République Algérienne démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique
Université Mentouri – Constantine

Faculté des sciences de l'Ingénieur
Département d'Informatique
Laboratoire Lire

Mémoire

Présenté pour l'obtention du diplôme de Magister en Informatique
Option : Information & Computation

Thème :

# Résolution du problème de l'emploi du temps : Proposition d'un algorithme évolutionnaire multi objectif

Présenté par :

Mme Troudi Fatiha

Dirigé par :

Dr. M.K. Kholladi

Devant le jury :

| | | |
|---|---|---|
| Pr. M. Boufaida | Professeur, Université de Constantine | Président |
| Dr. M.K Kholladi | Maître de conférences, Université de Constantine | Rapporteur |
| Dr. S. Meshoul | Maître de conférences, Université de Constantine | Examinateur |
| Pr. M. Baatouche | Professeur, Université de Constantine | Examinateur |

Année universitaire 2005-2006

# TABLE DES MATIERES

## Chapitre III L'optimisation Multiobjectif

# Introduction générale

Dans de nombreux domaines de la vie professionnelle, on se trouve confronter au problème de la planification d´horaire de travail. Dans les usines, des pièces doivent être cheminées à travers plusieurs machines, la gestion de ce trafic doit répondre à certaines espérances, tel que la maximisation de la production ou l´exploitation maximale des machines. Dans les hôpitaux , un certain nombre de personnels constitués d´infirmiers et de medecins doivent être attribués aux postes de travail de manière à obéir à certaines règles de gestion des hôpitaux. Donc cela nécessite souvent une élaboration périodique et stratégique de planning de travail de personnel et les raisons de cette élaboration sont multiples. Elles vont de la réduction des coûts de productivité à l´amélioration de la qualité de service aux clients, en passant par l´assurance d´une bonne qualité de vie aux employés.

Fortement combinatoire, les problèmes de gestion du temps, pour les résoudre, revient à résoudre un puzzle complexe, chaque organisation ( compagnies aériennes, entreprises de production, hôpitaux, universités,…etc ) possède ses propres normes et critères.

La planification du temps consiste à allouer des ressources données à des objets dans un intervalle de temps, de façon à satisfaire au mieux un ensemble d´objectifs tels que l´amélioration de la qualité de service et l´amélioration des conditions de travail.

Parmi la vaste famille des problèmes de planification d´horaire, on trouve celui de la confection d´emploi du temps dans les établissements éducatifs, notamment dans les universités qui consomment de nombreuses ressources humaines et donc financières. Ce problème est très important. En effet un mauvais emploi du temps influe directement et négativement sur le niveau de l´acquisition des étudiants.

Le problème de l´emploi du temps est un problème ardu dont la réalisation à la main est une tâche draconienne qui peut mobiliser plusieurs personnes plusieurs jours de travail. Sans oublier, que toute modification des données du problème peut complètement remettre en cause la solution trouvée.

Ces difficultés, ont induit l'idée d'assister par ordinateur l'élaboration des emplois du temps en adoptant des outils basés sur des algorithmes d'optimisation

robustes permettant de faciliter cette tâche. Ainsi depuis la fin des années cinquante, plusieurs contributions relatives au problème de l'emploi du temps sont apparus ; cela est dû, essentiellement, à sa complexité, sa grande taille et surtout au nombre important de problèmes pratiques que ce dernier regroupe.

D'une manière générale, le problème de l'emploi du temps consiste à définir un certain nombre d'affectations qui permettent d'assigner plusieurs ressources(humaines, matérielles,…etc) sur une période de temps, tout en respectant les contraintes imposées par les entités citées (disponibilité des ressources humaines, matérielles,…etc).

Le problèmes de l'emploi du temps s'avèrent être NP difficile et la taille de leurs instances se caractérisent souvent par leur très grande taille.

Les contraintes considérées peuvent différer d'un problème à un autre suivant la spécificité ainsi que les caractéristiques attendues de l'emploi du temps recherché. Les contraintes sont souvent classées en deux catégories, la première regroupe les contraintes dures ( un emploi du temps qui ne satisfait pas ce genre de contraintes est infaisable ou inacceptable), la seconde catégorie regroupe des contraintes (appelées souvent contraintes molles, souples ou de préférence) dont la satisfaction a différent degrés d'importances mais dont le non respect n'empêche pas une application plus au moins acceptable de l'emploi du temps trouvé. Typiquement ces contraintes (de préférence) sont utilisées pour exprimer ce que doit être un « bon » emploi du temps. Ces contraintes sont plus difficiles à formaliser que les contraintes dures et leur traitement est plus délicat. Ainsi la majorité des approches existantes relaxent les contraintes de préférence et les introduisent comme une fonction objectif dont l'optimisation permet de se rapprocher le plus possible de la satisfaction des contraintes.

De ce fait, ce sont principalement les modèles mono objectif qui sont utilisés lors de la résolution de ce type de problème. Notre étude s'est concentrée sur le fait que le problème de l'emploi du temps est lui aussi un problème d'optimisation multiobjectifs. L'étude qui a été entreprise dans ce travail s'intéresse aux différentes techniques d'optimisation multiobjectfs envisageables pour la résolution du problème de l'emploi du temps.

Ce mémoire est organisé en quatre chapitres. Dans le premier, nous introduisons les notions liées aux problèmes de planification d'horaires de travail et les différents types de plannings dans différents domaines de travail avec un aperçu des méthodes utilisées pour la réalisation de ces plannings.

Dans le deuxième chapitre, nous introduisons les notions liées aux problèmes d'optimisation combinatoire. Nous donnons les différentes raisons de

leurs difficultés, les méthodes de modélisation et de résolution proposées telles que les techniques de recherche Branch & Bound, la programmation dynamique, la programmation linéaire, l'algorithme A*. Ces méthodes se présentent généralement comme des heuristiques de recherche arborescente où il s'agit de trouver la solution par amélioration ou construction itérative. Ces méthodes se distinguent aussi par le caractère exact de la solution. Vu q'un problème d'optimisation ne se limite pas à trouver une solution qui maximise ou minimise une fonction objectif car la solution doit aussi satisfaire un ensemble de contraintes, nous présentons alors un état de l'art sur les approches proposées pour la résolution des problèmes de satisfaction des contraintes. A la fin de ce chapitre, une étude approfondie est consacrées aux méthodes approchées dites métaheuristiques proposées dans la littérature, nous mettons l'accent sur leur mécanismes de diversification et d'intensification ainsi que leurs application pour la résolution du problème d'emploi du temps. Les différentes stratégies d'hybridation de ces techniques sont aussi présentées.

Dans le troisième chapitre, nous introduisons les notions liées à l'optimisation multiobjectif. Nous mettons l'accent sur la nature des problèmes d'optimisation multiobjectif et sur les méthodes de résolution proposées pour ces problèmes. Nous présentons les techniques de recherche Branch & Bound, de programmation dynamique et d'algorithme A*. Nous présentons tout particulièrement les métaheuristiques appliquées au cas multiobjectif. Un intérêt particulier sera porté sur l'étude des algorithmes génétiques multiobjectif.

Dans le dernier chapitre, nous proposons un algorithme évolutionnaire multiobjectif pour la résolution du problème d'emploi du temps.

# CHAPITRE I

## LA PLANIFICATION D'HORAIRES DE TRAVAIL

Ce chapitre met en scène la problématique de la planification des horaires dans un contexte général et sa complexité au quotidien dans les entreprises. En effet, la question de l'aménagement du temps de travail et de ses enjeux préoccupe toute société ou établissement actif ce qui a incité les chercheurs à proposer des méthodes et des techniques pour aider à gérer au mieux les horaires de travail. Pour cela nous définissons les différents types de plannings dans différents domaines de travail et plus particulièrement dans le domaine pédagogique.

## I-1 La problématique de la planification d'horaires de travail :

Les problèmes de planification d'horaires de travail se retrouvent autant dans les entreprises d'industrie que dans les services publics tels que : la santé, l'éducation etc…

La planification d'horaires de travail est un processus très complexe, qui vise à organiser des activités humaines (principalement de travail) dans le temps et à optimiser l'utilisation des ressources, de façon à couvrir un besoin exprimé par une charge de travail prévisionnelle sous diverses contraintes. Elle aboutit à des programmes définissant les horaires de travail et de repos de la force de travail [CHA02].

Pour mieux cerner ce qui est la planification et la complexité à sa réalisation, on s'intéresse à un ensemble de questions :

### I-1-1 Qu'est ce que la planification ?

La planification est un instrument de gestion dont l'objectif est d'aboutir à des programmes permettant d'organiser et planifier le travail des salariés afin de rester pérenne dans l'économie globale. Ceci passe par la détermination des capacités de tout un chacun et par le recensement des activités futures et des besoins en personnel.

La planification vise à affecter les ressources humaines pour chaque intervalle de temps sur un horizon donné, de telle manière que les besoins par intervalle soient couverts et que les différentes contraintes soient satisfaites [CHA02].

### I-1-2 Qu'est ce qu'un planning ?

Les plannings sont des calendriers de travail, où figurent à la fois le temps, l'affectation du personnel, les jours et les horaires de travail, et les congés et repos [WEI94]. En effet, ils apparaissent dans les cas suivants :

- Si le travail doit être assuré pendant plus d'une journée, il faut prévoir la succession de plusieurs personnes sur le même poste dans la journée. Un outil d'aide est nécessaire lorsque le nombre de postes dépasse la quinzaine, par exemple gérer les absences imprévues des salariés.

- Si le travail doit être assuré pendant plus de 35 heures par semaine, un outil automatique devient indispensable lorsque le nombre de postes dépasse la trentaine pour gérer la succession de plusieurs personnes dans la semaine, ainsi

que les absences imprévues.

Les plannings peuvent être utilisés pour planifier les horaires de présences du personnel ou les tâches effectuées par le personnel :

- Planning des horaires de présence : ce type de planning est utilisé pour prévoir les horaires de présence du personnel sans préciser les tâches journalières à effectuer soit pour des raisons de sécurité, soit pour une meilleure souplesse .

- Planning des tâches : ce type de planning est utilisé dans les entreprises à haute technicité, comportant plusieurs métiers et compétences distincts, où il est souhaitable d'affecter le personnel en fonction des tâches. Ce qui exige une décomposition fine des opérations et le repérage des tâches que chaque personne est capable d'accomplir.

Les plannings peuvent être journaliers (spécifiant les pauses et périodes de travail de la journée de chaque employé), hebdomadaires (utilisés pour une paie hebdomadaire), mensuels (utilisés pour le calcul des coûts pour les besoins de la paie mensuelle) ou annuels (permettant de gérer les congés annuels des employés).

Selon leur spécificité et les branches d'activités concernées, les plannings portent différents noms. Un planning spécifiant les programmes de travail de chaque employé nominativement sur un horizon (un intervalle de temps où un planning est élaboré) d'un mois est appelé tableau de service. Lorsque le planning représente les programmes de travail et de repos non nominatifs sur un nombre entier de semaines, on parle de grille de travail.

Certains plannings sont cycliques, s'ils reflètent une certaine périodicité des horaires individuels c'est à dire si au bout d'une durée D (mesurée généralement en semaine), le salarié retrouve son planning de départ. Autrement, ils sont dits acycliques c'est à dire ils sont différents chaque semaine.

### I-1-3 A Quoi sert un planning ?

Depuis le début des années 80, la gestion des ressources humaines à été reconnue comme une activité stratégique pour l'entreprise. Avec cette reconnaissance, l'intérêt d'élaborer des plannings s'est vu accroître de plus en plus car ils permettent :

- aux entreprises exerçant une activité continue ou quasi-continue de répartir convenablement leur personnel (compagnies aériennes, entreprises de transports, hôpitaux, etc…),

- aux entreprises cherchant à se rendre plus accessibles à la clientèle d'étaler les horaires d'ouverture (grands magasins, banques, etc…),

- à toutes les entreprises de surmonter leur exigences de productivité et de mieux gérer les présences et absences de leur personnel.

Les situations où un planning est utile son nombreuses. Elles justifient l'existence de différentes formes de plannings dans un même système : plannings à court, moyen et à long terme.[CHA02]

I-1-4 Comment est évalué un planning ?

Pour que les plannings élaborés soient satisfaisants, ils doivent vérifier un ensemble de contraintes et établir un meilleur compromis entre les différents acteurs (exemple : le chef d'entreprise, le planificateur, le commercial, le syndicaliste et le salarié).

Lorsque les différentes solutions alternatives sont connues, une négociation se déroule de la manière suivante : chaque acteur donne son opinion. Les points d'accord sont très vite expédiés et les points litigieux sont débattus. Et des solutions de compromis sont dégagées.

Les difficultés de négociation augmentent avec le nombre d'acteurs et le nombre de solutions alternative. L'aspect combinatoire (pour l'élaboration des plannings) rend d'autant plus difficile la négociation, car les opinions sont plus difficiles à formuler [REM03]. Les moyens informatiques apportent une aide certaine notamment dans l'acquisition et la confrontation des données individuelles.

I-1-5 Qui peut se charger de l'élaboration d'un planning ?

Dans la plupart des entreprises, cette tâche peut être centralisée ou déléguée à des cadres de l'entreprise appelés planificateurs.

Le planificateur doit prendre la décision qui correspond le mieux aux préférences des différents acteurs, justifier son choix, car son expérience de la tâche fait de lui un interlocuteur privilégié pour évaluer rapidement et effectuer des jugements de

l'orientation à donner à la recherche de solutions de meilleur qualité afin d'aboutir à un choix pertinent.

Une collaboration réussie doit permettre au planificateur de participer efficacement à l'élaboration des plannings. La génération automatique des planning joue un rôle primordial.

## I-2 Différents types de plannings :

Dans la construction de plannings d'horaires de travail, si créer un planning optimisé d'une journée est aisé, mais créer un bon planning pour un mois ou une année est beaucoup plus complexe. En plus de la complexité combinatoire du problème, il faut tenir compte de la diversité des contraintes applicables et qui sont souvent contradictoires.

Pour ce qui suit, on évoquera les différents types de plannings et les approches utilisées pour réaliser ces types de plannings.

### I-2-1 Types de plannings dans le domaine de la santé :

Les plannings dans le domaines de la santé sont des calendriers de travail où figurent à la fois le temps, et l'affectation des personnels (jours et horaires de travail, congés et repos). Ils sont établis au niveau de chaque équipe, ils sont à la fois une tâche, un document d'organisation du travail, et un élément contribuant à la gestion administrative du personnel. Cette tâche est parmi les plus difficiles et les plus délicates. Difficile parce qu'elle repose sur la recherche de solutions combinatoire, répond à des contraintes multiples, remise en cause de manière fréquente par l'absentéisme et délicate car elle impose toujours une négociation avec les acteurs (médecins, infirmiers) de l'équipe et la direction du service de soins et l'administration. Les documents établis sont des calendriers sur lesquels on inscrit les affectations des médecins et des infirmiers ; ils sont généralement des tableaux à double entrée avec en ligne le personnel et en colonne le temps.

L'objectif de la confection d'horaires en ce milieu est donc une combinaison variable de considérations en terme de coûts, de qualité des soins et de satisfaction du personnel. Mais les gestionnaires font souvent face à la difficulté d'obtenir des horaires réalisables qui satisfassent les contraintes.

Plusieurs méthodes ont été utilisées dans la littérature spécialisée pour gérer ce

type de plannings telles que la programmation par contraintes, la recherche locale (recuit simulé, tabou), les algorithmes évolutionnaires et d'autres méthodes. Parmi les techniques qui ont rencontré un certain succès, on peut citer :

Méthode heuristique de type recherche Tabou :

Cette méthode [JAU00] a pour objectif de développer diverses heuristiques de type recherche tabou pour la confection d'horaires d'infirmières avec des mouvements et des voisinages différents. C'est à dire développer une classe d'heuristiques qui peut s'adapter facilement aux caractéristiques des unités de soins pour lesquels des horaires sont développés. Dans cette optique, l'idée est de regarder des mouvements très locaux du type échange de quarts de travail jusqu'à des mouvements du type changement de l'horaire d'une infirmière pour les grosses unités de soins. Dans une première étape, les différentes heuristiques seront développées de façon indépendante, mais avec le souci de pouvoir les intégrer dans une même heuristique avec des stratégies variables dans une seconde étape. Comme, la recherche tabou est une technique d'optimisation sans contraintes, les transitions d'un état à un autre peuvent engendrer des violations de contraintes, sauf si elles ont été conçues spécifiquement.

Méthode de génération de colonnes :

Cette méthode [GEN01] consiste à reformuler le problème de confection d'horaires comme un problème se décomposant en un problème maître et des problèmes auxiliaires. Le problème maître contient les contraintes relatives à l'ensemble des horaires tandis que chaque problème auxiliaire contient les contraintes relatives à l'horaire d'un médecin, ou d'un groupe de médecins avec un profil semblable. Le problème auxiliaire correspond à un problème de plus courts chemins avec contraintes de ressources, soit avec un certain nombre d'étiquettes associées à chacun des sommets du graphe représentant les quarts de travail de l'horizon de planification (6 semaines par exemple). Le compromis efficacité-temps de calcul doit faire en sorte de minimiser ce nombre d'étiquettes, et le défi consiste à intégrer toutes les contraintes particulières des unités de soins avec un nombre minimum de telles étiquettes par sommet du graphe.

Cette méthode résout alternativement le problème maître et les problèmes auxiliaires jusqu'à l'obtention de la solution optimale. Le problème maître peut être résolu par l'algorithme du simplexe, qui consiste à couvrir les tâches avec un ensemble restreint de colonnes. Les problèmes auxiliaires génèrent des chemins à ajouter au problème maître pour améliorer la solution courante.

Programmation par contraintes :

Cette méthode [PES00] qui tire profit de nombreuses autres disciplines : mathématiques

discrètes, analyse numérique, intelligence artificielle, recherche opérationnelle et calcul formel a prouvé son intérêt et son efficacité dans de nombreux domaines. L'approche proposée dans le cadre de ce projet a pour objectif de développer un modèle de programmation par contraintes pour la confection d'horaires d'infirmières permettant de modéliser rapidement des contraintes complexes et produisant rapidement de bonnes solutions pour les problèmes peu contraints.

Dans le même type de projet (programmation par contraintes), un outil d'aide à l'élaboration des roulements infirmiers « Gymnaste » [WEI94] a été développé, il vise à mettre au point un logiciel d'aide à la planification et à la négociation des roulements infirmiers (prise en compte des vœux individuels, gain de temps, temps partiel, temps coupé, temps choisi, remplacements d'infirmiers inter-unités fonctionnelles…etc). L'approche proposée dans le cadre de ce projet consiste à considérer ce problème comme relevant d'une collaboration homme-machine intelligente : à l'utilisateur d'apprécier les facteurs humains nécessaires à la planification, à la machine de résoudre de façon optimale le problème proposé. Grâce à la programmation par contraintes et en particulier à sa souplesse et sa dynamicité, il est possible de faire collaborer simplement des algorithmes efficaces et l'intelligence humaine. La véritable difficulté réside dans la nécessité de bien dissocier ce qui relève de la négociation de ce qui relève du calcul combinatoire.

Algorithme hybride (programmation par contraintes et recherche Tabou) :

Ce projet [CAN02] a pour but la confection d'horaires de médecins par l'application d'une méthode générique combinant la programmation par contraintes et la recherche tabou afin d'étudier leur efficacité dans le contexte particulier d'exploration de voisinage par énumération implicite au moyen de la programmation par contraintes.

Intégration des méthodes de réécriture et de recherche opérationnelle pour la modélisation et la résolution de contraintes :

Le but de ce projet [BRAEF] est de coupler deux techniques différentes, la recherche opérationnelle (R.O) et les techniques de réécriture afin de résoudre des problèmes de gestion humaines et matérielles d'un service de radiologie d'un hôpital. L'idée générale est que si la (R.O) fourni des algorithmes efficaces, ces derniers ne sont valides que sur des structures mathématiques précises or il existe des contraintes symboliques difficilement représentables de cette manière. Pour cela les techniques de réécritures sont utilisées. Elles sont générales mais gourmandes en temps, pour traiter ces dernières et de se servir de la (R.O) pour traiter la partie combinatoire du problème.

Dans [ISK91,ZED04] d'autres approches sont proposées.

I-2-2 Types de plannings dans le domaine de transport :

Le transport est une activité complexe qui fait intervenir des investissements lourds, du personnel qualifié et une informatique très coûteuse.

En effet, dans le transport routier, il est toujours nécessaire de gérer aux mieux les ressources existantes en optimisant les investissements. Comme les clients exigent toujours plus de flexibilité, il faut offrir des services sur mesure, replanifier en permanence et en temps réel et gérer le personnel qualifié qui est une opération très complexe car il faut tenir compte de plusieurs contraintes (contrats, temps de travail, pénurie du personnel qualifié,…).

Dans le transport maritime, la gestion des escales et la gestion du personnel docker est aussi une activité complexe qui nécessite un effort considérable de la part des planificateurs. Les navires doivent rester à quai un temps minimum et les équipes docker doivent être disponibles. Cette activité représente un enjeu économique majeur. En effet, la qualité de la planification des travaux influe directement sur la rentabilité de l'activité de l'entreprise d'où la nécessité de la gestion des escales ( planifier le placement des navires sur les quais, planifier la disponibilité des ressources matérielles nécessaires, positionner des équipes sur des navires) afin d'optimiser les coûts liés aux chargements et déchargements des navires et la gestion du personnel docker (les besoins en équipe et en qualification pour chaque tâche issue de la gestion des escales et les contraintes liées à la gestion du personnel) afin d'optimiser l'affectation des ressources tout en tenant compte des contraintes liées à l'organisation du travail.

Dans le transport aérien, la gestion des flux de trafic aérien correspond aussi à des problèmes d'optimisation combinatoire dont la résolution est très complexe. En effet, le contrôle de la circulation aérienne organise les flux aériens afin d'assurer la sécurité des vols( en terme de risque de collision), d'améliorer la capacité du réseau de routes sur lequel les avions se déplacent et de construire des programmes de vols optimisé.

