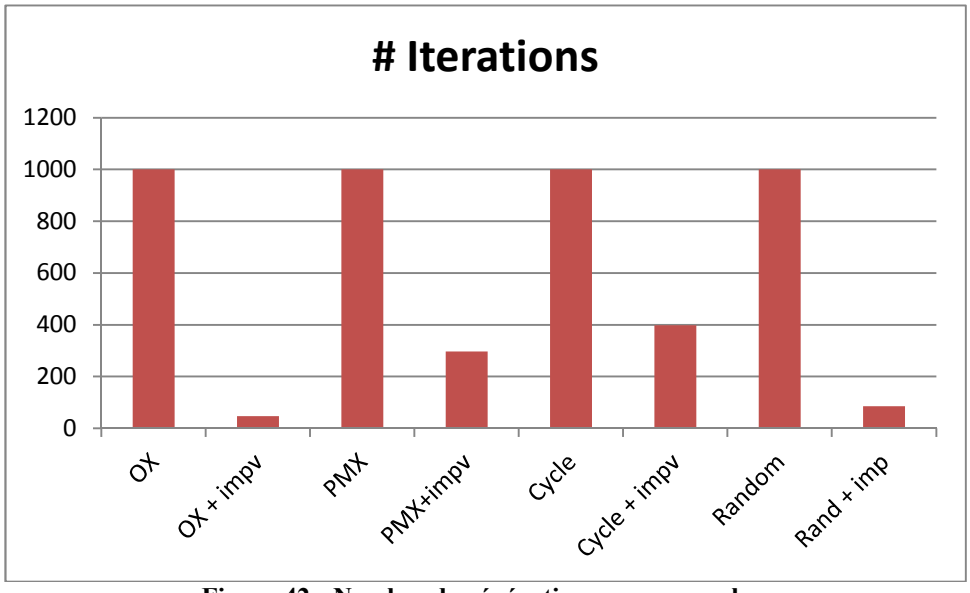

Figure 42 : Nombre de générations par approches

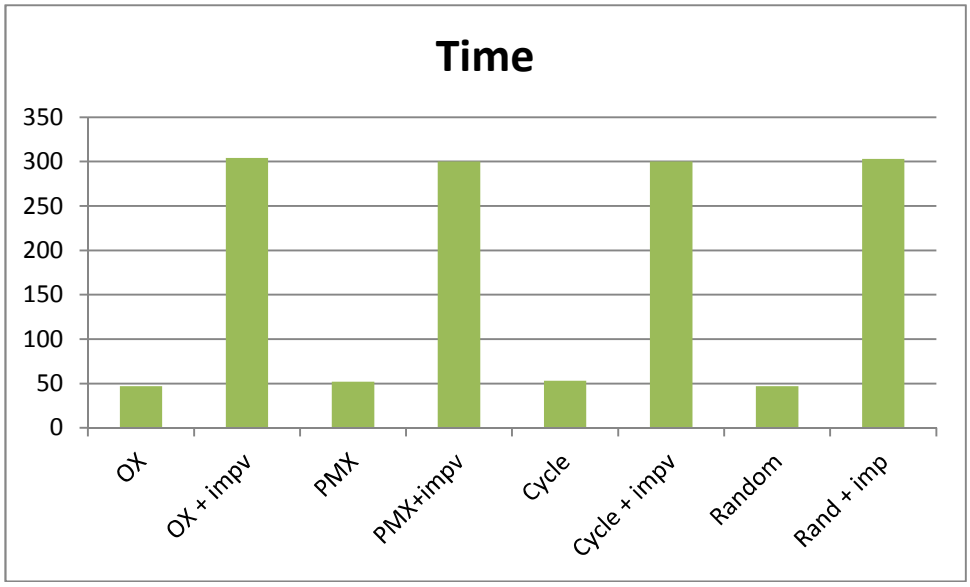

Figure 43 : Temps de calcul par approche

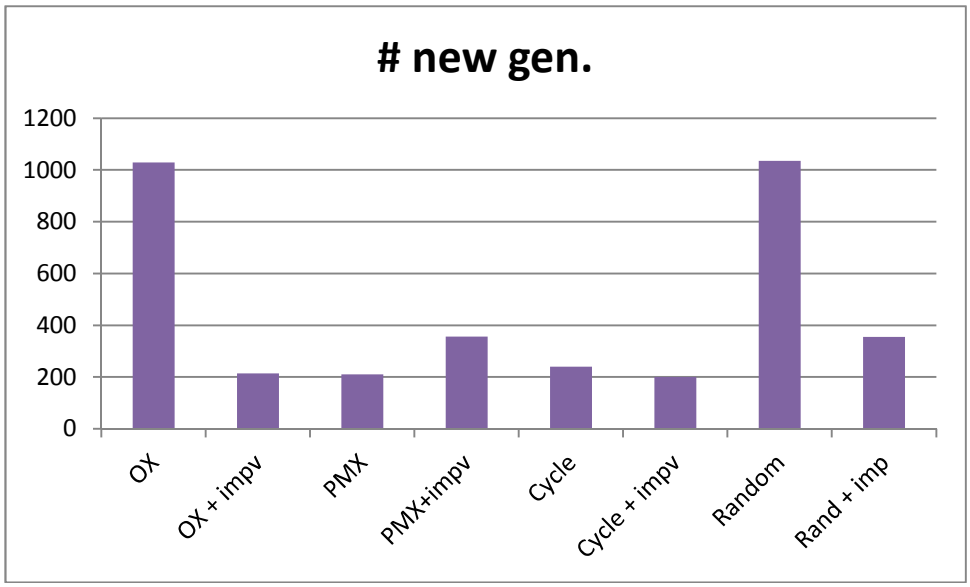

Figure 44 : Nombre total de descendants retenus pour les futures générations.