Plusieurs méthodes ont été proposées dans la littérature spécialisée pour confectionner des plannings dans le domaine de transport. Parmi ces techniques :

1- La programmation par contraintes :

Un outil de planification a été développé pour le secteur des transports routiers « j'Road planner ». Il s'adresse à toutes les exploitations ( de plus de 100 chauffeurs) qui ont pour objectif de mieux gérer l'organisation des tournées et d'optimiser l'affectation

des chauffeurs. Cet outil s'appuie notamment sur l'utilisation de la programmation par contraintes appliquée aux problématiques rencontrées dans le secteur des transports. Il propose automatiquement à l'utilisateur des solutions d'affectation en respectant à la fois la réglementation en vigueur et l'organisation du travail de l'entreprise (contraintes rigides ou souples) pour guider le calcul dans la recherche de la solution [DAUMA].

2- Les algorithmes génétiques :

Cette méthode consiste en la sectorisation de l'espace aérien. Ainsi la structure du réseau aérien a été synthétisé à l'aide d'un réseau de transport contenant essentiellement un ensemble de nœuds (aéroports ou balises), un ensembles d'arcs (routes aériennes) et un ensemble de paires origine-destination décrivant les demandes de flux entre les villes. La division de l'espace aérien en secteurs est faite en utilisant les algorithmes génétiques [DEL95].

3- Les réseaux de neurones :

La prédiction de trajectoires d'avions est un problème crucial pour les système de gestion du trafic aérien. La méthode proposée dans le cadre de ce projet est basée sur l'utilisation de réseaux de neurones auxquels ont fait apprendre un ensemble de trajectoires avant de les utiliser pour en prédire de nouvelles. En effet, en utilisant les premières positions connues de l'avion, son type et le niveau de vol qu'il désire atteindre, on prédit le reste du mouvement de montée. Pour y parvenir il faudra au préalable réaliser l'apprentissage du réseau de neurones sur un ensemble de trajectoires complètement connues constituant la base d'apprentissage. Ensuite, on compte sur la capacité des réseaux de neurones à s'adapter à des cas non appris pour pouvoir prédire d'autres trajectoires [YAN99].

### I-2-3 Types de plannings dans le domaine de la pédagogie :

La confection d'horaires (ou confection d'emploi du temps) dans les établissements scolaires est un travail très important, difficile à réaliser, c'est typiquement un problème de résolution de contraintes, NP-complet, dont la solution n'est pas , a priori connue dans le cas général. Pour fournir une solution , nécessite d'être capable de s'adapter aux changements dynamiques de l'environnement en tenant compte de la diversité des contraintes telles que l'interdépendance des programmes d'enseignement, la multitude des matières étudiées et les contraintes sur ces matières (cours, cours magistraux, TD, TP…), la durée des cours, les contraintes de disponibilité des enseignants, la disponibilité limitée des salles. C'est un problème qui peut être défini comme un problème qui fait assigner quelques évènements dans un nombre limité de

périodes. Il peut être divisé en deux catégories principales : la confection d'horaires des cours et la confection d'horaires des examens. Ces problèmes sont soumis à beaucoup de contraintes qui sont d'habitude divisées en deux catégories : « les contraintes dures » et « les contraintes souples »[BUR97].

La confection de plannings d'horaires est donc une tâche très difficile et sa solution manuelle peut exiger beaucoup d'effort ce qui a attiré énormément l'attention de la communauté scientifique. Comme notre travail se rapporte au problème de résolution d'emploi du temps d'université, on va essayer de voir l'historique des différentes recherches étudiées dans la littérature :

Une large variété d'approches et modèles ont été proposés pour traiter une variété de problèmes d'emploi du temps. Les problèmes s'étendent de la construction des emplois du temps semestriels ou annuels dans les universités, écoles ou collèges aux emplois du temps d'examens à la fin de ces périodes. Les premières activités d'emploi du temps ont été effectuées manuellement et un emploi du temps typique, une fois construit est resté statique avec seulement quelques changements nécessaires. Cependant la nature des enseignements à changé considérablement au cours des années et ainsi les exigences en matière de confection d'emploi du temps sont devenues beaucoup plus compliquées qu'ils ont eu l'habitude de l'être. Par conséquent le besoin de la génération automatisée d'emploi du temps augmente et ainsi le développement d'un système de génération d'emploi du temps qui produit des solutions valables est essentiel. En conséquence, pendant les 30 dernières années, beaucoup d'approches liées à l'automatisation des emplois du temps ont été publiées aux conférences et journaux. De plus , plusieurs applications ont été développés et mises en œuvres avec divers succès [SCH95a]. Les premières techniques employées dans la résolution du problème d'emploi du temps ont étaient basées sur la simulation de l'approche humaine dans la résolution du problème, ces techniques ont été appelées « les heuristiques directes », elles sont basées sur l'idée de créer un emploi du temps partiel en planifiant d'abord le cours le plus contraint, ensuite, cette solution partielle est étendue jusqu'à ce que tous les cours seront planifiés. L'étape suivante été l'application des techniques générales telles que la programmation linéaire et la coloration de graphes pour résoudre ce problème d'emploi du temps. De là, les premières publications sur la construction d'emploi du temps employant ces techniques générales sont attribuées à Kuhn et Haynes [SAN01].

L'intérêt de génération d'emploi du temps a augmenté dramatiquement dans les années 60 principalement en la raison de la disponibilité d'ordinateurs pour exécuter les algorithmes développés . Autour de la fin des années 60 quelques tentatives qui ont traité le problème en considérant des études de cas commençaient à être publiés. Par

exemple en 1969, Lawrie a développé un modèle pour le problème de confection d'horaire en employant l'approche de programmation linéaire. Pendant les années 1970, plusieurs publications ont abordé le problème d'emploi du temps . Les principales techniques qui semble avoir été plus répandu dans les années 1970 et les années 1980 sont les techniques ayant pour racine l'intelligence artificielle et sont basées sur les méthodes du recuit simulé, la recherche tabou et les algorithmes génétiques[SAN01]. En 1985, De Werra, a décrit les divers problèmes traitant le problème d'emploi du temps d'une façon formelle et a fourni les différentes formulation dans une tentative de les résoudre. Il a aussi décrit les approches considérées les plus importantes à ce temps là [WER85]. En 1986, Carter, a fait une analyse sur de réelles applications de confection d'emploi du temps de plusieurs universités. Junginger, a décrit dans la même année, les recherches faites en Allemagne sur le problème d'emploi du temps scolaires et les approches qui étaient basées sur des heuristiques directes, en particulier il a décrit les divers logiciels mis en œuvre et leur utilisation dans les divers établissements. En 1994, Corne, a fait une enquête sur l'application des algorithmes génétiques au problème d'emploi du temps et a discuté les futures perspectives de telles approches en comparant les résultats obtenus avec ceux obtenus avec d'autres approches[SAN01].

Bien qu'il y ait des publications dans les années 1990 sur la résolution du problème d'emploi du temps en employant les techniques basées sur l'IA, il y avait une nouvelle apparition d'une approche, aussi enracinée dans l'IA appelée la programmation de satisfaction de contraintes (CSP). En 1991, Abramson, a employé l'approche du recuit simulé comme technique d'optimisation. En 1993, Cooper et Kingston, ont décrit un programme informatique qui a résolu un problème d'emploi du temps d'un lycée fortement contraint sans aucune simplification. Un langage de spécification du problème d'emploi du temps a été fourni pour aider à éviter beaucoup de contraintes d'une façon uniforme. En 1994, Costa, a discuté des différents types de contraintes qui doivent être tenues en compte [SAN01]. En 1999, Tsang, Mills, Williams, Ford et Borret, ont discuté de l'importance de la technique de satisfaction de contraintes pour la résolution du problème de confection d'horaires et ont fourni une introduction dans ce domaine. Dans la même année, Schaerf, a fourni une enquête sur les différentes techniques employées pour la génération des emplois du temps. Les techniques de satisfaction de contraintes ont été soulignées comme un complément important aux outils qui sont employés dans la résolution du problème d'emploi du temps[SAN01].

Dans les dernières décennies, les sujets de résolution du problème d'emploi du temps ont été principalement limités à la (RO) (les techniques employées étaient naturellement mathématiques). Dans la décennie actuelle, la contribution de l'IA a fourni au problème de résolution de l'emploi du temps une heuristique moderne telle que les algorithmes génétiques, le recuit simulé et la recherche tabou [SCH95a].

Par conséquent, les problèmes d'emploi du temps ont attirés l'attention de la communauté scientifique incluant la (RO) et l'IA pour environ 40 ans et au cours de la dernière décennie, il y a eu un intérêt accru dans ce domaine et plusieurs méthodes ont été décrites dans la littératures comme suit :[BUR02]

- Les méthodes séquentielles :

Ces méthodes ordonnent les évènements en les assignant séquentiellement dans des périodes de temps valides pour qu'aucun évènement dans une période ne soit en conflit avec un autre dans une période de temps donné . Dans les méthodes séquentielles, les problèmes de confection d'horaires sont généralement représentés par des graphes où les évènements (cours/examens) sont représentés par des nœuds et les conflits entre les évènements sont représentés par les arcs[WER85]. Par exemple si quelques étudiants doivent suivrent deux évènements, il y a un arc entre les nœuds qui représentent le conflit. La construction d'un emploi du temps peut donc être modelée comme un problème de coloration de graphe. Chaque fois la période dans l'emploi du temps correspond à une couleur et les nœuds du graphique sont colorés de telle façon qu'aucun des nœuds adjacents ne soit coloré par la même couleur. Une variété d'heuristiques de coloration de graphe pour la construction des conflits d'emploi du temps est disponible dans la littérature. Ces heuristiques ordonnent les évènements basés sur une évaluation de comment il est difficile de les prévoir. Les heuristiques qui sont souvent employées sont :

- le plus grand degré d'abord :

Les évènements qui ont un grand nombre de conflits avec d'autres évènements sont prévu tôt. Le raisonnement est que les évènements avec un grand nombre de conflits sont plus difficiles à prévoir et donc doivent être aborder d'abord.

- le plus grand degré pondéré :

C'est une modification du plus grand degré d'abord où le poids de chaque conflit est représenté par le nombre d'étudiants impliqués dans le conflit.

- le degré de saturation :

Dans chaque pas de la construction de l'emploi du temps, l'événement qui a un nombre petit de périodes valables disponibles pour la planification dans l'emploi du temps est choisi lointainement.

- Les méthodes basées contraintes :

Dans ces méthodes, le problème d'emploi du temps est modelé comme un jeu

de variables ( c à d les évènements), les valeurs ( c à d les ressources comme les pièces et les périodes) vont être assignées pour satisfaire un certain nombre de contraintes . D'habitude quelques règles sont définies pour l'assignation de ressources aux évènements. Quand aucune règle n'est applicable à la solution partielle actuelle un retour arrière est exécutée jusqu'à une solution est trouvée qui satisfait toutes les contraintes.

- Les méthodes métaheuristiques :

Pendant les deux dernières décennies une variété d'approches métaheuristiques comme le recuit simulé, la recherche tabou, les algorithmes génétiques, les colonies de fourmis et les approches hybrides ont été étudiées pour la résolution du problème d'emploi du temps . Les métaheuristiques commencent par une ou plusieurs solutions initiales et emploient les stratégies de recherche qui essayent d'éviter des optimums locaux. Tous ces algorithmes de recherche peuvent produirent des solutions de qualité mais ont souvent un coût informatique considérable.

Les algorithmes génétiques sont une classe des algorithmes de recherche stochastiques qui emploient la génétique comme modèle pour la résolution du problème. L'application de la reproduction, la sélection et la mutation peut donner beaucoup d'avantages pour la survie de nouvelles générations. De façon similaire, le recuit simulé a été décrit comme une méthode d'optimisation basée sur une analogie physique. Il a été démontré que le recuit simulé est une bonne technique pour la résolution des problèmes d'optimisation combinatoire dure. La recherche tabou, est une métaheuristique qui guide une procédure de recherche locale pour explorer l'espace de solution au delà de l'optimum local. Les algorithmes de colonies de fourmis s'inspirent du comportement naturel des fourmis où chaque fourmi dépose, le long de son chemin une substance chimique (le phéromone), tous les membres de la colonie perçoivent cette substance et orientent leur marche vers les régions les plus « odorantes », il en résulte la faculté collective de retrouver le plus court chemin. Ces algorithmes sont très indiqués pour les problèmes distribués par nature et les problèmes susceptibles d'évolution dynamique. Les approches hybrides associent souvent une métaheuristique et une méthode locale afin d'affiner la solution. Cette coopération peut prendre la simple forme d'un passage de relais entre la métaheuristique et la technique locale, comme elles peuvent être entremêlées de manière plus complexe.

Parmi les modèles proposés pour confectionner des emploi du temps citons quelques exemples :

les méthodes évolutives [BABOU] : où l'approche proposée est fondée sur une

programmation par contraintes utilisant un algorithme génétique comme moteur d'optimisation.

les méthodes utilisant les colonies de fourmis [SOC02] : où les auteurs de ce projet procèdent à une simplification du problème d'emploi du temps d'université en impliquant trois types de contraintes dures et trois types de contraintes souples. Le système de fourmi « Max-Min Ant System » se sert d'une routine de recherche locale séparée, proposée pour aborder ce problème, une construction de graphe approprié et une représentation de matrice de phéromones sont conçus et les résultats de ce système ont démontré qu'il peut construire des horaires meilleurs qu'un algorithme qui réitère le procédé de recherche locale des solutions.

Les méthodes utilisant la recherche tabou [SCH96] : où Schaerf à utilisé cette technique pour résoudre le problème. Il a employé le codage matriciel Mi,j qui contient le nom de la classe du professeur i à la période j. Les voisinages proposés sont :

- Echanger deux cours pour un même professeur.
- Déplacer un cours à une autre période.

Pour des instances de tailles moyennes, il a été démontré que les résultats obtenus ont été encourageants.

D'autres méthodes plus spécifiques faisant appel a des modèles de coloration de graphes [WER85], de relaxation lagrangienne [TRI80] et de réseaux de neurones [CAO91].

## I-3 Conclusion :

On peut conclure que la planification des horaires présente des enjeux à la fois sur un plan économique et un plan social. Toutefois, sa complexité impose de s'appuyer sur une démarche scientifique pour apporter des réponses pragmatiques à une catégorie générale de problèmes.

Il s'agit donc de développer des outils de planification d'horaires, basés sur des techniques efficaces d'optimisation de ressources qui permettent de construire des programmes de travail, respectant la réglementation du travail et garantissant une bonne couverture de charge tout en limitant les coûts.

Parmi tous les types de plannings cités, c'est sur les plannings pédagogiques que nous allons porter notre intérêt, et plus particulièrement sur les plannings ou emploi du temps des cours d'université.

Malgré l'éventail de logiciels qui ont essayé de traiter le problème de l'emploi du temps et la multitude d'approches utilisées, le problème reste toujours posé, car le problème lui même n'est pas standard. Aucune modélisation standard, qui englobe toutes les variantes du problème, n'a été formulée.

Le problème de l'emploi du temps des cours est un processus complexe, c'est un problème d'optimisation combinatoire très difficile à résoudre. Car une solution au problème est représentable par un ensemble de propriétés. Le but est d'atteindre la meilleure combinaison de ces propriétés. Une autre cause de complexité du problème est le volume du problème. En effet la taille du problème pour des établissements modestes atteint facilement l'ordre de 1000 enseignements à ordonner par semaine, d'une façon concurrente sur trois plans : périodes, enseignants et locaux. Aussi l'estimation du degré de satisfaction des contraintes de préférence est souvent difficiles à formuler. En plus du fait qu'elles peuvent être parfois contradictoires.

Plusieurs approches ont été proposées pour le problème de l'emploi du temps . Les premières tentatives de résolutions étaient les méthodes basées sur la théorie des graphes, la programmations linéaire, des modèles qui ont prouvé leurs inadéquations pour ce problème. Ce genre d'approche a cédé le pas à des méthodes relativement nouvelles qui s'adaptent assez bien pour ce type de problème, ces méthodes se regroupent principalement parmi les métaheuristiques telles que la recherche tabou, le recuit simulé, les colonies de fourmis et les algorithmes génétiques. Cette famille de recherche approchée est dotée de mécanismes généraux lui permettant une bonne investigation de l'espace de recherche. Un survol des principes de base a été fait.

Le chapitre suivant, sera dédié à l'étude détaillée d'une multitude de ces approches, les fondements de base les régissant et leurs applicabilité au problème de l'emploi du temps. Nous dresserons aussi une synthèse critique qui se veut objective.

# CHAPITRE II

# METAHEURISTIQUES ET OPTIMISATION COMBINATOIRE

La vie courante nous offre un vaste éventail de problèmes où on se trouve en face d'un ensemble important de solutions potentielles dans lequel il s'agit de trouver la meilleur solution qui soit. Le nombre important de ces choix rend impossible leurs parcours exhaustif en vue de recenser le choix le plus adéquat.

L'objectif de ce chapitre est de présenter une étude sur les méthodes appliquées pour la résolution de ces problèmes tout en formulant une analyse critique quant à leur efficacité. Notre attention sera particulièrement portée sur les problèmes d'emploi du temps.

## II-1 Introduction :

L'optimisation combinatoire est une voie d'études importante, elle occupe une place très considérable en recherche opérationnelle et en informatique, elle est le domaine des mathématiques discrètes qui traite de la résolution du problème suivant :

> Soit X un ensemble de solutions admissibles. Soit f une fonction permettant d'évaluer chaque solution admissible. Il s'agit de déterminer une solution s* appartenant à X qui minimise (ou maximise) f. l'ensemble X des solutions admissibles est supposé fini et est en général défini par un ensemble C de contraintes.

En effet la plupart de ces problèmes sont très complexes et ne possèdent donc pas à ce jour de solutions algorithmiques efficaces.

Gérer l'alternance de périodes d'intense activité et de périodes creuses, répartir les charges de travail de façon équitable, rejoindre les différents contrats de travail des enseignants avec les séances de la semaine sont parmi les enjeux de la gestion des emplois du temps et la confection de ces derniers est très difficile car il ne faut en aucun cas sous estimer l'effort nécessaire pour leur trouver une solution. Ces problèmes font partie de problèmes d'optimisation combinatoire pour les quels, dans la majorité des cas, il est très difficile de trouver la solution optimale.

Etant donné l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO) et en intelligence artificielle (IA). Ces méthodes peuvent être classées en deux grandes catégories :

- les méthodes exactes (complètes) capables de trouver la solution optimale si elle existe et de prouver l'inconsistance du problème sinon,

- les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité.

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques[1] spécifiques

[1] une heuristique est une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsqu'on lui fourni une instance d'un problème. ( du grec Heuriskein, signifiant « art de trouver). Les heuristiques ne garantissent pas les choix les plus efficaces, mais elles doivent y parvenir le plus souvent.

pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles telles que les techniques de séparation et évaluation progressive ou les algorithmes avec retour arrière. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable. Malgré les progrès réalisés (notamment en matière de la programmation linéaire en nombre entiers), comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux problèmes de taille importante.

Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille si la complétude n'est pas primordiale. Les premières méthodes approchées étaient conçues spécifiquement pour un problème donné.

En effet, à partir des années 1980, une nouvelle génération de méthodes approchées ont apparues souvent appelées métaheuristiques, elles sont inspirées par des analogie : avec la physique (recuit simulé), avec la biologie ( algorithmes évolutionnaires, recherche avec tabou) ou avec l'éthologie ( colonies de fourmis).

Ces méthodes sont très puissantes, adaptables et applicables à une large classe de problèmes de plus grande taille qui étaient impossible de les traiter auparavant.

Avant de faire une présentation relativement complète des principales méthodes, nous suggérons de décrire brièvement et de manière générale quelques notions fondamentales inspirées de [PAR98].

## II-2 Notions fondamentales :

Parmi les objectifs de la théorie de complexité est de classer les problèmes en fonction des ressources (temps de calcul, espace mémoire, etc…) nécessaires à leur résolution algorithmique. Ceci a montré qu'il existe des problèmes qui ont une solution calculable mais dont toute réalisation effective sur une machine est pratiquement inutilisable parce que le temps de calcul ou la place mémoire nécessaire sont trop importants. L'enjeu est donc de discerner entre les problèmes qui ont une solution réalisables et ceux qui, quelles que soient les améliorations futures des machines, ne peuvent intrinsèquement avoir une telle solution. Quand on prend pour critère le temps d'exécution, on exprime cette frontière en considérant qu'un problème est réalisable si son temps d'exécution est polynomial en fonction de la taille de l'entrée. Si le degré du polynôme est élevé cette notion devient un peu irréalisable, mais la distinction entre temps polynomial et temps non polynomial donne naissance à une classification des

problèmes qui est très utile pratiquement.

II-2-1 Notion de complexité :

D'une manière générale, pour résoudre un problème, on est appelé à trouver l'algorithme le plus efficace. Cette notion d'efficacité induit normalement toutes les ressources de calcul nécessaires pour exécuter un algorithme. Or, le temps d'exécution est généralement le facteur dominant pour déterminé si un algorithme est assez efficace pour être utilisé dans la pratique, pour cela on se concentre principalement sur cette ressource.

On appelle complexité en temps d'un algorithme le nombre d'instructions élémentaires mises en œuvre dans cet algorithme afin de résoudre un problème donné. Une instruction élémentaire sera une affectation, une comparaison, une opération algébrique, la lecture et l'écriture etc.… Mais comme le décompte précis de toutes les instructions d'un programme risque d'être assez pénible, et qu'entre deux exécution du même algorithme avec un jeu de paramètres différent, le nombre d'instructions exécutées peut changer, on se contentera, en général, d'apprécier un ordre de grandeur de ce nombre d'instructions. C'est ce qu'on désigne sous le nom de complexité de l'algorithme. Donc pour mesurer la complexité temporelle d'un algorithme, on s'intéresse plutôt aux opérations les plus coûteuses :

- Racine carrée, Log, Exp, Addition réelle …
- Comparaisons dans le cas des tris …

On dit que ***f(n) = O (g(n))*** ( f(n) est de complexité g(n)), chaque fois qu'il existe k et $n_0$ tels que :

$$n > n_0 \Rightarrow f(n) < kg(n)$$

II-2-2 Algorithmes polynomiaux et Algorithmes exponentiels :

Un algorithme polynomial est un algorithme dont la complexité est ***O(p(n))*** où p est une fonction polynomiale et n dénote la longueur de données. Tout algorithme dont la complexité ne peut être bornée par un tel polynôme d'ordre n, est un algorithme exponentiel (bien que cette définition inclue certaines complexité non-polynomiales comme $n^{\log n}$, qui n'est pas considéré comme fonction exponentielle).

II-2-3 Problèmes combinatoires :

Un problème combinatoire est un problème où il s'agit de trouver la meilleur

combinaison possible de solutions. Un tel problème peut être soit un problème de décision, un problème de recherche ou un problème d'optimisation, selon la question à la quelle on est censé répondre.

II-2-3-1 Problème de décision :

Un problème de décision est un problème où la résolution se limite à la réponse par « oui » ou par « non » à la question de savoir s'il existe une solution au problème. Par conséquent, il n'est pas nécessaire de trouver la solution proprement dite.

II-2-3-2 Problème de recherche :

Dans ce cas précis, la résolution du problème ne s'arrête plus au point de savoir si le problème admet ou non une solution. L'algorithme doit être en mesure de fournir la solution si celle ci existe . Par conséquent, tout problème de décision peut être étendu à un problème de recherche.

II-2-3-3 Problème d'optimisation :

Un problème d'optimisation est obtenu à partir d'un problème de recherche en associant à chaque solution une valeur. Il consiste à trouver parmi un ensemble de solutions potentielles celle qui répond le mieux à certains critères décrits sous forme d'une fonction objectif à maximiser ou minimiser c'est à dire on cherchera une solution de valeur optimale, minimale, si on minimise la fonction objectif, et maximale, si on la maximise.

II-2-4 La réduction polynomiale:

On dit qu'un problème $P_1$ se réduit polynomialement en un problème $P_2$, s'il existe un algorithme polynomial A construisant, à partir d'une donnée $D_1$ de $P_1$ , une donnée $D_2$ de $P_2$ telle que la réponse pour $D_1$ soit oui si et seulement si la réponse pour $D_2$ est oui.

II-2-5 La réduction de Turing:

La réduction polynomiale permet de comparer les problèmes de décision. La réduction de Turing permet de comparer les problèmes de recherche.
On dit qu'un problème de recherche $P_1$ se réduit polynomialement à un problème de recherche $P_2$ par la réduction de Turing s'il existe pour résoudre $P_1$ un algorithme $A_1$ utilisant comme sous–programme un algorithme $A_2$ résolvant $P_2$ , de telle sorte que la

complexité de $A_1$ est polynomiale, quand on évalue chaque appel de $A_2$ par une constante.

II-2-6 Les différentes classes de complexité :

II-2-6-1 La classe P :

La classe P contient tous les problèmes relativement faciles c'est à dire ceux pour lesquels on connaît des algorithmes efficaces. Plus formellement, ce sont les problèmes pour lesquels on peut construire une machine déterministe( e.g. une machine de Turing[2]) dont le temps d'exécution est de complexité polynomiale (le sigle P signifie « Polynomial time »).

II-2-6-2 La classe NP :

Les problèmes de la classe NP sont ceux pour lesquels on peut construire une machine de Turing non déterministe dont le temps d'exécution est de complexité polynomiale (le sigle NP provient de « Nondeterministic Polynomial time ») (et non de « Non Polynomial).

Contrairement aux machines déterministes qui exécutent une séquence d'instructions bien déterminée, les machines non déterministes ont la remarquable capacité de toujours choisir
la meilleur séquence d'instructions qui mène à la bonne réponse lorsque celle-ci existe. Ce concept abstrait est en fait la base de toute la théorie de la NP-complétude.

II-2-6-3 La classe NP-complets :

Parmi l'ensemble des problèmes appartenant à NP, il en existe un sous-ensemble qui contient les problèmes les plus difficiles : on les appelle le problèmes NP-complets. Un problème NP-complets possède la priorité que tout problème dans NP peut être transformé (réduit) en celui-ci en temps polynomial. C'est à dire qu'un problème est NP-complets quand tous les problèmes appartenant à NP lui sont réductibles. Si on trouve un algorithme polynomial pour un problème NP-complets, on trouve alors automatiquement une résolution polynomiale de tous les problèmes de la classe NP.

[2] Une machine de Turing est constituée d'un ensemble fini de bandes composées d'un nombre infini de cellules. Chaque cellule, si elle n'est pas vide, contient un symbole représentant une information nécessaire au calcul. Chaque bande ne comporte qu'un nombre fini de cellules non vides. Sur chacune des bandes une t^te de lecture-écriture se déplace de cellule en cellule faisant ainsi évoluer les informations contenues dans les bandes et le comportement de la est entièrement décrit par une table finie représentant les actions possibles des têtes de lecture-écriture.

II-2-6-4 La classe NP-difficiles :

Un problème est NP-difficile s'il est plus difficile qu'un problème NP-complet, c'est à dire s'il existe un problème NP-complet se réduisant à ce problème par la réduction de Turing.

Ceci explique pourquoi, lors de l'étude d'un nouveau problème, on commence par chercher à classer ce problème. Si l'on parvient à montrer qu'il est polynomial, le problème sera résolu. Si par contre, on parvient à montrer qu'il est NP-complet, la recherche d'un algorithme exact pour résoudre un tel problème ne sera pas de première priorité, et il sera approprié de se concentrer sur des méthodes heuristiques que la plupart des spécialistes de l'optimisation combinatoire ont orienté leurs recherche pour les développer. Une méthode heuristique est souvent définie comme une procédure exploitant au mieux la structure du problème considéré, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible.

II-3 Caractéristiques des problèmes d'optimisation combinatoire :

Le premier type des problèmes d'optimisation combinatoire est le problème d'optimisation sans contraintes. Dans ce type de problèmes, l'optimisation peut s'effectuer en tout point de l'espace de recherche puisqu'il y a absence de contraintes.

Parfois un problème n'a pas de critère à optimiser. Les problèmes qui n'ont pas de critère d'optimisation, mais qui possèdent un ensemble de contraintes sont des problèmes de satisfaction de contraintes.

L'ajout au problème précédent d'un critère d'optimisation crée un autre type de problème appelé problème d'optimisation combinatoire avec contraintes. Ce dernier est le cas général des problèmes d'optimisation mono-objectif. Les problèmes d'optimisation avec contraintes sont très importants puisque la majorité des problèmes d'optimisation le sont.

Un problème d'optimisation mono-objectif se définit alors comme la recherche de l'optimum (du minimum ou du maximum) d'une fonction donnée. Les variables de cette fonction sont souvent contraintes d'évoluer dans une certaine partie de l'espace de recherche.

Mathématiquement, un problème d'optimisation mono-objectif s'écrit :

| | | |
|---|---|---|
| Minimiser | $f(x)$ | (une seule fonction à optimiser) |
| avec | $g(x) \leq 0$ | (m contraintes d'inégalités) |

et $h(x) = 0$ (p contraintes d'égalité)
où $x \in \Re^n$, $g(x) \in \Re^m$, et $h(x) \in \Re^p$

En effet un problème de maximisation peut être aisément transformé en un problème de minimisation en considérant l'équivalence suivante :
Maximiser $f(x) \Leftrightarrow$ Minimiser $- f(x)$

Vocabulaire et définitions :

- Fonction objectif : c'est la fonction f qu'on cherche à optimiser

- Variable de décision : elles sont regroupées dans le vecteur x qui correspond à l'ensemble des variables du problème.

- Minimum global : un point x* est appelé minimum global de la fonction f si :
$$\forall x, x \neq x^* \Rightarrow f(x^*) < f(x)$$

- Minimum local fort : : un point x* est appelé minimum local fort de la fonction f si :
$$\forall x \in v(x^*), x \neq x^* \Rightarrow f(x^*) < f(x)$$
où $v(x^*)$ représente le voisinage de x*

- Minimum local faible : un point x* est appelé minimum local faible de la fonction f si :
$$\forall x \in v(x^*), x \neq x^* \Rightarrow f(x^*) \leq f(x)$$

II-4 Méthodes de résolution des problèmes d'optimisation mono-objectif:

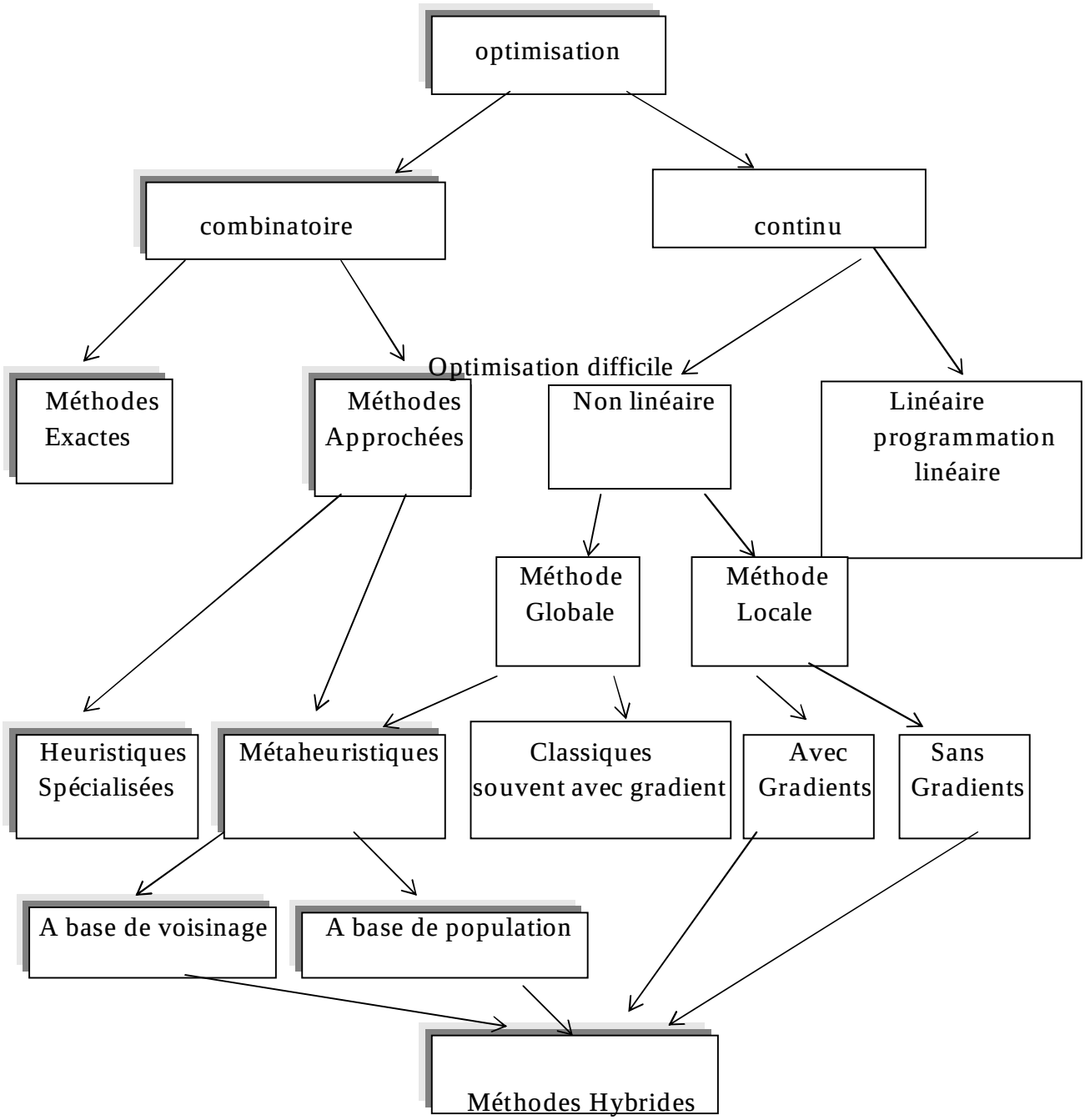



Fig 2.1 Classification des méthodes d'optimisation mono-objectif

Le schéma précédent présente une classification générale des méthodes de résolution des problèmes d'optimisation mono-objectif. On distingue en premier lieu l'optimisation continue de l'optimisation combinatoire. Pour l'optimisation continue, on sépare le cas linéaire du cas non linéaire, où l'on retrouve le cadre de l'optimisation difficile, dans ce cas on fait appel à une méthode locale qui exploite ou non les gradients de la fonction objectif. Si le nombre de minimums locaux est très élevé, le recours à une méthode globale s'impose : on retrouve alors les métaheuristiques. Pour l'optimisation combinatoire, on utilise les méthodes exactes. Lorsqu'on est confronté à un problème difficile on a recours aux méthodes approchées, dans ce cas le choix est parfois possible entre une heuristique spécialisée, dédiée au problème considéré, et une métaheuristique. Parmi les métaheuristiques, on peut différencier les métaheuristiques à base de voisinage, et les métaheuristiques à base de population. Enfin les méthodes hybrides associent souvent une métaheuristique et une méthode locale.

II-4-1 Les méthodes exactes :

Les méthodes exactes sont des méthodes qui garantissent la complétude de la résolution autrement dit ces méthodes donnent à tous les coups la solution optimale. Le temps de calcul nécessaire de telles méthodes augmente en général exponentiellement avec la taille du problème à résoudre. On distingue dans ce cas l'approche constructive qui est probablement la plus ancienne et occupe traditionnellement une place très importante en optimisation combinatoire. Une méthode constructive construit pas à pas une solution de la forme $s = ( <V_1,v_1> <V_2,v_2>\ldots<V_n,v_n> )$ en partant d'une solution partielle initialement vide s = 0, elle cherche à étendre à chaque étape la solution partielle $s = ( <V_1,v_1> \ldots <V_{i-1},v_{i-1}> )$ ( i <= n ) de l'étape précédente. Pour cela, elle détermine la prochaine variable $V_i$, choisit une valeur $v_i$ dans Di et ajoute $<V_i,v_i>$ dans s pour obtenir une nouvelle solution partielle $s = ( <V_1,v_1>\ldots<V_n,v_n> <V_i,v_i> )$. Ce processus se répète jusqu'à ce que l'on obtienne une solution complète.

Durant la recherche d'une solution, la méthode constructive fait intervenir des heuristiques pour effectuer chacun des deux choix : le choix de la variable suivante et le choix de la valeur pour la variable. Les méthodes de cette classe diffèrent entre elles selon les heuristiques utilisées. En général , les heuristiques portent plus souvent sur le choix de variables que sur le choix de valeurs car les informations disponibles concernant le premier choix semblent souvent plus riches. La performance de ces méthodes dépend largement de la pertinence des heuristiques employées, c'est à dire, de leur capacité d'exploiter les connaissances du problème.

Un premier type de méthodes constructives est représenté par les méthodes

gloutonnes. ces méthodes consistent à fixer à chaque étape la valeur d'une variable sans remettre en cause les choix effectués précédemment.

Un deuxième type de méthodes constructives est représenté par les méthodes avec retour arrière. Ces méthodes de retour arrière avec une stratégie de recherche en profondeur d'abord consistent à fixer à chaque étape la valeur d'une variable. Aussitôt qu'un échec est détecté, un retour arrière est effectué, c'est à dire, une ou plusieurs instanciations déjà effectuées sont annulées et de nouvelles valeurs recherchées. Les méthodes avec retour arrière sont en général complètes et de complexité exponentielle. Pour réduire le nombre de retour arrière (et le temps de recherche), on utilise des techniques de filtrage afin d'anticiper le plus tôt possible les échecs. Par exemple : ALICE, PROLOG III sont des systèmes de programmation sous contraintes fondés sur le principe de retour arrière.

Un troisième type de méthodes constructives est représenté par de nombreux algorithmes basés sur le principe de séparation et évaluation progressive, qui ont pour principe la construction d'un arbre de recherche dont le problème initial (problème de minimisation) est la racine. On divise le problème en sous problèmes (en deux ou plus) en introduisant par exemple une contrainte supplémentaire, qui peut être satisfaite ou non. L'optimum peut appartenir à l'un quelconque de ces sous problèmes. Tout sous problème infaisable sera éliminé. Si possible on calcule la solution du problème, sinon, on calcule une borne inférieure, si elle est supérieure de la meilleure solution déjà obtenue on élimine le sous-problème. Dans le cas restant, on subdivise à nouveau le sous problème. Pour améliorer l'efficacité de la recherche , on utilise des techniques variées pour calculer des bornes permettant d'élaguer le plus tôt possible des branches conduisant à un échec. Parmi ces techniques on peut citer : la relaxation de base en programmation linéaire et la relaxation lagrangienne .

Une autre méthode exacte, la méthode de programmation dynamique, c'est une méthode découverte par Bellman en 1956, elle c'est avérée une méthode très efficace pour la résolution des problèmes d'optimisation combinatoire, elle est conçu sur le modèle de l'algorithme du plus court chemin dans un graphe. Elle consiste à décomposer la résolution du problème initial en une suite de problèmes plus simples, la résolution du n-ème se déduisant de celle du (n-1)-ème par une équation récurrence[FEA05].

Plusieurs chercheurs ont tenté de résoudre les problèmes NP-difficiles par les méthodes exactes par exemple dans notre cas : pour la résolution du problème de l'emploi du temps en utilisant les approches basées sur la théorie des graphes ,on constate que ces approches partagent en commun le fait de s'appuyer sur les notions :

du nombre chromatique, l'indice chromatique ou le nombre de stabilité. Il s'agit alors de modéliser le problème réel en un problème de coloration ou de recherche de sous graphes stables :

- Modélisation par coloration de graphe : il s'agit de colorier les sommets d'un graphe avec un nombre minimum de couleurs (nombre chromatique du graphe) tels que deux sommets adjacents quelconques n'ont pas la même couleur. Dans ce cas les sommets correspondent aux enseignements et deux enseignements sont mis en correspondance s'ils ne peuvent, pour une raison ou une autre, se dérouler en même temps. Le nombre chromatique qu'on peut trouver correspond au minimum de périodes nécessaire pour la programmation de tous les enseignements. Chaque couleur correspondra alors à une période donnée.

- Modélisation par ensembles de stables : un graphe est alors construit comme précédemment où les sommets représentent les enseignements et les arêtes les contraintes de non simultanéité. Il s'agit dans ce cas, de dégager un partitionnement des sommet du graphe en sous graphes stables de cardinalités inférieures à L. L désigne le nombre de locaux disponibles. Chaque sous graphe stable correspond à un sous ensemble d'enseignements qui doivent être planifiés à la même période.

Les approches de résolution se basant sur la théorie des graphes soufrent cependant de plusieurs lacunes, la plus importantes réside dans l'impossibilité de modéliser l'ensemble de toutes les contraintes.

D'autres méthodes exactes ont tenté aussi de résoudre le problème de l'emploi du temps tels que: les méthodes constructives [OST82] et de relaxation lagrangienne [TRI80].

Cependant, malgré les progrès réalisés au niveau des méthodes exactes, qui ont aidé à résoudre les problèmes de manière optimale, ces méthodes rencontrent généralement des difficultés face aux instances de taille importantes car la recherche d'une solution optimale peut être totalement inappropriée dans certaines applications pratiques en raison de la dimension du problème, de la dynamique qui caractérise l'environnement de travail, du manque de précision dans la récolte des données, de la difficulté de formuler les contraintes en terme explicites ou de la présence d'objectifs contradictoires.

Compte tenu de ces difficultés, la plupart des spécialistes de l'optimisation

combinatoire ont orienté leur recherche vers le développement de méthodes heuristiques qui exploitent au mieux la structure du problème considéré. Cela a conduit à une avancée importante pour la résolution pratique de nombreux problèmes.

II-4-2 Les méthodes approchées :

Contrairement aux méthodes exactes, les méthodes approchées ne procurent pas forcément une solution optimale, mais seulement une bonne solution (de qualité raisonnable) en un temps de calcul aussi faible que possible.

Une partie importante des méthodes approchées est désignée sous le terme de métaheuristiques. Plusieurs définitions d'une métaheuristique ont été proposées dans la littérature[BLU03,WKIP], cette définition est celle adoptée par le « ***Metaheuristics Network »[METAH]: « A metaheuristics is a set of concepts that can be used to define heuristic méthods that can be applied to a wide set of different problems ».***

Plusieurs classifications des métaheuristiques ont été proposées ; la plupart distinguent globalement deux catégories : les méthodes à base de solution courante unique, qui travaille sur un seul point de l'espace de recherche à un instant donné, appelées méthodes à base de voisinage comme les méthodes de recherche locale( méthode de la descente), de recuit simulé et de recherche tabou, et les méthodes à base de population, qui travaillent sur un ensemble de points de l'espace de recherche, comme les algorithmes évolutionnaires et les algorithmes de colonies de fourmis.

II-4-2-1 Les méthodes à base de voisinage :

Dans les problèmes d'optimisation où l'on cherche à optimiser une fonction objectif sur un espace de recherche donné, une petite perturbation sur un point de cet espace induit souvent une petite variation des valeurs de la fonction objectif en ce point. On déduit que les bonnes solutions ont tendance à ce trouver à proximité d'autres bonnes solutions, les mauvaises étant proches d'autres mauvaises solutions. D'où l'idée qu'une bonne stratégie consisterait à se déplacer à travers l'espace de recherche en effectuant de petits pas (petits changements sur le point courant) dans des directions qui améliorent la fonction objectif. Cette idée est la base d'une grande famille d'algorithmes appelée méthodes à base de voisinage ou de recherche locale.

Les méthodes de voisinage (ou méthodes de recherche locale) s'appuient toutes sur un même principe : elles résolvent le problème d'optimisation de manière itérative. Elles débutent avec une configuration initiale (souvent un tirage aléatoire dans l'espace des configurations), et réalisent ensuite un processus itératif qui consiste à effectuer un

mouvement[3] choisi par le mécanisme d'exploration[4] en tenant compte de la fonction de coût. ce processus s'arrête et retourne la meilleure configuration trouvée quand la condition d'arrêt est réalisée. Cette condition d'arrêt peut porter sur le nombre d'essais effectués, sur une limite temporelle ou sur le degré de qualité de la meilleure configuration courante. Cette versatilité permet de contrôler le temps de calcul, la qualité de la solution optimale trouvée s'améliorant au cours du temps. De manière générale les opérateurs de recherche locale s'arrêtent quand une solution localement optimale est trouvée, c'est à dire quand il n'existe pas de meilleure solution dans le voisinage.

Les méthodes de voisinage diffèrent essentiellement entre elle par le voisinage utilisé et la stratégie de parcours de ce voisinage.

La version la plus simple des méthodes de recherche locale est la méthode de descente.

II-4-2-1-1 la méthode de descente :

Cette méthode procède de manière itérative, en choisissant à chaque itération un point dans le voisinage de la solution courante. S'il est meilleur c'est à dire il améliore la solution courante, il devient la nouvelle solution courante, sinon un autre point est choisi, et ainsi de suite c'est à dire que ce procédé est répété aussi longtemps que la valeur de la fonction objectif diminue. La recherche s'interrompe dès lors qu'un minimum local de f est atteint.