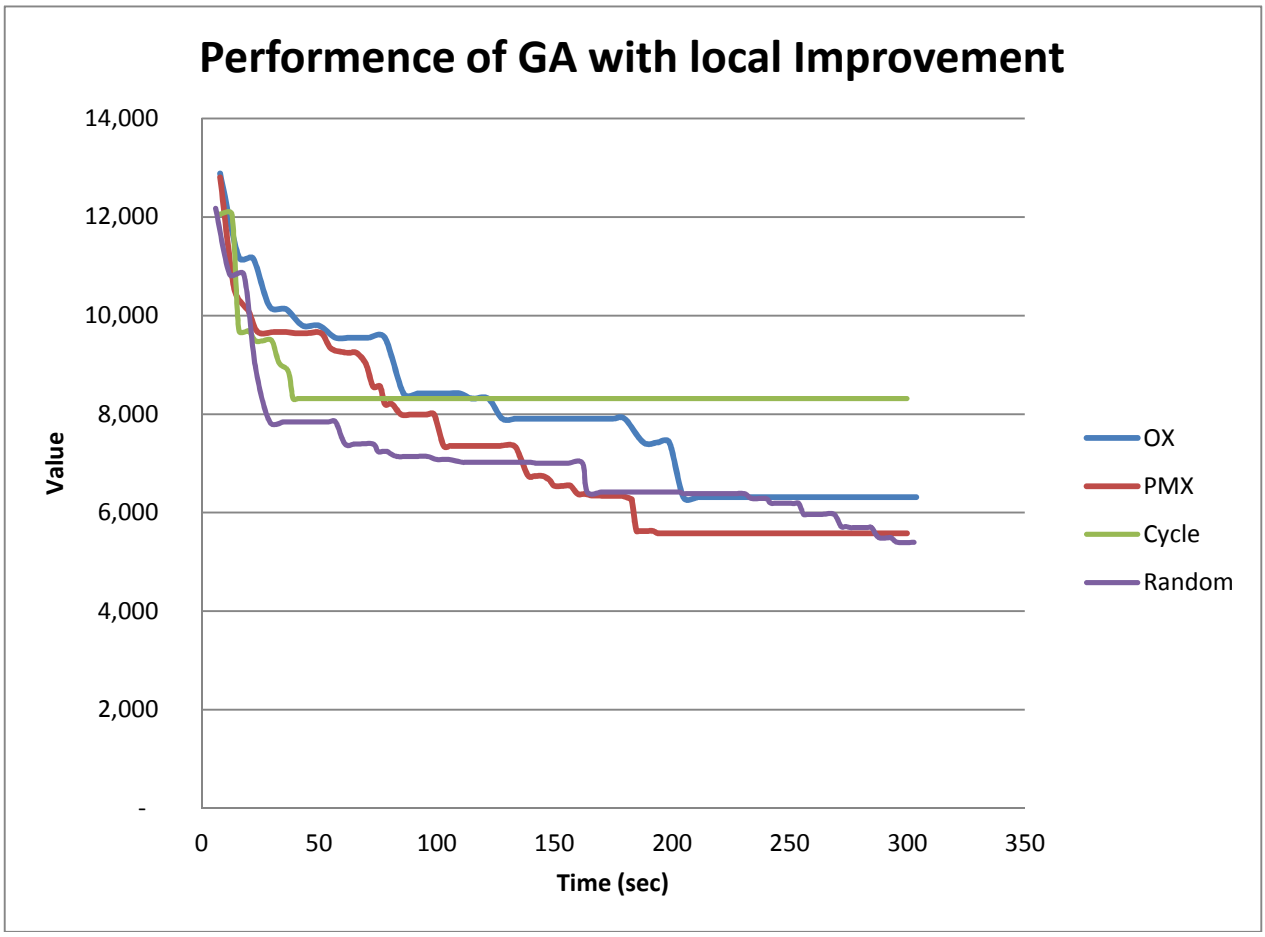

Figure 45 : Convergence des approches de croisements avec optimisation

D'après les résultats ci-dessus, nous pouvons déduire que les algorithmes génétiques couplés avec une optimisation locale offrent de meilleures solutions, mais on note que le processus d'optimisation est gourmand en puissance de calcul. La Figure 41 montre clairement que les instances résolues avec une optimisation locale surpassent les techniques sans optimisation locale par un facteur d'au moins 1 à 5, mais l'inconvénient majeur de cette approche est sa gourmandise en puissance de traitement qui multiplie par six le temps de calcul d'une boucle générationnelle. La Figure 45 présente une comparaison entre les quatre approches de croisements couplés par une optimisation locale, mais de façon surprenante : le croisement aléatoire PMX surpasse l'opérateur de croisement LOX en contradiction avec les résultats de la section précédente. Ces résultats nous ont poussés à mettre en place une technique de croisement intelligent qui est une évolution du croisement aléatoire, qui s'adapte avec l'instance étudiée en fonction des performances de chaque approche.

6.8. Conclusions

Dans ce chapitre on a proposé une implémentation centralisée de l'agent de décision responsable de l'affectation et du planning des missions qui peut être considéré comme une variante de deux problèmes difficiles : le VRP-TW et le MC-VRP. L'implémentation proposée se base sur des algorithmes évolutionnaires couplés