**L'algorithme suivant présente le squelette de la méthode de descente.**

```
Initialisation
    choisir une solution admissible initiale s ;
    poser s* := s ;
Processus itératif
    tant que le critère d'arrêt n'est pas satisfait faire
        générer N(s) ;
        déterminer  s' ∈ N(s)  telle que
        f(s') = min f(s'') ;
                s'' ∈ N(s)
        s := s' ;
        si f(s) < f(s*) alors s* := s ;
        sinon le critère d'arrêt est satisfait
```

[3]Mouvement : opération élémentaire permettant de passer d'une configuration A à une configuration A' voisine de A.

[4]Mécanisme d'exploration : procédure qui précise comment passer d'une configuration A à une autre configuration A' appartenant au voisinage de A.

Plusieurs versions de la méthode de descente ont été proposées :

- descente aléatoire : la nouvelle solution est choisie aléatoirement,
- descente déterministe : le meilleur voisin est choisi,
- descente vers le premier meilleur : le premier voisin meilleur que la solution courante est choisi.

Historiquement, les méthodes de descente ont toujours compté parmi les méthodes heuristiques les plus populaires pour traiter les problèmes d'optimisation combinatoire. Toutefois elles comportent deux obstacles majeurs qui limitent considérablement leur efficacité :

- suivant la taille et la structure du voisinage N(s) considéré, la recherche de la meilleure solution voisine est un problème qui peut être aussi difficile que le problème initial ;

- une méthode de descente est incapable de progresser au-delà du premier minimum local rencontré car elle reste bloquée dans un minimum local dès qu'elle en rencontre un. Or les problèmes d'optimisation combinatoire comportent typiquement de nombreux optima locaux pour lesquels la valeur de la fonction objectif peut être fort éloignée de la valeur optimale.

Pour faire face à ces carences, d'autres méthodes de recherche locale plus sophistiquées ont été développés. Ces méthodes acceptent des solutions voisines moins bonnes que la solution courante afin d'échapper aux minima locaux de la fonction f. les différences principales entre ces méthodes se situent au niveau du choix de la solution voisine et au niveau du critère d'arrêt. Elles sont plus performantes q'une simple méthode de descente mais également beaucoup plus coûteuses en terme de ressources informatiques. Aussi, il faut tenir compte qu'un effort non négligeable est nécessaire pour ajuster convenablement les paramètres qu'elles font intervenir dans le but de guider efficacement la recherche.

#### II-4-2-1-2 La méthode du recuit simulé (Simulated Annealing):

La méthode du recuit simulé a été introduite en 1983 par trois chercheurs de la société IBM ( S. Kirkpatrick, C.D. Gelatt et M.P. V.ecchi) [KIR83]. Cette méthode tire son origine de la physique. Son principe de fonctionnement repose sur une imitation du phénomène de recuit en science des matériaux. Ce processus utilisé en métallurgie pour améliorer la qualité d'un solide, cherche un état d'énergie minimale qui correspond à une structure stable de ce métal. L'état optimal correspondrait à une structure moléculaire régulière parfaite. En partant d'une haute température où le métal serait

liquide, on refroidit le métal progressivement en tentant de trouver le meilleur équilibre thermodynamique. Chaque niveau de température est maintenu jusqu'à obtention d'un équilibre.

Les origines du recuit simulé remontent aux expériences réalisées par Metropolis dans les années 50 pour simuler l'évolution d'un tel processus de recuit physique. Metropolis et son équipe utilisent une méthode stochastique pour générer une suite d'états successifs du système en partant d'un état initial donné. Tout nouvel état est obtenu en faisant subir un déplacement (une perturbation) aléatoire à un atome quelconque. Soit $\Delta E$ la différence d'énergie occasionnée par une telle perturbation. Le nouvel état est accepté si l'énergie du système diminue ($\Delta E \leq 0$). Sinon, il est accepté avec une probabilité définie par : $p(\Delta E,T) = \exp(-\Delta E/(C_b \times T))$ où T est la température du système et $C_b$ une constante physique connue sous le nom de constante de Boltzmann.

A chaque étape, l'acceptation ou non d'un nouvel état dont l'énergie est supérieure à celle de l'état courant est déterminée de manière probabiliste : un réel $0 \leq \theta < 1$ est tiré aléatoirement et ensuite comparé avec $p(\Delta E,T)$. Si $\theta \leq p(\Delta E,T)$, alors le nouvel état est accepté pour remplacer l'état courant, sinon, l'état courant est maintenu. Après un grand nombre de perturbations, un tel processus fait évoluer le système vers un état d'équilibre thermodynamique selon la distribution de Boltzmann qui est définie par la probabilité de se trouver dans un état d'énergie E : $Pr(E) = c(T) \times \exp(-E/C_b \times T)$ où c(T) est un facteur de normalisation.

L'utilisation d'un tel processus du recuit simulé pour résoudre des problèmes d'optimisation peut être vu comme une version étendue de la méthode de descente. L'amélioration principale consiste en la possibilité d'accepter des mouvements qui dégradent la fonction de coût. donc Le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coût plus faible tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût. A chaque nouvelle itération, un voisin $s' \in N(s)$ de la configuration courante s est généré de manière aléatoire. Selon les cas, ce voisin sera soit retenu pour remplacer celle-ci, soit rejeté. Si ce voisin est de performance supérieure ou égale à celle de la configuration courante, c'est à dire $f(s') \leq f(s)$, il est systématiquement retenu. Dans le cas contraire, s' est acceptée avec une probabilité $p(\Delta f,T)$ qui dépend de deux facteurs : d'une part l'importance de la dégradation $\Delta f = f(s') - f(s)$ ( les dégradations plus faibles sont plus facilement acceptées), d'autre part un paramètre de contrôle T, la température ( une température élevée correspond à une probabilité plus grande d'accepter des dégradations). La température est contrôlée par une fonction décroissante qui définit un schéma de refroidissement. Les deux paramètres de la méthode définissent la longueur des paliers et la fonction permettant de calculer la suite décroissante des températures. En pratique, l'algorithme

s'arrête et retourne la meilleure configuration trouvée lorsque aucune configuration voisine n'a été acceptée pendant un certain nombre d'itération à une température ou lorsque la température atteint la valeur zéro.

La performance du recuit simulé dépend largement du schéma de refroidissement utilisé. De nombreux schémas théoriques et pratiques ont été proposés[COL88]. De manière générale, les schémas de refroidissement connus peuvent être classés en trois catégories :

- réduction par palier : chaque température est maintenue égale pendant un certain nombre d'itération, et décroît ainsi par paliers
- réduction continue : la température est modifiée à chaque itération.
- Réduction non-monotone : la température décroît à chaque itération avec des augmentations occasionnelles.

Le schéma de refroidissement de la température est donc une des parties les plus difficiles à régler dans ce cas. Ces schémas sont cruciaux pour l'obtention d'une implémentation efficace.

II-4-2-1-3 Les méthodes d'acceptation à seuil (Threshold algorithms) :

Les méthodes d'acceptation à seuil [DUE90] sont des méthodes de recherche locale qui dérivent directement de l'algorithme du recuit simulé. La différence essentielle entre ces deux méthodes se situe au niveau de l'acceptation d'une solution de moins bonne qualité à chaque étape. Dans une méthode d'acceptation à seuil, une telle décision est prise de manière déterministe, sans avoir recours aux principes du recuit thermodynamique. A chaque itération k, l'acceptation d'un voisin s' se base uniquement sur une fonction auxiliaire r(s,s') et un seuil $T_k$ : s' est accepté si r(s',s) < k. dans le cas le plus simple r(s',s) = $\Delta f$= f(s')-f(s). le seuil $T_k$ a la même fonction que la température dans le cas du recuit simulé. Il est initié à une valeur élevée puis décroît progressivement après certain nombre (variable) d'itérations. Les seuils forment une suite de valeurs positives décroissantes $T_1 \geq T2 \geq \ldots T_{k-1} \geq Tk \geq 0$ et $\lim T_k \rightarrow 0$ afin de diminuer au cours du temps la possibilté d'accepter une configuration qui dégrade la fonction de coût. La difficulté essentielle de cette approche se situe au niveau de la détermination des seuils pour une application donnée.

II-4-2-1-4 Les méthodes de bruitage :

La méthode de bruitage s'applique à des problèmes dont les configurations portent sur des réels. Elle fait appel à une notion de bruitage de la donnée qui est

effectuée en ajoutant à chaque réel de la donnée initiale une composante calculée comme le produit de trois termes :

1- une fonction aléatoire à valeur sur l'intervalle [0,1],
2- un paramètre qui contrôle le niveau de bruit,
3- le plus grand des réels concernés, dans le but de normaliser le niveau de bruit par rapport à la donnée.

A chaque étape, il est effectué une descente par rapport à la donnée bruitée et le niveau de bruit est progressivement décrémenté. Il existe des variantes pour cette méthode, par exemple il est possible d'effectuer à chaque étape une descente sur la donnée non bruitée et de sélectionner le meilleur candidat, il est aussi possible de remplacer régulièrement la configuration courante par la meilleure trouvée depuis l'initialisation de la méthode. L'utilisation du bruitage permet à la recherche de ne pas rester bloquer sur un minimum local.

II-4-2-1-5 La méthode tabou :

La méthode tabou a été développée par Glover [GLO89,GLO90]. Pour certains chercheurs cette méthode apparaît plus satisfaisante sur le plan scientifique que le recuit simulé, car la partie « aléatoire » de la méthode a disparu.

La méthode tabou partage avec l'algorithme du recuit simulé l'idée de guider la recherche local pour éviter les optimums locaux. Contrairement au recuit simulé qui génère de manière aléatoire une seule solution voisine s' dans le voisinage N(s) à chaque itération, la méthode tabou examine un échantillonnage de solutions N(s) et retient la meilleure s' même si s' est plus mauvaise que s . La recherche tabou ne s'arrête donc pas au premier optimum trouvé. Cependant cette stratégie peut entraîner des cycles, par exemple un cycle de longueur 2 : s $\longrightarrow$ s' $\longrightarrow$ s' $\longrightarrow$ s'….. pour empêcher ce type de cycle, on mémorise les k dernières configurations visitées dans une mémoire à court terme et on interdit tout mouvement qui conduit à une de ces configurations. Cette mémoire est appelée la liste tabou (qui a donné le nom de la méthode), une des composants essentielle de cette méthode. Elle permet d'éviter tous les cycles de longueur inférieure ou égale à k. La valeur de k dépend du problème à résoudre et peut éventuellement évoluer au cours de la recherche. Donc il y a une modification temporaire de la structure de voisinage de la solution s permettant de quitter des optima locaux. Le voisinage N*(s) intégrant ces modifications de structure est régit par l'utilisation de structures de mémoire spécifiques : mémoire à court terme ou mémoire à long terme.

Comme on vu précédemment, la mémoire à court terme correspond à la mise en place d'une liste tabou qui contient les quelques dernières solutions qui ont été récemment visitées. Le nouveau voisinage N*(s) exclut donc toutes les solutions de la liste tabou. Lorsque la structure de donnée correspondant aux solutions est trop complexe ou occupe une grand place mémoire, il est courant de ne garder dans la liste tabou que des informations soit sur les caractéristiques des solutions, soit sur les mouvements. En conservant des caractéristiques des solutions ou des mouvements, il est possibles alors qu'une solution de bien meilleure qualité ait un statut tabou. Accepter tout de même cette solution revient à outrepasser son statut tabou, c'est l'application du critère d'aspiration. Ce mécanisme particulier est mis en place afin de lever le statut tabou d'une configuration, sans pour autant introduire un risque de cycles dans le processus de recherche. La fonction d'aspiration peut être définie de plusieurs manières. La fonction la plus simple consiste à révoquer le statut tabou d'un mouvement si ce dernier permet d'atteindre une solution de qualité supérieure à celle de la meilleure solution trouvée jusqu'alors.

La mémoire à long terme permet d'une part d'éviter de rester dans une seule région de l'espace de recherche et d'autre part d'étendre la recherche vers des zones plus intéressantes. Par exemple la mémoire à base de fréquence (la frequency-based memory) attribue des pénalités à des caractéristiques des solutions plusieurs fois visitées au cours de la recherche. Cette technique simple permet donc de diversifier la recherche facilement. Par ailleurs, les mouvements ayant conduit a des bonnes solutions peuvent être aussi encouragés. Par exemple garder en mémoire une liste de solutions élites qui seront utilisées comme nouveau point de départ quand la recherche deviendra improductive pendant plusieurs itérations consécutives. Autrement dit la méthode tabou est améliorée par deux techniques intéressantes : l'intensification et la diversification qui se basent toutes les deux sur l'utilisation de la mémoire à long terme et qui se différencient simplement selon la façon d'exploiter les informations de cette mémoire.

L'intensification se fonde sur l'idée d' apprentissage de propriétés favorables : les propriétés communes souvent rencontrées dans les meilleures configurations visitées sont mémorisées au cours de la recherche, puis favorisées pendant la période d'intensification. Une autre manière d'appliquer l'intensification consiste à mémoriser une liste de solutions de bonne qualité et à retourner vers une de ces solutions.

La diversification a pour objectif inverse de l'intensification : elle cherche à diriger la recherche vers des zones inexplorées. Sa mise en œuvre consiste à modifier temporairement la fonction de coût pour favoriser des mouvement n'ayant pas été effectués ou à pénaliser les mouvements ayant été souvent répétés. L'intensification et la

diversification jouent donc un rôle complémentaire.

Plusieurs améliorations de la recherche tabou ont été proposées dans la littérature, on peut citer :

- la recherche tabou robuste[TAI91], qui utilise une liste tabou de longueur variable et aléatoire, et une forme de mémoire à long terme.
- La recherche tabou réactive[BAT94], dont l'idée principale est d'accroître la longueur de la liste tabou si le nombre de solutions revisitées est élevé, et de réduire la longueur de la liste si ce nombre est faible. Une autre particularité de la recherche tabou réactive est qu'elle effectue une série de déplacements aléatoires si elle se trouve bloquée dans une région de l'espace de recherche.

II-4-2-1-6 Autres méthodes à base de voisinage :

D'autres métaheuristiques à base de voisinage existent :

- la recherche par voisinage variable [MLA97] est une méthode récente, basée sur la performance des méthodes de descente. Introduite par Mladenovic et Hansen , la méthode propose simplement d'utiliser plusieurs voisinages successifs quand on se trouve bloqué dans un minimum.

Avant tout, il est nécessaire de définir un ensemble de $k_{max}$ voisinages, dénotés par $N_{k=1... kmax}$ ( et de préférence tels que $N_k \subset N_{k+1}$). On choisit une solution de départ s par heuristique. Ensuite, à partir d'une solution initiale s' choisie dans le premier voisinage N(s) de s, on applique une méthode de descente (ou une autre méthode de recherche locale) jusqu'à arriver dans un minimum local (ou que la recherche locale s'arrête). Si la solution trouvée s″ est meilleure que s alors on recentre la recherche en repartant du premier voisinage, sinon on passe au voisinage suivant (qui a priori est plus grand). La recherche s'arrête quand tous les voisinages ne sont plus capables d'améliorer la solution.

Le point crucial dans une recherche à voisinage variable, c'est bien évidemment la constitution des voisinages de plus en plus grands et inclus les uns dans les autres. Mais une bonne structure de voisinage conduit généralement à de bons résultats ou au moins à une recherche efficace.

Dans cette méthode, la diversification est gérée par deux techniques. La première consiste à choisir dans le voisinage courant une solution aléatoirement et la seconde c'est le changement de voisinage lui-même qui agrandit l'espace de recherche autorisé et donc agrandit le voisinage exploré. L'intensification de la recherche est

effectuée par l'appel à une procédure de recherche locale.

- la recherche locale guidée [VOU95] est une variante assez élaborée d'une méthode de descente classique. Le principe de base repose sur une modification dynamique de la fonction objectif, dans le but de pénaliser l'optimum local courant c'est à dire elle consiste à modifier la fonction à optimiser en ajoutant des pénalités. La recherche locale est appliquées alors sur cette fonction modifiée. La solution trouvée qui se trouve être un optimum local sert à calculer les nouvelles pénalités. Pour cela l'utilité de chacun des attributs de la solution est calculée et les pénalités associées aux attributs de valeurs maximales sont augmentées. Ces étapes successives sont répétées jusqu'à ce qu'un critère d'arrêt soit validé.

L'intensification de la recherche locale guidée est faite directement par l'appel à la recherche locale au sein de son algorithme. Les expressions d'utilités des différents attributs d'une solution servent, pour celles qui sont à leur maximum, à pénaliser la fonction de coût. Ce qui veut dire que la recherche va être forcée à s'effectuer dans d'autres directions que celles qui semblaient prometteuses lors des précédentes recherches. Cette mise à jour des fonctions de pénalité est donc un facteur de diversification de la recherche.

- GRASP ( Greedy Randomised Adaptive Search Procedure) est une méthode introduite par Feo et Resende [FEO95] . Elle est basée sur le principe qui combine une heuristique gloutonne et une recherche aléatoire. A chaque itération, une solution est construite comme dans une heuristique gloutonne ( en se servant d'une liste d'attributs comme liste de priorité). Cette solution est améliorée par l'intermédiaire d'une méthode de descente. En se basant sur la qualité générale de la solution ainsi obtenue, l'ordre de la liste des attributs est mis à jour et le processus est itéré jusqu'à satisfaction d'un critère d'arrêt.

L'aspect d'intensification dans cette méthode est obtenu par l'application de la recherche local au sein de son algorithme. Mais la mise à jour de la liste des attributs est aussi un facteur important permettant de contrôler la convergence de la méthode et donc de son intensification. L'aspect de diversification est assuré par l'utilisation de l'algorithme glouton qui en effectuant un choix aveugle à chaque étape de la construction d'une solution, il peut s'éloigner rapidement d'une bonne solution très proche (c'est donc un facteur de diversification dans cette méthode).

- la recherche locale réitérée [LOU02] Un inconvénient de la recherche locale répétée est qu'elle perd toute l'information sur la recherche passée, à chaque fois qu'une nouvelle recherche locale est démarrée. Plusieurs autres approches ont alors été

proposées, permettant de tirer profit de l'historique de la recherche pour initialiser la recherche locale. Parmi ces méthodes, on retrouve la recherche locale réitérée, dont l'idée de base consiste à utiliser les optimums locaux trouvés au cours des recherches locales précédentes, pour construire le point initial qui sera utilisé pour démarrer la suivante.

II-4-2-1-7 Emploi du temps et méthodes de voisinage :

Les problèmes d'emploi du temps peuvent être traiter par les méthodes de voisinage, pour cela de nombreux travaux ont été consacrés à la résolution de ce type de problème :

- Hertz [HER91] décompose le problème de l'emploi du temps en deux sous problèmes. Le premier consiste à planifier les cours de façon à prévenir les conflits sur les salles et les enseignants. Le deuxième, revient à regrouper les étudiants au sein des sections de cours. L'approche tente de prendre en compte l'ensemble de toutes les contraintes possibles (cours successifs, cours distancés,…).

  Dans cette approche les deux sous problèmes sont considérés en tant que problèmes d'assignement. Pour le problème de planification des cours il s'agit d'assigner à chaque cours une date de début. Dans le problème de regroupement des étudiants, il s'agit d'assigner à chaque étudiant une section de cours convenable.

- Dans l'approche de Costa[COS94], les regroupements des étudiants dans les classes est préfixé. Le problème consiste alors à assigner une période de début à chaque cours.

  - Pour générer un emploi du temps faisable les cours sont triés selon leur recevabilité. La recevabilité d'un cours correspond au nombre de périodes en lesquelles il peut être programmé en considérant les contraintes de préassignement, de conflit sur les enseignant et sur les salles. Les cours sont alors programmer l'un après l'autre à commencer par le cours le moins recevable. Le cours est affecté à la période générant le moins de conflits.

  - Le voisinage d'un emploi du temps T englobe tous les emplois du temps faisables qui peuvent être engendrés en changeant la période de début d'un seul cours.

§ Pour prévenir les cycles lors de la recherche, le couple (x, $t_1$) est introduit dans la liste tabou lorsque la recherche tabou fait passer la recherche de l'emploi du temps $T_1$ à $T_2$ en modifiant la période du cours x de $t_1$ vers $t_2$.

Ø Schaerf[SCH95] a utilisé une méthode tabou pour résoudre ce problème. Il emploi le codage matriciel $M_{j,k}$ qui contient le nom de la classe du professeur j à la période k. les voisinages proposés sont :

- échanger deux cours pour un même professeur
- déplacer un cours à une autre période

pour des instances de taille moyenne, les résultats ont été encourageant.

Ø Dans [ABR91], l'approche proposée est un algorithme à base de recuit simulé. L'emploi du temps est représenté par un vecteur de listes, chaque liste désigne l'ensemble des cours affectés à une même période. L'emploi du temps est évalué suivant une formule proposée par l'auteur exprimant qu'un enseignant ou un classe ne doit pas apparaître plus d'une fois en même temps et que le nombre de cours planifiés à la même période ne doit pas excéder le nombre de locaux disponible. Les atomes sont remplacés par les unités d'enseignement. L'énergie du système est remplacée par le coût du planning. Initialement, une allocation est faite dans laquelle les enseignants sont placés dans des périodes, locaux et professeurs choisis aléatoirement. Le coût initial et la température sont alors calculés, le coût est utilisé pour refléter la qualité du planning, juste comme l'énergie du système reflète la qualité de la substance étant refroidie. La température est utilisée pour contrôler la probabilité d'une augmentation en coût.

D'autres méthodes ont été utilisées comme la recherche local guidée et la méthode tabou avec une mémoire à long terme [WHI01,BUR98, SCH96].

II-4-2-2 Les méthodes à base de population :

Les sciences de la vie et les processus naturels ont de tout temps fasciné les ingénieurs. Ces derniers n'hésitent pas à s'inspirer des structures et des mécanismes du monde vivant pour développer des objets artificiels utilisables dans des contextes variés. Dans le domaine de l'optimisation combinatoire, la complexité des phénomènes naturels a servi de modèle pour des algorithmes toujours plus sophistiqués constituant ainsi la base d'un nouveau champ de la programmation informatique en pleine effervescence. On peut distinguer deux grandes classes de techniques :

- les algorithmes évolutinnaires qui sont inspirés par des concepts issus de la

théorie de l'évolution naturelle de Darwin, et

- les algorithmes de colonies de fourmis qui sont inspirés de l'éthologie.