par des méthodes d'optimisation locale. Plusieurs opérateurs ont été testés pour identifier les opérateurs les mieux adaptés à notre problème notamment les opérateurs de croisement, les opérateurs de mutation et les opérateurs de sélection. Des approches inspirées des techniques d'apprentissage qui

s’adaptent avec l’évolution des individus et des générations ont été mises en place. Un système de sélection hybride qui permet d’enrichir les générations par des individus générés par des heuristiques lancées en amont avec les algorithmes évolutionnaires a été également mis en place.

Le problème de pré-planification des missions récurrentes est un des problèmes identifiés au début du projet. Il nous semble prometteur de recourir à une résolution par un agent centralisé semblable a celui présenté dans ce chapitre. Ce problème consiste à affecter un semi-planning prédéfini à un système de N robots ayant des configurations différentes et des recommandations prédéfinies de consommation d'énergie. Les missions sont divisées en deux catégories : (i) les missions de routine qui sont régulières et connues à l'avance, comme le nettoyage régulier ou le transport régulier des médicaments (on sait à l'avance le calendrier de cette catégorie de missions). (ii) Les missions inattendues, où seule une estimation approximative de l’ensemble des missions peut être prévue. Le but de ce problème serait de pré-planifier les missions de routine et de réserver des plages sur chaque robot pour les missions inattendues.