Ces deux techniques appartiennent donc à la classe des méthodes à base de population.

Les méthodes à base de population comme leur nom l'indique, travaillent sur une population de solutions et non pas sur une solution unique comme dans les méthodes à base de voisinage.

II-4-2-2-1 Les algorithmes évolutionnaires:

Les algorithmes évolutionnaires sont des techniques d'optimisation itérative, inspirées donc de la théorie de l'évolution de Darwin. Un algorithme évolutionnaire simule un processus d'évolution sur une population d'individus, dans le but de faire évoluer vers les optimums globaux du problème d'optimisation considéré.

Un algorithme évolutinnaire typique réunit trois composants :

1- Une population constituée de plusieurs individus représentant des solutions potentielles du problème posé (en optimisation combinatoire classique : configuration du problème).

2- Une fonction d'adaptation (fitness) qui évalue la performance d'un individu par rapport au milieu ( en optimisation combinatoire : fonction du coût).

3- Un mécanisme d'évolution de la population composé de plusieurs opérateurs de modification et de sélection permettant (grâce à ces opérateurs prédéfinis), d'éliminer certains individus et d'en créer de nouveaux.

Le cycle d'évolution d'un algorithme évolutionnaire typique débute avec une population initiale (généralement obtenue de façon aléatoire) puis répète la boucle suivante (qui présente ce cycle d'évolution) :

1- mesurer l'adaptation (la qualité) de chaque individu de la population par le mécanisme d'évaluation,

2- sélectionner une partie des individus,

3- produire de nouveaux individus par recombinaisons des individus sélectionnés.

Ce processus se termine quand la condition d'arrêt est vérifiée, par exemple quand un

nombre maximum de cycles ou un nombre maximum d'évaluations est atteint.

Les différences principales entre les algorithmes évolutionnaires, et les méthodes classiques d'optimisation combinatoire sont les phases de sélection et d'évolution. La sélection permet de choisir les meilleurs individus et c'est à partir d'eux que l'on construit la génération suivante. L'évolution quant à elle repose sur deux principaux opérateurs, la recombinaison qui combine plusieurs individus parents pour créer des individus enfants pour la génération suivante et la mutation qui altère légèrement certains individus.

D'une manière générale, on peut distinguer trois grandes classes d'algorithmes évolutionnaires :les algorithmes génétiques, la programmation évolutive et les stratégies d'évolution. Ces méthodes se différencient par leur manière de représenter l'information et par leur façon de faire évoluer la population d'une génération à l'autre.

II-4-2-2-1-1 Les algorithmes génétiques:

Les algorithmes génétiques sont des algorithmes d'optimisation développés dans les années 70 avec le travail de Holland [HOL75] puis approfondis par Goldberg qui a largement contribuer à les vulgariser [GOL89]. Les algorithmes génétiques sont la branche des algorithmes évolutionnaires la plus connue et la plus utilisée. La particularité de ces algorithmes est qu'il font évoluer des populations d'individus codés par des chaînes de longueur fixe ( codage classique utilisé à l'origine : des chaînes de bits (0/1) ) tout en utilisant des opérateurs d'évolution génétique de mutation et de croisement. Les individus codés de cette manière sont appelés chromosomes. Les opérateurs agissent sur un ou plusieurs individus sans connaissance sur ce qu'ils manipulent (ni sur le problème).

Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Pour l'utiliser adéquatement, on doit disposer des six éléments suivants : (étapes préalables à la programmation).

1- Un principe de codage de l'élément de population (individu). Cette étape associe à chacun des points de l'espace d'état une structure de donnée. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été les premiers à être utilisés. Actuellement, on se sert plus souvent de codages directs (réels et entiers,…).

2- Un mécanisme de génération de la population initiale qui servira de base pour la

génération suivante. Le choix de la population initiale est important car il peut rendre plus au moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout l'espace de recherche. En pratique, on a souvent recours à la génération aléatoire de la population initiale.

3- Une fonction d'évaluation afin de mesurer les performances de chaque individu. Elle constitue le critère à base duquel l'individu sera ou non sélectionné pour être reproduit dans la génération suivante. Il faut savoir que la qualité de cette fonction conditionne, pour une grande part, l'efficacité de l'algorithme génétique.

4- Un mécanisme de sélection des individus candidats à l'évolution.

5- Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche. L'opérateur de croisement recompose les gènes d'individus existants dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace de recherche.

6- Des paramètres de dimensionnement : taille de la population, nombre total de génération ou critère d'arrêt, probabilités d'applications des opérateurs génétiques de croisement et de mutation.

Le principe général du fonctionnement d'un algorithme génétique standard est représenté sur la figure 2.2.

.

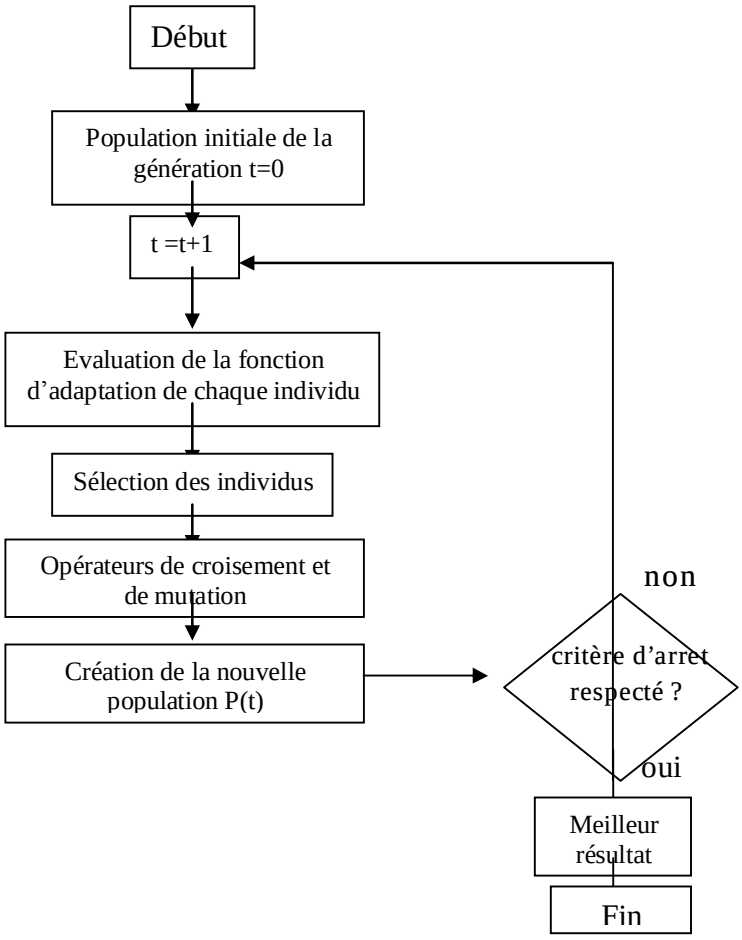



Fig. 2.2 : Principe général d'un algorithme génétique standard

L'opérateur de croisement est appliqué avec une probabilité $P_c$ (généralement autour de 0.6) L'opérateur de mutation est appliqué avec la probabilité $P_m$ qui est très inférieur à $P_c$ ( généralement entre 0.005 et 0.01).

Différents critères d'arrêt de l'algorithme peuvent être choisis :

- le nombre de génération que l'on souhaite exécuter peut être fixé à priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.
- L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement.

L'ensemble des points précédent peut être détaillé comme suit :

1. le codage

les algorithmes génétiques travaillent sur des codages et non sur des solutions

réelles. Ces codes sont appelés chromosomes. Il faut définir et coder convenablement le problème afin de cerner l'espace des solutions possibles. Ce codage doit, de plus, être aussi compact que possible pour permettre une évolution rapide.

Historiquement le codage utilisé par les algorithmes génétiques était le codage binaire. Cependant, ce type de codage n'est pas toujours bon pour certains problèmes d'optimisation pour cela d'autres types de codage sont utilisés, le codage entier, le codage réel …etc.

2. la genèse de la population

Si la position de l'optimum dans l'espace de recherche est totalement inconnue, le choix de la population initiale se fait d'une manière aléatoire. Si par contre, des informations à priori sur le problèmes sont disponibles, il paraît bien évidemment naturel de générer les individus dans un sous-espace particulier afin d'accélérer la convergence. Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités et lorsque c'est possible ne générer que des éléments de la population respectant les contraintes.

3. la sélection

la sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. Un individu ayant une forte valeur d'adaptation a alors plus de chances d'être sélectionné qu'un individu mal adapté à l'environnement (ceci va permettre de favoriser la reproduction des « bons » individus).

La sélection ne crée aucune nouveauté, elle se contente de choisir parmi la population mère quelles seront ou ne seront pas en mesure de contribuer à la création de la population fille, suivant une stratégie particulière.

Cependant, la sélection est relativement délicate à manipuler. Un excès de sélection peut entraîner une perte de la diversité au sein de la population qui bloque ainsi la résolution du problème donné, en convergeant par exemple vers des optima locaux. Une sélection trop faible pose la difficulté inverse : la non convergence de l'algorithme. Pour cela on distingue deux techniques de sélection : la sélection stochastique et la sélection déterministe.

a) <u>la sélection stochastique</u>

Cette technique permet de favoriser les meilleurs individus mais de manière stochastique, ce qui laisse la chance aux individus moins performants d'être eux aussi sélectionnés. Par contre, il peut arriver que le meilleur individu ne soit pas sélectionné.

On distingue trois types de stratégie : ***La sélection proportionnelle, La sélection proportionnelle avec reste stochastique et la sélection par tournois.***

1- ***<u>La sélection proportionnelle</u>***

Dans ce cas chacun a une chance d'être sélectionné en fonction de son efficacité. La méthode roulette wheel selection (RWS), ou loterie, est la plus classique de ce mode. La première étape consiste à attribuer à chaque individu i une probabilité $P_i$ de sélection proportionnelle à son fitness $f_i$ et à la somme des fitness de tous les individus de la population.

Si N est le nombre d'individus de la population alors :

$$P_i = f_i / \sum_{i=1}^{N} f_i$$

La deuxième étape détermine les N individus de la population fille tirés au hasard en fonction de ces probabilités. Le tirage s'effectue généralement avec remise offrant la possibilité à un excellent individu de se retrouver plusieurs fois dans la population finale. Cette sélection permet donc à chacun des individus de la population d'avoir une chance aussi minime soit-elle, d'être sélectionné.

2- ***<u>La sélection proportionnelle avec reste stochastique</u>***

Ce mode de sélection tire profit du précédent auquel est ajouté un aspect plus déterministe. L'ensemble des $P_i$ défini précédemment, est conservé et intervient dans l'équation qui détermine le nombre d'occurrences de la chaîne i reproduite dans la population fille. Le reste stochastique $R_i$ est alors défini en fonction de $N_i$.

$$N_i = E(N^* P_i) \qquad R_i = (N^* P_i) - N_i$$

$$S = N - \sum_{i=1}^{N} N_i \qquad S_i = R_i / S$$

Les $N_i$ représentent l'ensemble des chromosomes reproduit de manière déterministe. Généralement la somme des $N_i$ est inférieure à N. il reste donc un certain nombre de chaînes à sélectionner défini par S. A partir des restes $R_i$ et du nombre de chaînes S, les probabilités $S_i$ (pour chaque individu, d'être sélectionné par le reste stochastique) sont calculées.

Pour obtenir les dernières chaînes sélectionnées, une sélection proportionnelle est effectuée S fois en fonction des probabilités $S_i$.

Cette sélection offre, comme la précédente, une chance à toutes les chaînes, mais impose tout de même la présence des meilleurs parmi la population fille. Elle évite donc une disparition prématurée des bonnes chaînes due à un tirage aléatoire défavorable, surtout lorsque les populations de chromosomes sont de faibles tailles.

3- ***la sélection par tournois***

La sélection par tournois est une alternative aux techniques de sélection proportionnelle. Le tournoi le plus simple consiste à choisir aléatoirement un certain nombre d'individus dans la population et à sélectionner pour la reproduction celui qui a la plus grande adaptation. Au cours d'une génération, il y'a autant de tournois que d'individus à remplacer. Les individus qui participent à un tournois restent dans la population et sont de nouveau disponibles pour les tournois ultérieurs. La variance de ce processus est également élevée. La pression de sélection est ajustée par le nombre de participants à un tournoi. Choisir de nombreux participants conduit à une forte pression de sélection car un individu moyen ou faible aura moins de chance d'être sélectionné.

Une méthode dérivée fait intervenir un tournoi après l'évaluation des chaînes déjà recombinées. Chaque couple d'enfants entre alors dans un tournoi avec ses parents respectifs afin de conserver les deux meilleurs individus des quatre en combinaison.

b) la sélection déterministe

On sélectionne les meilleurs individus ( au sens de la fonction d'adaptation), cela suppose un tri de l'ensemble de la population. Les individus les moins performants sont éliminés de la population, et le meilleur individu est toujours sélectionné, on parle alors d'élitisme.

La sélection élitiste

La stratégie élitiste est utilisée en plus des méthodes précédentes et non pas séparément. Elle consiste à conserver dans la population, d'une génération à l'autre, au moins l'individu ayant la meilleure adaptation. Les individus reproduits remplacent les individus les moins bons de la génération courante pour obtenir la nouvelle génération, préservant ainsi les meilleurs.

Une des variantes de la stratégie élitiste, impose la présence des X% meilleurs individus de la population initiale dans la population finale. Ainsi, ces chaînes sont tout simplement protégées d'un quelconque dérapage du hasard, elles sont automatiquement sélectionnées. Ce type de variante provoque donc une amélioration constante des performances de la population puisque le ou les meilleurs individus sont toujours conservés d'un population à l'autre.

4. le croisement

Traditionnellement, le croisement est vu comme l'opérateur de recherche essentiel d'un algorithme génétique. Après la sélection, un croisement peut éventuellement avoir lieu. Dans ce cas, deux individus de la nouvelle génération échange une ou plusieurs partie(s) de leur génotype pour former deux individus différents de ceux d'origine.

Cet opérateur est essentiel, car il permet d'obtenir de nouveaux individus distincts de ceux déjà existants et donc d'explorer tout l'espace de recherche. Il existe différentes méthodes pour croiser deux chromosomes :

- le croisement ″à un point″ :

  Ce type de croisement est standard dans les algorithmes génétiques. Il consiste à choisir un emplacement aléatoirement sur une chaîne et d'intervertir tous les gènes d'un coté de ce point entre les deux chaînes.
- le croisement ″à n points″ :

  Le croisement à n points est une généralisation du croisement à un point avec n points de coupure sur les chaînes. Il s'agit alors de déterminer n points sur ces chaînes, puis d'échanger sur les chaînes les blocs entre ces points afin d'obtenir les enfants.

5. la mutation

Elle constitue une exploration totalement au hasard de l'espace de recherche et permet également d'éviter la perte du matériel génétique potentiellement utile. Mais, par rapport au croisement, son rôle reste secondaire. En général, on lui attribue une probabilité faible, de l'ordre de 0.01.

Cet opérateur consiste à apporter de l'innovation dans la population en modifiant un sel gène aléatoirement. Si toutes les chaînes possèdent une valeur identique sur le même gène, alors ni la sélection, ni le croisement ne pourront créer un individu pourvu d'une différence au niveau de ce gène. La mutation permet des variations de cet ordre, rendant possible la sortie d'un optimum local.

Bien que sa part dans les algorithmes génétiques soit moindre par rapport au croisement, son emploi est cependant indispensable. Dans un codage binaire muter un bit revient, tout simplement, à basculer sa valeur de 0 à 1 et inversement.

6. résumé de l'algorithme génétique de base :

[Début] : générer aléatoirement une population de n chromosomes.

[Evaluation] : évaluer l'adaptation (fitness) f(i) de chaque chromosome dans la population.

[Nouvelle génération] : créer une nouvelle population en répétant les étapes suivantes jusqu'à ce que la nouvelle population est complétée.

1. [Sélection] : Sélectionner des chromosomes parents de la population relativement à leurs fitness (meilleur fitness, une grande chance d'être sélectionné,…)
2. [Croisement]: Avec la probabilité du croisement fixée, croiser les parents pour former les nouveaux descendants. Si aucun croisement n'a eu lieu les descendants sont des copies exactes des parents.
3. [Mutation] : Avec la probabilité de mutation fixée, muter le nouveau descendant à chaque locus (position dans le chromosome).
4. [Acceptation] : Placer les nouveaux descendants dans la nouvelle population

[Remplacer] : Utiliser la nouvelle population générée pour l'exécution de l'algorithme de nouveau.

[Test] : Si la condition fin est satisfaite stop et retourner la meilleure solution dans la population courante.

[Boucler] : Aller à l'étape 2.

II-4-2-2-1-2 La programmation évolutive :

La programmation évolutive a été initialement introduite pour simuler l'intelligence qui est définie sur l'hypothèse suivante : la caractéristique principale de l'intelligence est la capacité d'adaptation comportementale d'un organisme à son environnement. Aujourd'hui la programmation évolutive s'est adaptée à l'optimisation combinatoire.

La programmation évolutive s'appuie sur un codage approprié du problème à résoudre et sur les opérations de mutation adaptées au codage. Le codage d'un tel algorithme dépend du problème à résoudre. Par exemple, pour un problème d'optimisation dans le domaine des réels, les individus d'une population seraient des vecteurs de réels.

Un cycle d'évolution typique pour la programmation évolutive est le suivant : chaque configuration de la population courante est copiée dans une nouvelle population. Les configurations sont ensuite mutées, conduisant à de nouvelles configurations. L'ensemble des configurations entre ensuite dans une étape de compétition pour survivre dans la génération suivante.

II-4-2-2-1-3 Les stratégies d'évolution:

Les stratégies d'évolution sont conçues dès le départ pour résoudre des problèmes d'optimisation continus. Dans ces algorithmes, les individus sont représentés par des points(vecteurs de réels), et les seuls mécanismes utilisés sont la mutation et la sélection.

Une façon très simple d'utiliser ces algorithmes est la technique (1+1)-ES ( 1 individu normal plus 1 individu mutant) qui manipule un seul individu. A chaque génération, l'algorithme réalise une mutation de l'individu pour obtenir un individu enfant, effectue une descente sur chacun d'entre eux puis sélectionne (conserve) le résultat le mieux adapté des deux. Les conditions d'arrêt sont souvent le nombre d'itération, le temps de calcul ou l'écart (au sens de la fonction de fitness) entre deux générations.

II-4-2-2-2 Les algorithmes de colonies de fourmis :

Les algorithmes de colonies de fourmis forment une classe des métaheuristiques récemment proposée pour des problèmes d'optimisation difficile par Dorigo [DOR96], Ces algorithmes s'inspirent des comportements collectifs de dépôt et de suivi de piste observés dans les colonies de fourmis. Une colonie d'agents simples (les fourmis) communiquent indirectement via des modifications dynamiques de leur environnement (les pistes de phéromones) et construisent ainsi une solution à un problème en s'appuyant sur leur expérience collective.

Les algorithmes de colonies de fourmis sont nés à la suite d'une constatation : les insectes sociaux en général, et les fourmis en particulier, résolvent naturellement des problèmes relativement complexe. Les biologistes ont étudié comment les fourmis arrivent à résoudre collectivement des problèmes trop complexes pour un seul individu, notamment les problèmes de choix lors de l'exploitation de source de nourriture.

Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées phéromones. Elles sont attirées par ces substances, qu'elles perçoivent grâce à des récepteurs situés dans leurs antennes. Les fourmis peuvent déposer des phéromones au sol, grâce à une glande située dans leur abdomen, et former ainsi des pistes odorantes, qui pourront être suivies par leurs congénères.

Les fourmis utilisent les pistes de phéromone pour marquer leur trajet, par exemple entre le nid et une source de nourriture. Une colonie est ainsi capable de choisir le plus court chemin vers une source à exploiter, sans que les individus aient une vision globale du trajet.

En effet, les fourmis le plus rapidement arrivées au nid, après avoir visité la source de nourriture, sont celles qui empruntent le chemin le plus court car la quantité de phéromone présente sur le plus court trajet est légèrement plus importante que celle présente sur le chemin le plus long. Ainsi, une piste présentant une plus grande concentration en phéromone est plus attirante pour les fourmis, elle a une probabilité plus grande d'être empruntée. La piste courte va alors être plus renforcée que la longue, et, à terme, sera choisie par la grande majorité des fourmis.

Les principes de fonctionnement de la métaheuristique de colonie de fourmis sont comme suit : le problème est représenté par un jeu de solutions, une fonction objectif assignant une valeur à chaque solution et un jeu de contraintes. L'objectif est de trouver l'optimum global de la fonction objectif satisfaisant les contraintes. Les différents états du problèmes sont caractérisés comme une séquence de composants.

Les fourmis construisent des solutions en se déplaçant sur un graphe G=(C,L), où les nœuds sont les composants de C et où l'ensemble L connecte les composants de C. les contraintes du problème sont implémentées directement dans les règles de déplacement des fourmis ( soit en empêchant les mouvements qui violent les contraintes, soit en pénalisant de telles solutions).

Les fourmis artificielles peuvent être caractérisées comme une procédure de construction stochastique construisant des solutions sur le graphe G=(C,L). chaque fourmi dispose d'une mémoire utilisée pour stocker le trajet effectué, d'un état initial et de conditions d'arrêt. Les fourmis se déplacent d'après une règle de décision probabiliste fonction des pistes de phéromone locales, de l'état de la fourmi et des contraintes du problème. En plus des règle régissant le comportement des fourmis, un autre processus majeur est pris en compte : l'évaporation des pistes de phéromone. En effet, à chaque itération, la valeur des pistes de phéromone est diminuée. Le but de cette diminution est d'éviter une convergence trop rapide et le piégeage de l'algorithme dans des minimums locaux et favorisant l'exploration de nouvelles régions.

II-4-2-2-3 Autres méthodes à base de population :

D'autres méthodes à base de population existent :

- la programmation génétique qui est apparue initialement comme sous-domaine des algorithmes génétiques, mais actuellement elle tend à être considérée comme une branche à part entière.

- La recherche par dispersion ( scatter search)

Cette méthode a été proposé par Glover[GLO94], la recherche par dispersion fait évoluer une population de solutions et s'appuie sur le principe suivant : une population de solutions (assez importante au départ) est générée (en essayant de proposer des solutions diverses les unes des autres) et chaque individu est amélioré par l'application d'une recherche locale. De cette population on extrait un ensemble de référence contenant les meilleures solutions de la population initiale. Ensuite ces solutions sont combinées entre elles ( et avec les nouvelles solutions générées) puis améliorées jusqu'à ce qu'il n'y ait plus de nouvelles solutions générées par combinaison. Ensuite la moitié de la population est régénérée (remplacée par des solutions diverses) et le processus recommence jusqu'à satisfaction d'un critère d'arrêt.

- Les algorithmes mémétiques[6]

Les algorithmes mémétiques ont été proposés pour la première fois par Moscato[MOS99]. ils s'inspirent de certains modèles d'adaptation dans la nature, qui combinent l'évolution adaptative de populations d'individus avec l'apprentissage des individus au cours de leur vie.

Du point de vue optimisation, les algorithmes mémétiques sont des extensions des algorithmes évolutionnaires utilisant des méthodes de voisinage pour améliorer leurs individus. Cette méthode, bien que très puissante, souffre d'un grand défaut : les temps de calcul peuvent devenir prohibitifs lors de l'utilisation de population de grand taille.

- Les algorithmes basés sur un essaim de particules

L'optimisation par essaim de particules est une technique d'optimisation globale stochastique développée par Kennedy en 1995[KEN95], en s'inspirant du comportement social des individus qui ont tendance à imiter les comportements réussis qu'ils observent dans leur entourage, tout en y apportant leurs variations personnelles.

L'optimisation par essaim de particules utilise une population de solutions potentielles (particules), qu'elle fait évoluer à l'aide des échanges d'informations (essaim) entre particules. En faite, chaque particule ajuste sa trajectoire vers sa meilleure position dans le passé et vers la meilleur position des particules de son voisinage. La variante globale de ces algorithmes considère la totalité de l'essaim comme voisinage. Les particules profitent ainsi des découvertes et expériences antérieurs de toutes les autres particules.

[6]La mémétique est l'étude des mèmes, autrement dit d'entités réplicatives d'information. Le terme de mémétique a été proposé pour la première fois par Richard Dawkins dans son œuvre The Selfish Gene en 1976, et provient d'une association entre gène et mimesis (du grec « imitation »).

- Les algorithmes basés sur l'estimation de distribution

Ces méthodes, connues sous le nom d'algorithmes basés sur l'estimation de distribution, construisent un modèle probabiliste pour les bonnes solutions, qu'elles utilisent pour guider la recherche. Contrairement aux algorithmes évolutionnaires qui construisent les nouvelles solutions en utilisant une information locale sur les bons individus à travers des opérateurs de croisement et de mutation, les algorithmes basés sur l'estimation de distribution utilisent l'information globale contenue dans la population.

II-3-2-2-4 Emploi du temps et méthodes à base de population :

L'application des algorithmes génétiques aux problèmes d'emploi du temps est très abondante dans la littérature :

Dans [ROS95], les auteurs introduisent un mécanisme intéressant permettant de guider la recherche. Ce mécanisme permet de détecter les conflits dans une solution. Pour cela, un score de violation de contraintes est gardé pour chaque gène de chromosome à l'étape d'évaluation des individus. Cette information est exploitée par l'opérateur de mutation, elle sera utilisée pour favoriser l'altération des gènes à score de violations de contraintes élevé.

Les auteurs proposent aussi une amélioration de l'étape d'évaluation visant à réduire son temps de calcul. La délta-évaluation consiste à utiliser l'évaluation des parents afin de calculer la valeur d'adéquation des enfants. Les individus enfants présentent des similarités avec leurs parents, delà seul les modifications sur la fonction objectif dues aux parties du chromosomes altérées seront examinées dans l'évaluation de l'individu enfant.

Dans [ABR92], l'auteur regroupe dans une même liste les cours ayant lieu à la même période. L'objectif de la résolution se limite à éliminer les cas de conflits. Il définit sa fonction objectif comme étant la somme des violations des contraintes. Après le croisement d'Abramson qui définit pour chaque période un site de croisement, la liste des cours associés à la même période sera composée des premiers éléments du parent1 suivie des derniers éléments du parent2.

D'autres approches sont proposées telles que :

- Les algorithmes génétiques pour la résolution des problèmes d'emploi du temps des cours aux universités [BUR94a, BUR94b]
- Les algorithmes mémétiques pour la résolution de problèmes d'emploi du temps

des examens aux universités [BUR95]

- Les algorithmes de colonies de fourmis pour la résolution des problèmes d'emploi du temps des cours aux universités [SOC02]

II-4-3 Les méthodes hybrides :

Le mode d'hybridation qui semble le plus fécond concerne la combinaison entre les méthodes de voisinage et les méthodes évolutives. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance de recherche de méthodes de voisinage et de recombinaison des algorithmes évolutionnaires sur une population de solutions. Un tel algorithme utilise une ou plusieurs méthodes de voisinage sur les individus de la population pendant un certain nombre d'itération ou jusqu'à la découverte d'un ensemble d'optima locaux et invoque ensuite un mécanisme de recombinaison pour créer de nouveaux individus.

Les algorithmes hybrides sont considérés parmi les méthodes les plus puissantes. Cette puissance réside dans la combinaison des deux principes de recherche fondamentalement différents comme on a vu dans le paragraphe précédent. Le rôle de la méthode de voisinage est d'explorer en profondeur une région donnée de l'espace de recherche alors que la méthodes évolutive introduit des règles de conduite générales dans le but de guider la recherche au travers de l'espace de recherche. Dans ce sens, les opérateurs de combinaison ont un effet diversificateur bénéfique à long terme.

Cette approche d'hybridation a permis de produire beaucoup de travaux dans la littérature par exemple :

- les algorithmes mémétiques
- la méthode GRASP
- L'algorithme MA/PM (Memetic Algorithm with Population Management) [MSKS]
- …

II-5 Analyse des métaheuristiques :

Malgré l'évolution permanente au niveau des méthodes exactes, elles rencontrent toujours des difficultés face aux problèmes de taille importante car le temps de calcul devient prohibitif. Compte tenu de ces difficultés on a opté pour les métaheuristiques pour la résolution du problème d'emploi du temps, ces méthodes formant un outil formidable pour la résolution efficace des problèmes posés, exploitent au mieux les problèmes d'optimisation dont le but de trouver une solution de qualité

raisonnable en un temps de calcul aussi faible que possible. Mais la question qui se pose : quelle est la meilleure métaheuristique qu'on peut utiliser ?

Il n'est malheureusement pas possible de répondre de manière directe et précise à cette question : en effet, si certaines similitudes sont évidentes entre ces différentes approches (voisinages), elles diffèrent sur des points relativement délicats (fonction d'énergie du recuit simulé, liste des mouvements interdits de tabou, opérateurs de croisement des algorithmes génétiques…). Des comparaisons s'avèrent difficiles.

Les inventeurs des algorithmes génétiques ont introduit deux notions très importantes : l'exploitation et l'exploration. L'exploitation consiste à rechercher plus particulièrement dans les zones de recherche déjà définies (stratégie de recherche en profondeur d'abord), tandis que l'exploration réalise un survol de l'ensemble de l'espace des configurations afin de découvrir des zones prometteuses ( stratégie de recherche en largeur d'abord). Ces deux notions complémentaires ne sont pas restreintes aux algorithmes génétiques mais concernent l'ensemble des méthodes de recherche et peuvent servir à caractériser ces méthodes (dans ce qui suit on s'intéresse beaucoup plus aux méthodes les plus répandues : méthode de recuit simulé, méthode de recherche tabou, les algorithmes génétiques et les algorithmes de colonie de fourmis).

II-5-1 Exploitation :

Les méthodes de voisinage utilisent la fonction de voisinage afin de diriger rapidement la recherche vers des configurations de bonne qualité. En cherchant les candidats potentiels uniquement à faible distance des meilleurs configurations retenue, elle réalisent de ce fait une recherche axée sur l'exploitation.

Dans les algorithmes génétiques, la sélection a pour effet de concentrer la recherche autour des configurations de meilleure performance et les opérateurs de croisement sont destinés pour faire profiter aux enfants des « bonnes » caractéristiques des parents. de ce fait la notion d'exploitation dans les algorithmes génétiques est amenée à la fois par le processus de sélection et par l'opérateur de croisement. Dans les méthodes de voisinage, l'exploitation est principalement assurée par la préférence accordée à une configuration de bonne performance dans la procédure de réparation. Plusieurs méthodes introduisent des mécanismes spécifiques supplémentaires d'exploitation : c'est le cas de l'intensification dans la méthode tabou.

Cependant l'utilisation exclusive du principe d'exploitation ne suffit pas à donner des résultats satisfaisants. En effet, ce principe ne permet pas une recherche efficace et conduit à limiter la recherche dans une zone non renouvelée qui finit par

s'épuiser. Le cas de l'amélioration itérative rapidement piégée dans un optimum local illustre cruellement ce phénomène. Une autre illustration souvent évoquée est fournie par le problème de convergence prématurée des algorithmes génétiques : du fait de la sélection, la population finit par n'être constituée que d'individus tous similaires. L'une des préoccupations majeures dans les algorithmes génétiques consiste d'ailleurs à préserver le plus longtemps possible une diversité suffisante dans la population. Face à ce type de difficulté, la solution pour se débarrasser de ce problème consiste à mettre en place des mécanismes correctifs qui vont aider à diriger la poursuite de la recherche vers de nouvelles zones, c'est à dire, recourir à l'exploration.

II-5-2 Exploration :

La première stratégie envisageable est la relance. Elle consiste à recommencer entièrement le processus de recherche ( en gardant la meilleure configuration obtenue) à partir d'un nouveau tirage aléatoire dans l'espace des configurations.

La deuxième stratégie qui est aussi simple à mettre en œuvre que la précédente consiste à introduire des perturbations aléatoires au cours de la recherche. C'est le cas pour les mutations dans les algorithmes génétiques ainsi que pour la génération aléatoire d'un voisin dans le recuit simulé. Dans les deux cas, la configuration courante est altérée de manière aléatoire et un mécanisme d'acceptation est appliqué a posteriori.

La troisième stratégie consiste à mémoriser au cours de la recherche les caractéristiques des régions visitées et à introduire un mécanisme permettant de s'éloigner de ces zones. C'est ce que fait la méthode tabou avec la liste tabou (court terme) et avec la diversification (long terme). La recherche locale guidée reprend cette même idée. On peut remarquer que cette stratégie est plus complexe à implanter que les précédentes et qu'elle nécessite l'utilisation d'une mémoire.

Les métaheuristiques se différencient également selon la manière dont elles font varier l'intensité de l'exploration au cours de la recherche. Le recuit simulé présente une manière spécifique de procéder : le niveau des perturbations aléatoires (responsables de l'exploration), lié au paramètre de température, est initialement fixé à un niveau élevé et abaissé progressivement au cours de la recherche. La méthode tabou illustre une manière très différente de doser l'exploration : un mécanisme particulier d'exploration (la diversification) succède régulièrement à de longues phases de recherche. Dans les algorithmes de colonies de fourmis, il existe une autre façon de gérer l'exploration : on passe par le réglage via les deux paramètres $\alpha$ et $\beta$, qui déterminent l'influence relative des pistes de phéromone et de l'information heuristique. Plus la valeur de $\alpha$ sera faible, plus la diversification (exploration) sera forte, car les fourmis éviteront les pistes. A

l'inverse, plus $\alpha$ sera élevé, plus l'intensification (exploitation) sera importante, car plus les pistes auront une influence sur le chois des fourmis. Le paramètre $\beta$ agit de façon similaire.

La recombinaison constitue aussi un autre principe général qui complète les deux notions précédentes (exploitation et exploration) afin d'essayer d'atteindre des zones non encore explorées. Ces méthodes permettent de commencer la recherche dans une nouvelle région tout en ayant déjà des individus satisfaisants au sens de la fonction de coût. Initialement issue des algorithmes génétiques (le croisement), elle consiste à recombiner plusieurs configurations pour en créer de nouvelles. Des travaux récents ont montré q'une recombinaison approprié de solutions peut améliorer sensiblement l'efficacité de la recherche

Selon le point de vue de plusieurs auteurs, la puissance d'une métaheuristique est d'abord liée à son aptitude à intégrer des connaissances spécifiques du problème. La connaissance du problème la plus fondamentale réside dans le codage du problème et dans le choix de la fonction de voisinage. En général , il n'existe pas de codage universellement efficace. Un « bon » codage doit permettre de restreindre l'espace de recherche et d'intégrer des connaissances du problème.

Pour améliorer leur efficacité pour un problème donné, les métaheuristiques adaptent leurs opérateurs en fonction des connaissances spécifiques que possède le concepteur sur ce problème. Cette adaptation (si les connaissances nécessaires sont disponibles) demandent un effort de conception pour spécialiser l'opérateur. Ainsi la recherche tabou se spécialise selon le choix des caractéristiques sauvegardées dans la liste tabou. De leur coté, les algorithmes génétiques se sont éloignés du modèle standard pour intégrer également des connaissances du problème, proposant à présent des opérateurs spécifiques ainsi que des codages des individus ne reposant plus uniquement sur les chaînes de bits. D'un autre coté, le recuit simulé est au contraire un exemple de méthode facile à adapter à un problème et ne demandant pas d'exploiter des connaissances spécifiques.

D'une manière générale, plus une méthode offrant les possibilités d'intégrer des connaissances spécifiques du problème, plus elle dispose de moyens potentiels pour conduire efficacement la recherche et produire de bons résultats. Mais demande des efforts en matière d'adaptation et de spécialisation. Au contraire, une méthode qui prétend n'intégrer aucune connaissance ne peut être compétitive.

## II-6 Problèmes de satisfaction de contraintes :

Le problème d'emploi du temps est un problème défini en termes de contraintes (de temps, d'espaces ou plus généralement de ressources comme d'ailleurs d'autres problèmes tels que :

- les problèmes de planification et d'ordonnancement : planifier une production, gérer un trafic ferroviaire…
- les problèmes d'affectation du personnel à des tâches, des entrepôts à des marchandises,…

etc…

Ces différents problèmes fortement contraint ayant la particularité commune d'être d' être caractérisés par une très forte combinatoire, peuvent être considérés comme des problèmes de satisfaction de contraintes (CSP : Constraint Satisfaction Problems) car ces problèmes se formulent aisément en CSP quand ils ne nécessitent que des contraintes fortes.

L'utilisation d'une technique de satisfaction de contraintes est adaptée aux problèmes très contraints où une exploration de l'espace de recherche est envisageable.

Dans ce type de problème, la difficulté est de trouver une solution satisfaisant toutes les contraintes et non de trouver une solution minimisant ou maximisant une fonction, on fait d'ailleurs souvent abstraction de cette fonction en ne tenant compte que des contraintes : on cherche une solution réalisable et non pas la meilleure solution.

### II-6-1 Qu'est ce qu'un CSP :

Un CSP est défini comme étant un ensemble de contraintes impliquant un ensemble de variables, dont chacune est définie sur son domaine propre. L'objectif consiste à trouver un ensemble de valeurs, choisies dans les domaines susmentionnés, à affecter à ces variables de sorte que toutes les contraintes soient satisfaites.

### II-6-2 Concepts de base d'un CSP:

Plus formellement, un problème de satisfaction de contraintes est défini par le triplet (X,D,C) tel que :

- $X = ( X_1, X_2, \ldots\ldots, X_n )$ est l'ensemble des n variables du problème.
- $D = ( D_1, D_2, \ldots\ldots, D_n )$ est un ensemble de n domaines finis dont chacun est associé à une variable de X. C'est à dire le domaine $D_i$ est associé à la variable $X_i$ (leurs valeurs possibles).

- C = ( $C_1$,$C_2$,……,$C_m$ ) est un ensemble de m contraintes. Chaque contrainte $C_i$ est défini par un couple ( $v_i$,$r_i$ ) tel que :

  - $v_i$ = {$X_{i1}$,…..$X_{ini}$} est un ensemble de ni variables sur lesquelles porte la contrainte $C_i$ ; ni est appelé arité de la contrainte.

  - $r_i$ est une relation définie par un sous-ensemble de produits cartésiens $D_{i1}$x…..x$D_{ni}$ des domaines associés aux variables de $v_i$. Elle représente les n-uplets de valeurs autorisées pour ces variables.

Définitions :

Pertinence : on dit qu'une variable $X_i$ est pertinente pour la contrainte $C_k$, si $C_k$ porte sur la variable Xi.

Arité de la contrainte : l'arité de la contrainte $C_k$ est le nombre de variables pertinents pour $C_k$.

CSP binaire : un CSP binaire est un CSP P = ( X,D,C ) dont toutes les contraintes $C_k \in C$ ont une arité égale à 2. En d'autres termes, chaque contrainte a exactement 2 variables pertinentes. Notons que tout CSP n-aire peut être ramené à un CSP binaire équivalent [ROS89].

Matrice de contraintes : une matrice de contraintes est une matrice Mat à m lignes (m étant le nombre de contraintes) et n colonnes (n étant le nombre de variables du CSP) telle que :

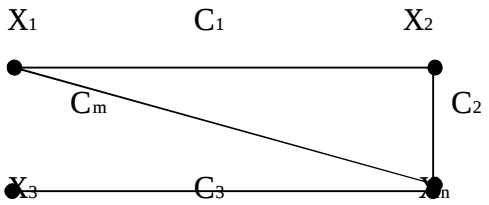
$$Mat[i,j] = \begin{cases} 1 & \text{si } X_i \text{ est pertinent pour } C_i \\ 0 & \text{sinon} \end{cases}$$

| | variables | $X_1$ | $X_2$ | $X_3$…….$X_n$ |
|---|---|---|---|---|
| | C1 | 1 | 1 | 0……0 |
| | C2 | 0 | 1 | 0……1 |
| contraintes | C3 | 0 | 0 | 1……1 |
| | Cm | 1 | 0 | 0……1 |

Matrice de contraintes Mat[i,j]

Sur cet exemple, on peut lire : $X_1$ et $X_2$ sont pertinentes pour $C_1$, X2 et Xn sont pertinentes pour $C_2$ etc…

Graphe de contraintes : une contrainte relie les deux variables pour lesquelles elles sont pertinentes par une arête, les sommets du graphe étant les variables.



Graphe de contraintes

Instanciation : étant donnée un CSP (X,D,C), on appelle instanciation du CSP I une application qui associe à chaque variable $X_i \in X$ une valeur I, telle que $Xi \in D$.

Instanciation partielle : une instanciation partielle $I_p$ de $X_p = \{X_{p1},\ldots\ldots,X_{pj}\} \subseteq X$ est une application qui associe à chaque variable $X_{pi} \in X_p$ une valeur $I_p(X_{pi}) \in D_{pi}$ ($D_{pi}$ est le domaine de $X_{pi}$).

Satisfaction de contraintes : soit P un CSP, une instanciation partielle $I_p$ satisfait la contrainte $C_i = (v_i, r_i)$ et on notera ($I_p \mid= C_i$) ssi $v_i \in X_p$ et $I_p(v_i) \in r_i$. On dira alors que $I_p$ viole $C_i$ ssi $v_i \in X_p$ et $I_p(v_i) \notin r_i$.

Instanciation consistante : soit P un CSP, une instanciation partielle $I_p$ des variables est dite consistante ssi $C_i = (v_i, r_i) \in C$ telle que $v_i \in X_p$ et $I_p \mid= C_i$.

Solution d'un CSP : une solution S d'un CSP P est une instanciation consistante de toutes les variables (c'est à dire $X=X_p$). On dira que S satisfait P ( $S \mid= P$).

II-6-3 Méthodes de résolution des CSPs :

Plusieurs approches sont utilisées pour résoudre les CSPs, on distingue les méthodes exactes (ou complètes) et les méthodes approchées (ou incomplètes).

Les méthodes exactes font une recherche arborescente en instanciant les variables une par une et effectuent des réparations en cas d'échec. Les méthodes approchées font une réparation d'une configuration en parcourant d'une manière aléatoire l'espace de recherche.

II-6-3-1 Méthodes exactes :

La plupart des algorithmes de recherche exacte pour les problèmes de satisfaction de contraintes sont basés sur l'algorithme Backtrack Standard. De nombreuses améliorations majeures ont été proposées pour cet algorithmes de base.

a) L'énumération (Backtrack) :

Cette méthode consiste à énumérer l'ensemble des solutions potentielles et à vérifier que chaque contrainte est satisfaite. Cette méthode donne une bonne réponse à tous les coups, mais conduit à développer un grand nombre de solutions potentielles (croissance exponentielle avec la taille des données) rendant souvent les traitements très lourds. Pour trouver systématiquement une solution, l'algorithme affecte au fur et à mesure une valeur de D à chaque variable correspondant dans X. il vérifie évidement que la valeur est correcte par rapport aux contraintes. A chaque affectation, les domaines des variables restantes diminuent. Il arrive qu'à l'affectation d'une variable $X_{n-1}$, l'algorithme ne puisse trouver de valeur pour $X_n$ car $D(X_n)$ est vide. A ce stade, l'algorithme revient en arrière, à la dernière variable affectée $X_{n-1}$. il modifie la valeur de $X_{n-1}$ en espérant que le domaine de la variable $X_n$ suivante ne sera plus nul. Si après avoir essayé toutes les valeurs du domaine $D(X_{n-1})$, il n'a pas toujours de solutions pour $X_n$, il recule et modifie la variable précédent $X_{n-2}$. s'il n'existe pas de solution, il reculera jusqu'à $X_0$, sinon il trouve obligatoirement la solution.

Le défaut de cet algorithme est qu'il peut tester toutes les valeurs possibles avant de trouver une solution. Compte tenu de la nature des problèmes gérés, ses dimensions deviennent rapidement ingérables, car elles augmentent selon une loi de type n puissance p. si jamais la première bonne solution se trouve dans les dernières possibilités ou si jamais il n'y a pas de solutions, le temps de calcul sera très long.

c) Amélioration du Backtrack :

1- Algorithmes avec retour arrière non chronologique :

Quand il rencontre un échec, l'algorithme Backtrack remet en cause le dernier choix et essaie alors une nouvelle valeur pour la variable courante x. lorsque toutes les valeurs ont été utilisées, il revient en arrière sur la variable précédente. Cependant, rien ne garantit que cette variable soit en cause dans les différents échecs rencontrés lors de l'affectation de x. par exemple, si cette variable n'est pas liée à x par une contrainte, elle ne peut être impliquée dans ces échecs. Par conséquent, essayer de nouvelles valeurs pour cette variables conduira aux mêmes échecs au niveau de l'affectation de x. pour

pallier ce défaut du Backtrack, plusieurs méthodes dotées de retour arrière non chronologiques ont vu le jour. Le retour arrière non chronologique consiste à analyser les causes des échecs et à revenir sur la variable la plus profonde qui est en cause dans l'échec. C'est la notion de saut en arrière ou backjump. La façon d'analyser les causes des échecs détermine alors l'algorithme. Parmi les algorithmes exploitant une technique de retour arrière non chronologique, on cite :

- l'algorithme Backjumping (BJ): lorsque toutes les extensions de l'affectation courante avec une valeur de x sont inconsistantes, BJ revient en arrière sur la variable la plus profonde dans l'arbre dont la valeur est en conflit avec une valeur de x.
- l'algorithme Graph-based Backjumping (GBJ) : le retour arrière repose sur le graphe de contraintes. Si l'affectation courante ne peut être étendue de façon consistante à une variable x, alors GBJ revient en arrière sur la variable la plus profonde du voisinage de x. soit y cette variable. Si toutes les valeurs du domaine de y ont été essayées, GBJ revient sur la variable la plus profonde qui appartienne soit au voisinage de y, soit au voisinage de toute variable instanciée après y qui soit une cause potentielle de l'échec d'une extension de l'affectation courante (x est en une), et ainsi de suite.
- L'algorithme Conflict-directed Backjumping (CBJ) : pour chaque variable instanciée x, CBJ maintient un ensemble dit ensemble des conflits. Cet ensemble contient toutes les variables instanciées avant x avec lesquelles x est en conflit pour au moins une de ses valeurs ainsi que les variables qui sont en cause dans l'échec d'une extension d'une affectation contenant x. lorsqu'un échec survient ou lorsque toutes les valeurs ont été essayées, on revient en arrière jusqu'à la variable la plus profonde de l'ensemble des conflits de la variable courante.

2- Algorithmes avec filtrage avant :

L'algorithme Backtrack, teste la consistance de l'affectation courante en vérifiant qu'elle ne viole aucune contrainte liant x et une variable affectée. Autrement dit, le test de consistance s'effectue en fonction des variables déjà instanciées. C'est le concept de consistance en arrière (ou look-back scheme). Une autre technique consiste à exploiter le concept de consistance en avant (ou look-ahead scheme). Suivant ce concept, à chaque affectation d'une variable x, les valeurs des variables non instanciées qui ne sont pas compatibles avec la valeur de x sont supprimées. Le calcul des valeurs à supprimer dépend alors du niveau de filtrage utilisé. Un algorithme utilisant ce concept cherche à étendre une affectation consistante A. dans ce but, il choisit une variable non instanciée et lui affecte une valeur v du domaine. Le choix des variables et des valeurs à instancier fait généralement appel à des heuristiques. Une fois l'affectation construite, il supprime du domaine de chaque variable non affectée les valeurs qui ne sont pas

compatibles avec v selon le niveau de filtrage utilisé. Si un domaine devient vide, alors l'affectation ne possède pas d'extensionconsistante. Il faut alors essayer une nouvelle valeur pour x. si toutes les valeurs ont été essayées, l'algorithme revient en arrière sur la variable affectée juste avant x. Si tous les domaines possèdent au moins une valeur, alors l'algorithme tente d'étendre en procédant comme précédemment. La recherche se termine quand toutes les variables sont instanciées (découverte d'une solution) ou si toutes les possibilités ont été étudiées (preuve de l'inconsistance du problème).

3- Algorithmes avec mémorisation :

Les algorithmes avec retour arrière non chronologique permettent d'éviter certaines redondances dans l'arbre de recherche. Toutefois, il subsiste tout de même des redondances. Aussi, pour ne pas visiter plusieurs fois les mêmes sous arbres, une solution consiste à mémoriser des informations portant sur le travail déjà réalisé. Ces informations sont généralement mémorisées sous la forme de contraintes induites désignées généralement par le terme nogood. Un nogood correspond à une affectation qui ne peut être étendue en une solution. Leur mémorisation permet donc d'interdire certaines affectations et ainsi d'éviter de reproduire plusieurs fois les mêmes sous arbres. Parmi ces méthodes, on peut citer :

- Jump-back learning
- Nogood recording
- Dynamic backtracking

II-6-3-2 Les méthodes approchées :

Les méthodes approchées réparent une configuration donné en parcourant de manière aléatoire l'espace de recherche. La méthode min-conflits [MIN92] est la plus répandue. Elle est construite autour d'une heuristique de réparation locale avec minimisation de conflits. L'heuristique consiste à choisir une variable intervenant dans une contrainte non satisfaite. Et à choisir pour cette variable, une valeur qui minimise le nombre de contraintes violées.

Les méthodes stochastiques sont aussi des méthodes approchées, mais elles ont une approche différente de celles citées précédemment. Les méthodes stochastiques commencent par une instanciation initiale et elles essaient de la réparer en faisant des changements de valeurs de certaines variables. Contrairement aux autres méthodes approchées, celles-ci admettent une dégradation de la solution, d'une façon qui suit certaines règles qui font que, l'algorithme puissent sortir d'un optimum local. Parmi ces méthodes on cite le recuit simulé, la méthode tabou, les algorithmes génétiques et la méthode basée sur le SMA(Système Multi Agent) : les colonies de fourmis.

### II-6-3-3 Emploi du temps et CSP :

Dans [FRA04], l'auteur utilise une approche énumérative pour résoudre le problème de l'emploi du temps. Ce problème d'optimisation d'emploi du temps se défini par six ensembles :
Un ensemble Professeurs = $\{P_1, \ldots, P_P\}$, un ensemble Formations = $\{F_1,\ldots,F_f\}$, un ensemble Enseignements = $\{E_1,\ldots, E_e\}$, un ensemble Salles = $\{S_1,\ldots,S_s\}$, un ensemble Créneaux = $\{C_1,\ldots,C_c\}$et un ensemble Contraintes contenant l'ensemble des contraintes entre les variables des cinq ensembles précédents. La connaissance du problème permet pour chaque cours de fixer quelques variables. Ainsi pour chaque professeur Pi de l'ensemble Professeurs ,est fixé quel enseignement et à quelle formation il le dispensera. Il reste donc qu'à fixer pour chaque cours la salle et le créneau qui seront utilisés et qui satisfassent les contraintes. Le CSP proposé se présente de la façon suivante :
$V = \{ P_1, \ldots, P_P, F_1,\ldots,F_f, E_1,\ldots, E_e, S_1,\ldots,S_s, C_1,\ldots,C_c\}$
$D = \{D(P_1) =\ldots= D(P_p) =[1,p], D(F_1) =\ldots= D(F_f) = [1,f], D(E_1) =\ldots= D(E_e) = [1,e], D(S_1) =\ldots= D(S_s) = [1,s], D(C_1) =\ldots= D(C_c) = [1,c]\}$
C= {les contraintes dures plus les contraintes faibles}.

## II-7 Conclusion :

Les problèmes d'optimisation combinatoire est plus particulièrement ceux de l'emploi du temps se présentent comme étant intraitables par les algorithmes classiques. La raison est que la topologie de tels problèmes est d'un degré de complexité élevé, présentant une difficulté découlant du volume immense d'informations qui composent le problème (données et contraintes).

Les méthodes proposées peuvent êtres classifiées principalement en deux grandes classes : les méthodes de recherche exacte et approchées. Les méthodes de recherche exacte tentent d'atteindre la solution optimale du problème, ce qui induit des temps de recherche importants. Alors que souvent les décideurs semblent largement se satisfaire d'une bonne solution approximative qui est atteinte avec des temps de recherche plus réduits.

De ce fait, les métaheuristiques se sont illustrées comme une alternative intéressantes pour la résolution des problèmes d'optimisation combinatoire. On retrouve deux classes de métaheuristiques : les méthodes de recherche à base de voisinage et à base de population. Les méthodes de recherche à base de population ont la particularité de travailler sur une population de solutions, cas des algorithmes génétiques et colonies de fourmis. Les méthodes de recherche à base de voisinage telles

que le recuit simulé et la recherche tabou travaillent quant à elles sur une seule solution à un moment donné.

La puissance des métaheuristiques est souvent améliorée par l'exploitation des possibilités d'hybridation. Cette dernière offre la possibilité de bénéficier de l'efficacité des différentes métaheuristiques à des stades différents de recherche.

Les problèmes d'optimisation (plus particulièrement le problème de l'emploi du temps) sont fortement contraints et sont caractérisés par une forte combinatoire. L'utilisation d'une technique de satisfaction de contraintes (CSP) est adaptée aux problèmes très contraints où une exploration de l'espace de recherche est envisageable. Dans ce type de problèmes, la difficulté est de trouver une solution satisfaisant toutes les contraintes et non de trouver une solution minimisant ou maximisant une fonction, on fait d'ailleurs souvent abstraction de cette fonction en ne tenant compte que des contraintes : on cherche une solution réalisable et non pas la meilleure solution.

Les approches citées plus haut lors de cette partie sont dédiées à la résolution des problèmes mono-objectf. Comme notre problème est lui aussi un problème d'optimisation multi-objectif, nous présentons dans la partie suivante une étude des problèmes d'optimisation multi-objectif, leurs particularités et les différentes approches de résolution.

# CHAPITRE III

# L'OPTIMISATION MULTIOBJECTF

Dans ce chapitre, nous présentons les concepts de base de l’optimisation multiobjectif. Nous dresserons aussi un constat des différentes techniques de résolution des problèmes d’optimisation multiobjectif.

## III-1 Introduction :

Pendant de nombreuses années, l'intérêt des chercheurs s'était porté sur les problèmes d'optimisation où il s'agit de maximiser ou minimiser une seule fonction objectif par rapport à un ensemble de paramètres ou variables de décision. Un constat qui ne concorde pas toujours avec la réalité, où la majorité des problèmes présentent plusieurs objectifs à optimiser (souvent contradictoires).

Dans le cas de notre problème d'emploi du temps, des critères tels que la procuration de journées libres et l'équilibrage de charge d'étude sur la semaine peuvent être considérés comme objectifs de l'optimisation. Cependant, ces deux critères sont relativement contradictoires, du fait que la procuration de journées libres implique inévitablement la surcharge de certaines journées par rapport à d'autres.

Le rôle de l'optimisation dans ce cas consiste à générer un ensemble de solutions optimales plutôt qu'une seule. Il appartiendrait donc au décideur de choisir une solution parmi celles proposées.

Ce chapitre est consacré à la présentation des concepts de base de l'optimisation multiobjectif. Nous étudierons les différentes techniques utilisées dans ce cadre telles que la programmation mathématique, les techniques de recherches arborescente et les méta heuristiques.

Dû à leur maniement d'une population de solutions, les algorithmes évolutionnaires ont été les plus utilisés dans le cadre de l'optimisation multiobjectif. Pour cela nous nous étalerons sur l'étude des différents mécanismes qui ont servi à l'implémentation des algorithmes génétiques pour ce type d'optimisation.

## III-2 Concepts de base :

Mathématiquement, un problème d'optimisation multiobjectif (POMO) est décrit comme suit :

$$\begin{aligned} &\text{Optimiser } f(x) = [f_1(x), f_2(x), \ldots\ldots, f_k(x)] \text{ (k fonctions à optimiser)} \\ &\text{Avec} \quad g(x) \leq 0 \quad \text{(m contraintes d'inégalités)} \\ &\text{Et} \quad h(x) = 0 \quad \text{(p contraintes d'égalités)} \\ &\text{Où} \quad x \in \Re^n,\ g(x) \in \Re^m,\ h(x) \in \Re^p \text{ et } f(x) \in \Re^k \end{aligned} \tag{3.1}$$

La résolution de ce problème consiste à minimiser au mieux ces k fonctions objectifs dans le but de ne pas trop dégrader les valeurs des optima obtenus par rapport à ceux obtenus lors d'une optimisation mono-objectif effectuée objectif par objectif.

On remarque donc, $f_1,f_2,.....,f_k$ désignent les différents objectifs à optimiser. Les variables $x_1,x_2,.....,x_n$ représentent les variables de décision. L'évaluation d'une solution x est décrite par un vecteur objectif $f(x) = [f_1(x),f_2(x),.....,f_k(x)]$. Nous serons donc en face de deux types d'espaces :

- l'espace de décision X de dimension n qui désigne l'ensemble des solutions $x=[x_1,x_2,.....,x_n]$.

- L'espace objectif de dimension k qui désigne l'ensemble des évaluations possibles correspondant aux vecteurs objectifs f(x).

A chaque point de l'espace de décision est associé un point de l'espace objectif désignant l'adéquation de la solution associée. Le POMO consiste à trouver une solution x* qui satisfasse l'ensemble de toutes les contraintes et qui optimise le vecteur objectif f(x*).
L'optimisation du vecteur objectif f revient soit :

- A minimiser l'ensemble de toutes les fonctions objectifs
- A maximiser l'ensemble de toutes les fonctions objectifs
- Minimiser certaines fonctions objectifs tout en maximisant d'autres.

Pour des raisons de simplification, un problème de maximisation peut être aisément transformé en un problème de minimisation en considérant l'équivalence suivante :

$$\text{Maximiser } f(x) \Leftrightarrow \text{ minimiser } -f(x) \qquad (3.2)$$

III-2-1 Multiplicité de solutions :

La résolution d'un POMO ne donne pas une solution unique, mais plusieurs solutions possibles car les POMO présentent des objectifs souvent contradictoires (la minimisation d'un objectif peut entraîner la maximisation d'un autre). Ainsi, on obtient un ensemble de solutions non optimales, car elles ne minimisent pas toutes les fonctions objectifs en même temps. Ces solutions sont appelées solutions de compromis. Pour cela, il est vital pour identifier ces meilleurs compromis de définir une relation d'ordre entre ces éléments. Dans le cas des POMO, cette relation est appelée relation de dominance.

III-2-2 Relation d'ordre et de dominance :

La relation de dominance nous permet de restreindre l'ensemble des solutions de compromis. Plusieurs types de relations de dominance existent dans la littérature [COL02]. Mais la plus célèbre et la plus utilisée est la dominance au sens de Pareto qui est défini par :

Le vecteur u domine le vecteur v si :

u est au moins aussi bon que v en regard à tous les objectifs et
u est strictement meilleur que v pour au moins un objectif.

Pour définir clairement et formellement cette notion de dominance au sens de Pareto, les relations =, ≤ et < usuelles sont étendues aux vecteurs.

Définition 1 :

Soient u et v deux vecteurs de même dimension, $u=(u_1,u_2,....,u_m)$ et $v=(v_1,v_2,....v_m)$ alors

$u = v$ ssi $\forall i \in \{1,2,....m\}, ui = vi$
$u \leq v$ ssi $\forall i \in \{1,2,....m\}, ui \leq vi$
$u < v$ ssi $u \leq v \wedge u \neq v$

Les relations ≥ et > sont définies de manière analogue.

Dans le cas des problèmes à un seul objectif, les relations usuelles <, ≤,...suffisent pour comparer les points, par contre, elles sont insuffisantes pour comparer des points issus des problèmes multiobjectifs par exemple les points (qui sont ici des vecteurs) u = (1,2) et v = (4,1) sont incomparables à l'aide de ces relations. Nous définissons donc maintenant la relation de dominance au sens de Pareto qui permet de comparer tous les points (vecteurs) de décision possibles.

Définition 2 : la dominance au sens de Pareto :

Soient u et v deux vecteurs de décision, $u= (u_1,u_2,....,u_m)$ et $v=(v_1,v_2,....v_m)$
Considérons un problème de minimisation

u domine v ssi $f(u) < f(v)$
u domine faiblement v ssi $f(u) \leq f(v)$
u est incomparable (non dominée) avec v ssi not $(f(u) \leq f(v)) \wedge$ not $(f(v) \leq f(u))$

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

Exemple de dominance :

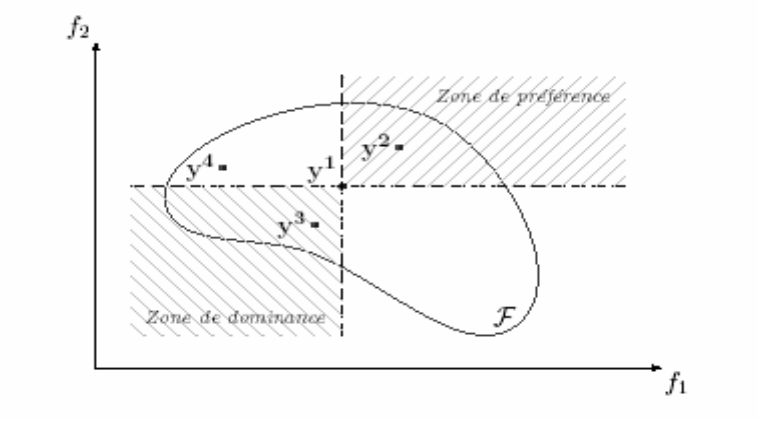



Fig 3.1- Exemple de dominance

La figure 3.1 représente un exemple de dominance. On a un problème d'optimisation bi-objectif. Les fonctions objectif sont $f_1$ et$f_2$. Chaque point $y^i$ est l'image de $x^i$ par $f$ : $y^i = f(x^i)$ avec i=1..4. Prenons le point $y^1$ comme point de référence. Nous pouvons distinguer trois zones :
- La zone de préférence est la zone contenant les points dominés par $x^1$.
- La zone de dominance est la zone contenant les points dominant $x^1$.
- La zone d'incompatibilité contient les points incomparables avec $x^1$.

On obtient alors les relations de dominance suivantes :
$x^1$ domine $x^2$ , $x^1$ est dominé par $x^3$ et $x^1$ est non dominé (incomparable) avec $x^4$.

Avec ce nouvel outil, nous pouvons maintenant définir l'optimalité dans le cas des problèmes multiobjectifs. Nous pouvons définir l'optimalité au sens de Pareto c'est à dire utilisant la dominance au sens de Pareto :

Définition 3 : L'optimalité globale au sens de Pareto :

Soit F l'image de l'ensemble réalisable X dans l'espace des objectifs. Un vecteur de décision $u \in X$ est dit Pareto globalement optimal si et seulement si $\neg\exists v \in X$, v domine u. dans ce cas $f(u) \in F$ est appelé : solution efficace.

Définition 4 : L'optimalité locale au sens de Pareto :

Un vecteur de décision $u \in X$ est dit Pareto localement optimal si et seulement si, pour un $\delta > 0$ fixé : $\neg\exists\ v \in X$, $f(v) \in B(f(u),\delta)$ et v domine u, où $B(f(u),\delta)$ représente une boule de centre f(u) et de rayon $\delta$.

La figure 3.2 donne un exemple d'optimalité locale. Le point f(x) est localement optimal, car il n'y a pas de point compris dans la boule B le dominant.

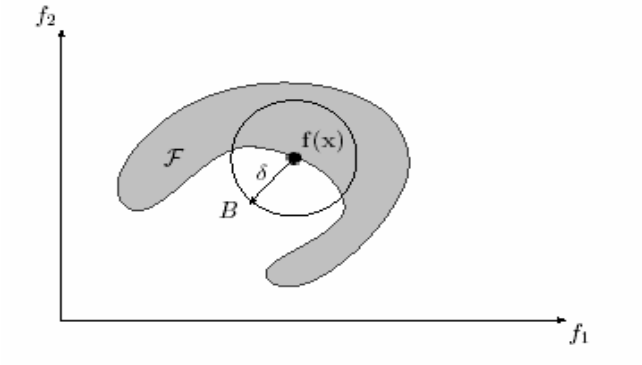



Fig 3.2 Optimalité Locale

III-2-3 Front Pareto et surface de compromis :

Dans le paragraphe (III-2-1), on a définit les solutions d'un problème multiobjectif comme solutions de compromis. Cet ensemble des meilleurs compromis est appelé aussi surface de compromis ou front Pareto, il est composé des points qui ne sont dominés par aucun autre. Formellement, on a les définitions suivantes :

Définition 5 : Soit F l'image dans l'espace des objectifs de l'ensemble réalisable X. Une solution $u \in X$ est dite non dominée par rapport à un ensemble $Y \subseteq X$ si et seulement si :

$$\neg\exists\ v \in Y,\ v \text{ domine } u \qquad (3.3)$$

Définition 6 : Ensemble des solutions non dominées :

Soit F l'image dans l'espace des objectifs de l'ensemble réalisable X. L'ensemble des solutions non dominées de X, est défini par l'ensemble ND(X) :

$$ND(X) = \{u \in X \mid u \text{ est non dominée par rapport à } X\} \qquad (3.4)$$

Définition 6 : Front Pareto :

Soit F l'image dans l'espace des objectifs de l'ensemble réalisable X. Le Front Pareto ND(F) de F est défini comme suit :

$$ND(F) = \{v \in F \mid \neg\exists\ w \in F,\ w < v\} \qquad (3.5)$$

Le Front Pareto est aussi appelé l'ensemble des solutions efficaces ou la surface de compromis.
Un exemple de surface de compromis (Front Pareto) est montré à la figure 3.3.

Définition 7 : Point Idéal :

Les coordonnées du point idéal (pi) correspondent aux meilleurs valeurs de chaque objectif des points du Front Pareto. Les coordonnées de ce point correspondent aussi aux valeurs obtenues en optimisant chaque fonction objectif séparément.

$$Pi_i = \min\{y_i \mid y \in ND(F)\} \quad (3.6)$$

Définition 7 : Point Nadir :

Les coordonnées du point Nadir (pn) correspondent aux pires valeurs de chaque objectif des points du Front Pareto.

$$Pn_i = \max\{y_i \mid y \in ND(F)\} \quad (3.7)$$

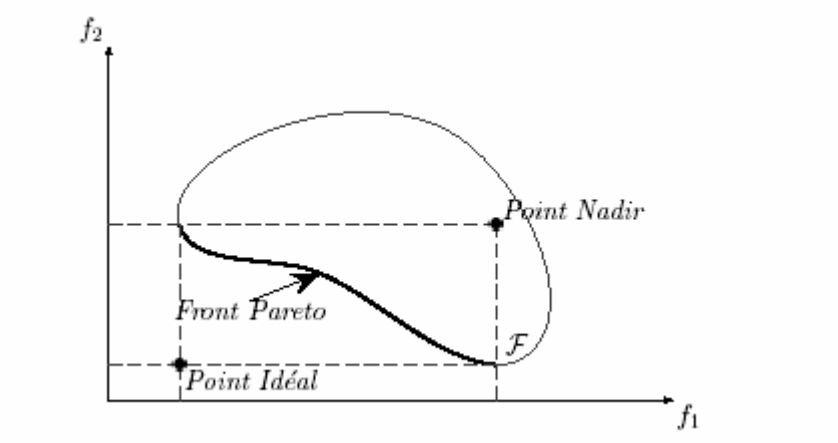



Fig 3.3 Front Pareto, Point Nadir et Point Idéal

Dans l'exemple de la fig 3.3, le problème considéré est un problème de minimisation avec deux objectifs. Deux points particuliers apparaissent clairement : le point idéal et le point Nadir. Ces deux points sont calculés à partir du Front Pareto. Le point idéal (resp. le point Nadir) domine (resp. est dominé par) tous les autres points de la surface de compromis. Bien que ces points ne soient pas forcément compris dans la zone réalisable, ils servent souvent de pôle d'attraction (resp. de répulsion) lors de la résolution du problème.
La figure 3.3 nous indique aussi que le Front Pareto peut avoir des propriétés particulières quant à sa forme. La principale caractéristique utilisée pour comparer les formes de ces courbes est la convexité. Nous rappelons alors la définition de la convexité :

Définition 8 : Convexité :

Un ensemble A est convexe, si et seulement si l'équivalence suivante est vérifiée :

$$x \in A \ \wedge \ y \in A \ \Leftrightarrow \ \text{Segment}(x,y) \subset A \quad (3.8)$$

La convexité est le premier indicateur de la difficulté du problème. En effet, plusieurs méthodes d'optimisation sont incapables de résoudre d'une façon optimale des problèmes non convexes.

## III-3 Optimisation et aide à la décision :

La résolution d'un POMO revêt deux aspects de difficultés distinctes : l'optimisation et l'aide à la décision. L'aspect optimisation, concerne le processus de recherche ou d'approximation de l'ensemble des solutions optimales (Pareto,...), l'aspect aide à la décision s'adresse au problème de sélection d'une solution représentant un bon compromis entres les différents critères. Une prise de décision humaine est donc nécessaire pour gérer les interdépendances entre les objectifs. Selon la façon suivant laquelle, les deux aspects sont combinés, trois approches principales sont possibles [TAL01].

1. Aide à la décision à priori :

Cette approche consiste à combiner les différentes fonctions objectifs en une seule fonction dite fonction d'utilité (ou fonction coût) . Ce procédé suggère que le décideur à une connaissance des différents poids des objectifs ainsi que la fonction d'utilité. Le problème est alors traité par un algorithme d'optimisation mono-objectif classique. L'utilisateur définit le compromis qu'il désire avant de lancer la méthode d'optimisation. Toutes les méthodes agrégatives sont des méthodes à préférence à priori.

2. Aide à la décision à posteriori :

Dans ce cas l'algorithme d'optimisation est lancé en premier générant plusieurs solutions. Le décideur choisit alors parmi celles-ci la solution qui lui convient le plus.

3. Aide à la décision interactive :

Une coopération entre le solveur et le décideur est installée. A partir des connaissances acquises lors de la recherche, le décideur formule ces préférences. Ces dernières sont introduites par le solveur ultérieurement. Ce processus est réitéré plusieurs fois.

## III-4 Approches de résolution :

Les méthodes d'optimisation multiobjectif peuvent être classées selon le schéma suivant [TAL01] :

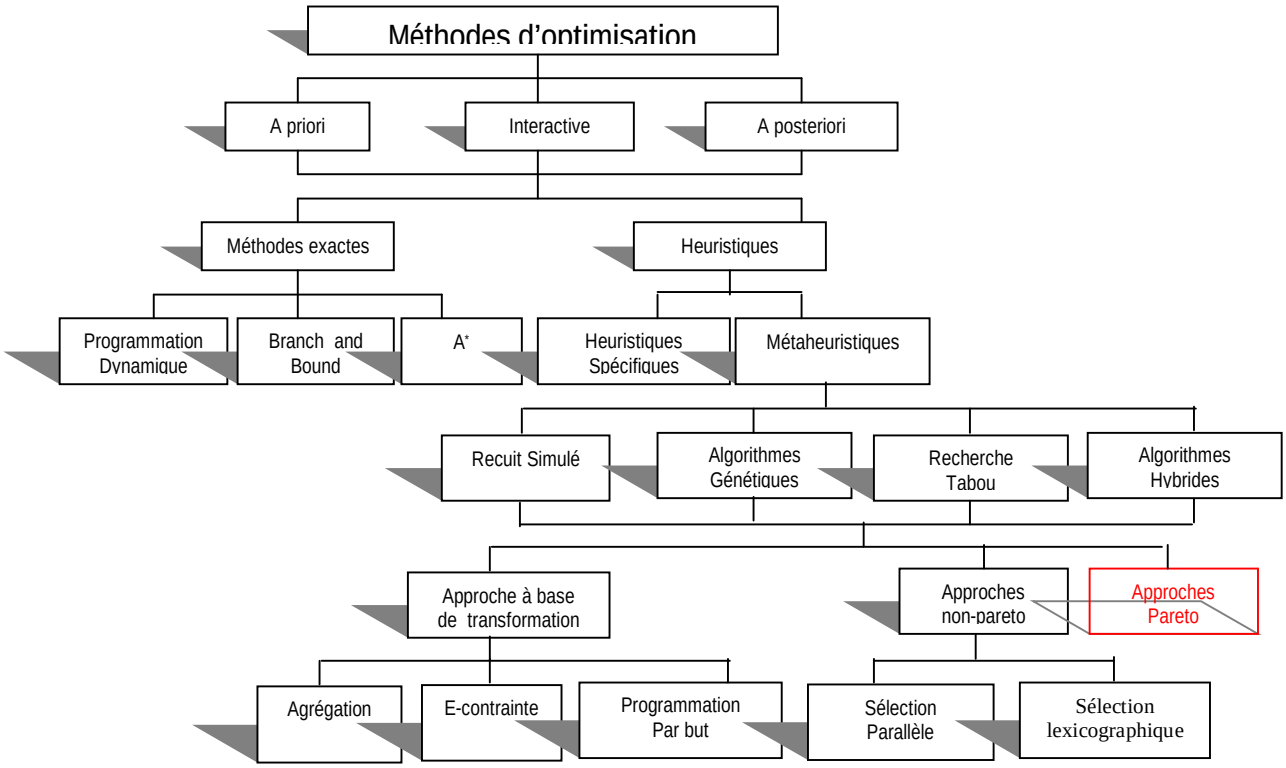



Fig 3.4 Classification des méthodes d'optimisation multiobjectif

Dans la littérature, on distingue deux classes de méthodes différentes :

- Les méthodes exactes telles que le Branch & Bound, l'algorithme A* et la programmation dynamique ont subit des révisions afin de les adapter au cas multiobjectif. Cependant ces approches sont destinées à des problèmes de petite taille. Elles permettent théoriquement de trouver une solution optimale grâce à un parcours exhaustif. Leur efficacité est mise en question dès que la taille du problème augmente et que le nombre de critères retenus croît. Elles ne sont pas adaptées à des problèmes NP-difficiles.

- Les méthodes heuristiques tentent d'approcher l'ensemble des solutions optimales. Elles s'appuient sur les connaissances du domaine (heuristiques spécifiques) ou sur des algorithmes (métaheuristiques). Ces dernières sont applicables à une large gamme de problèmes et leur efficacité demeure relativement bonne quand la taille et le nombre d'objectifs augmentent.

### III-4-1 Les méthodes exactes :

#### III-4-1-1 Algorithme A* :

Stewart et White proposèrent une version multiobjectif de l'algorithme A* (A*MO). L'algorithme A* est maintenu dans sa forme originale. Le poids d'un arc (u,v)

dans le graphe de recherche, correspond à un tuple ($c_1$,$c_2$,...,$c_n$) où $c_i$ désigne le coût relatif à l'objectif $f_i$ induit par le passage de u à v. le coût d'un chemin, correspond à la somme vectorielle des poids des arcs qui le composent. Les fonctions k*, g*, h* prennent une forme non scalaire où k*(u,v) désigne l'ensemble des coûts non dominés des chemins reliant u à v, g*(u) représente le coût de la recherche ayant aboutit à la solution intermédiaire u, h*(u) désigne l'ensemble des vecteurs coût non dominés parmi l'ensemble {k*(u,v)/ v∈T} où T représente l'ensemble des états terminaux. Les nœuds du graphe sont triés et parcourus comme pour l'algorithme A* classique. L'ordre de tri est décrit par les relations de dominance entre les vecteurs coût f. pour plus de détail, le lecteur peut consulter [STE91].

III-4-1-2 Programmation dynamique :

La technique de programmation dynamique multiobjectif a été implémentée par Carraway , Morin et Moskowitz pour la résolution du problème de routage dans les réseaux[CAR90].

L'application de la programmation dynamique pour l'optimisation multiobjectif est rare car ceci est difficile quand le nombre d'objectifs à optimiser est élevé (>2). la difficulté est liée au principe de monotoniété exigée par la méthode. Ce principe réclame un grand volume d'espace de stockage qu'il faudra utiliser pour sauvegarder l'ensemble des résultats des étapes antérieures, en plus d'un temps de calcul très élevé.

III-4-2 Les méthodes de résolution approchées :

Trois techniques sont élaborées pour la résolution approchée : on procède soit par transformation du problème en un problème mono-objectif, soit on se basant sur la notion d'optimalité Pareto ou finalement par traitement séparé des différents objectifs(non Pareto).

III-4-2-1 Approche à base de transformation :

Cette approche procède en transformant le problème du cas multiobjectif vers un problème mono-objectif. Dans ce cas on distingue trois techniques : les techniques d'agrégation, d'E-contrainte et la programmation par but.

a- Méthodes d'agrégation :

Les méthodes d'agrégation fusionnent les différentes fonctions objectives pour ramener le problème à un problème d'optimisation mono-objectif. Plusieurs méthodes existent, la technique la plus utilisée et la plus facile à implémenter, est la méthode de pondération des fonctions objectifs. Dans ce cas, les objectifs sont combinés au sein d'une même fonction coût. La transformation que l'on effectue consiste à prendre chacune des

fonctions objectif, à leur appliquer un coefficient de pondération et à faire la somme des fonctions objectifs pondérées. On obtient alors une nouvelle fonction objectif :

$$\text{Minimiser } f(x) \text{ avec } x \in Sr \Rightarrow \begin{cases} \text{minimiser } \Sigma_{i=1}^{k} \omega_i f_i(x) \\ \text{avec } x \in S_r,\ \omega_i \geq 0 \text{ et } \Sigma_{i=1}^{k} \omega_i = 1 \end{cases} \quad (3.9)$$

Les $\omega_i$, appelés poids, déterminent le degré de priorité d'un objectif donné.
$S_r$ représente le domaine réalisable.

Cette approche a l'avantage évident de pouvoir réutiliser tous les algorithmes classiques dédiés aux problèmes d'optimisation à un seul objectif. C'est souvent la première approche adoptée lorsqu'un chercheur se retrouve devant un nouveau problème multiobjectif.

Cependant cette approche souffre de deux inconvénients :
- le premier est dû au fait que pour avoir un ensemble de points bien répartis sur le Front Pareto, les différents vecteurs $\omega_i$ doivent être choisis judicieusement. Il est donc nécessaire d'avoir une bonne connaissance du problème.
- le deuxième inconvénient provient du fait que cette méthode ne permet pas, de calculer intégralement la surface de compromis lorsque celle-ci n'est pas convexe.

b- Méthodes E-contrainte :

Cette méthode n'est pas une méthode d'agrégation des fonctions objectifs. Elle utilise une autre façon pour transformer un problème d'optimisation multiobjectif en un problème mono-objectif : il s'agit de convertir m-1 des m objectifs du problème en contraintes et d'optimiser séparément l'objectif restant.

Autrement dit l'objectif le plus prioritaire est pris comme unique objectif de l'optimisation. Les autres objectifs sont alors transformés en des contraintes. Donc le problème est converti en un problème mono-objectif dont le but est d'optimiser la fonction objectif $f_p$ la plus prioritaire en contraignant les autres fonctions. A chaque objectif $i \neq p$ est associé une borne $\varepsilon_i$ indiquant la plus mauvaise valeur admise pour la fonction $f_i$. Le problème est alors formulé comme suit :

$$\text{Minimiser } f(x) \text{ Avec } x \in Sr \Rightarrow \begin{cases} \text{Minimiser } f_p(x) \\ \text{Avec } f_i(x) \leq \varepsilon_i,\ \forall\, i \in [1..n],\ i \neq p \\ x \in Sr \end{cases} \quad (3.10)$$

Cette méthode est autant plus intéressante quand l'utilisateur désir privilégier un objectif donné. Par contre , elle perd de son intérêt quand le nombre d'objectifs est élevé. Aussi d'autres inconvénients surgissent à savoir :

- la connaissance à priori des intervalles appropriés pour les valeurs $\varepsilon_i$ , est exigée. Ceci induit le calcul des bornes inférieures des différents objectifs.
- les contraintes rajoutées compliquent la résolution du problème.

c- Programmation par but :

Cette technique se base sur une définition a priori d'un ensemble de buts qu'on espère atteindre pour chaque fonction objectif[COE98]. L'algorithme tente de minimiser l'écart entre la solution courante et ses buts. Cette méthode transforme donc le POMO en un problème d'optimisation mono-objectif de la manière suivante :
- on choisit un vecteur de fonctions objectifs initial B,
- on choisit aussi une direction de recherche $\omega$ (vecteur de coefficients de pondération des fonctions objectifs),
- on cherche ensuite à minimiser un coefficient scalaire $\lambda$ qui représente l'écart par rapport à l'objectif initial B que l'on s'est fixé.
On obtient alors le problème qui s'écrit comme suit :

$$\begin{aligned} &\text{Minimiser} && \lambda \\ &\text{Tel que} && f_1(x) - \omega_1.\lambda \le B_1 \\ & && \quad . \\ & && \quad . \\ & && f_m(x) - \omega_m.\lambda \le B_m \\ &\text{et que} && g(x) \le 0 \\ &\text{avec} && x \in \Re^n,\ f(x) \in \Re^m,\ g(x) \in \Re^p \end{aligned} \qquad (3.11)$$

Ce problème est donc un problème d'optimisation mono-objectif, la fonction objectif à minimiser est le scalaire $\lambda$

Cette approche est efficace et facile à implémenter, cependant elle est sensible au choix des paramètres $\omega$ et B (qui doivent être bien choisis par l'utilisateur), un choix arbitraire de ceux-ci peut conduire à des résultats non cohérents.

III-4-2-2 Approche par traitement séparé des objectifs (approche non Pareto) :

Cette approche consiste à réaliser la recherche en traitant les différents objectifs séparément. Dans les algorithmes génétiques, la prise en compte des différents objectifs apparaît au niveau de la phase de sélection (sélection parallèle, sélection lexicographique). Parmi ces approches on peut citer :

a) Sélection parallèle (VEGA) (Vector Evaluated Genetic Algorithm) :

Cette méthode permet de traiter un problème d'optimisation multiobjectif sans avoir à agréger les fonctions objectifs en une seule[COE98]. Elle a été proposée en 1985 par Schaffer comme une extension d'un algorithme génétique simple pour la résolution d'un problème multiobjectif [SCH85]. La seule différence avec un algorithme génétique simple est la manière dont s'effectue la sélection. L'idée est simple. Si nous avons k objectifs et une population de n individus, une sélection de n/k individus est effectuée pour chaque objectif. Ainsi k sous-populations vont être créées, chacune d'entre elles contenant les n/k meilleurs individus pour un objectif particulier. Les k sous-populations sont ensuite mélangées afin d'avoir une nouvelle population de taille n. le processus se termine par l'application des opérateurs génétiques de modification (croisement et mutation).

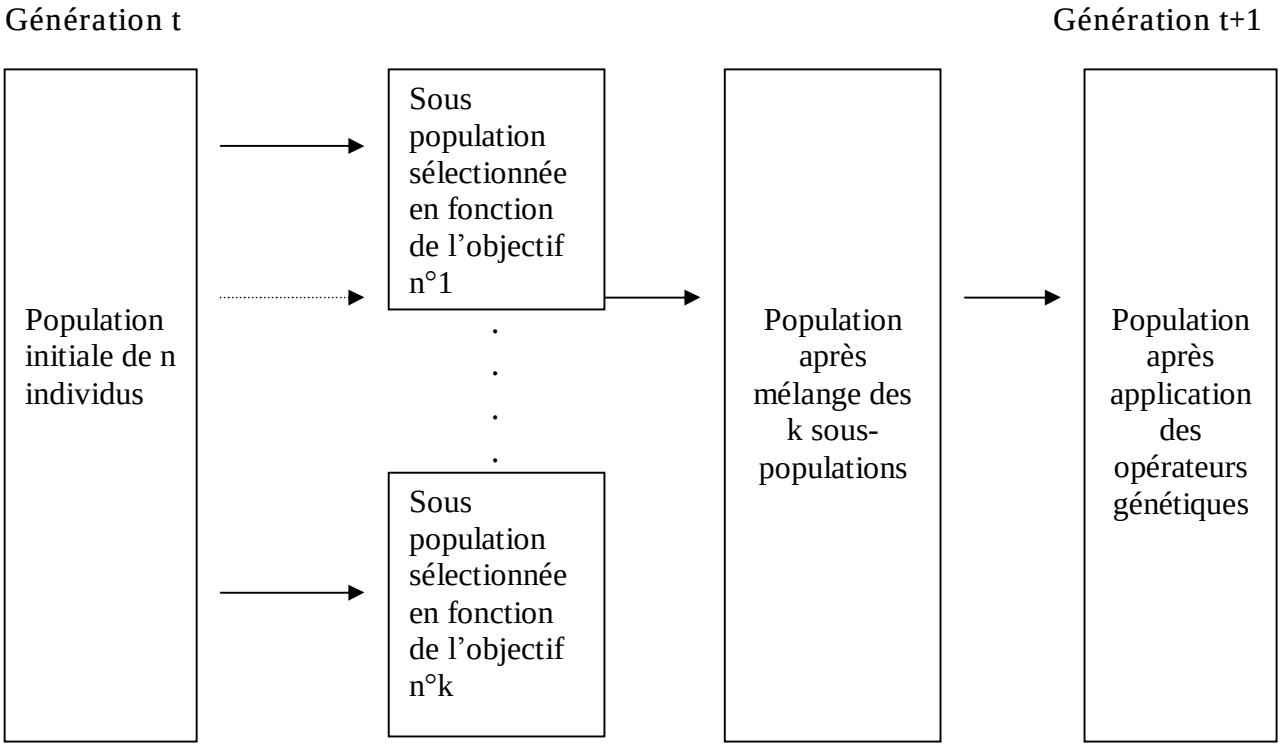



Fig 3.5 Schéma de fonctionnement de VEGA

La méthode VEGA a tendance à créer des sous-populations dont les meilleurs individus sont spécialisés pour un objectif particulier. L'évolution de la population favorise l'apparition des espèces. En effet, comme la méthode de sélection ne tient compte que d'un seul objectif, elle privilégie les individus qui obtiennent une bonne performance pour cet objectif. Dès lors ces individus ne seront sélectionnés que lorsqu'on effectuera la sélection sur cet objectif. Les individus que Schaffer appelle les individus « milieu », parce qu'ayant une performance générale acceptable mais ne possédant aucun critère fort, vont être éliminés car ils ne seront sélectionné dans aucune sous-population. Cette disparition

entraîne la spécialisation des individus pour chaque objectif. Ce résultat est contraire au but initial de la méthode qui était de trouver un compromis entre les différents critères.
Schaffer propose deux heuristiques pour améliorer sa méthode :

- La première est un croisement restreint qui ajoute une préférence pour sélectionner les parents non dominés. Cette méthode a tendance à éviter la disparition des individus « milieu » mais elle a tendance également à accentuer la convergence.

- La seconde encourage le croisement entre individus spécialisés sur des objectifs différents. Mais les effets sont identiques à la première heuristique.

Malgré ces imperfections, cette méthodes est très souvent utilisée car facilement implémentable dans un algorithme génétique classique.
De nombreuses variations autour de cette technique ont été effectuée :

- Mélange de VEGA avec dominance de Pareto[TAN95].
- Application à un problème contraint [SUR95].

b) L'ordonnancement lexicographique :

Cette méthode consiste à considérer les fonctions objectifS les unes après les autres et à minimiser à chaque fois un problème mono-bjectif, en complétant au fur et à mesure l'ensemble des contraintes [COE98].
Cette méthode procède en k étapes :
Soient les fonctions objectifs $f_i$ avec i=1,….,k, supposons un ordre tel que $f_1$ plus important que $f_2$, $f_2$ plus important que $f_3$,…..,$f_{k-1}$ plus important que $f_k$ : il faut :

* Etape 1 :
Minimiser $f_1(x)$
Avec $x \in Sr$
on note $f_1^*$ la solution de ce problème.

* Etape2 :
Minimiser $f_2(x)$
$f_1(x) = f_1^*$
Avec $x \in Sr$
On note $f_2^*$ la solution de ce problème.
...
* Etape k :
Minimiser $f_k(x)$
$f_1(x) = f_1^*, f_2(x) = f_2^*, ..., f_{k-1}(x) = f_{k-1}^*$
Avec $x \in Sr$

La procédure est donc répétée jusqu'à ce que tous les objectifs soient traités. La solution obtenue à l'étape k sera la solution du problème.

L'inconvénient de cette méthode est qu'elle requiert un choix de la séquence des objectifs à minimiser. Ainsi, deux choix différents génèrent deux solutions distinctes.

III-4-2-3 Approche Pareto :

Cette approche s'appuie directement sur la notion d'optimalité Pareto. La notion de dominance est alors utilisée comme moyen de comparaison et d'ordonnancement des solutions. Le principal avantage de cette approche est qu'elle soit capable de générer un ensemble de solutions Pareto appartenant aux parties concaves de la frontière Pareto. Dans les algorithmes génétiques la prise en considération du caractère dominant ou dominé des solutions apparaît dans la phase de sélection.
En effet, il existe deux générations des approches Pareto :

- la première génération est caractérisée par l'utilisation :

- d'un mécanisme de sélection Pareto basé sur la technique de ranking [GOL89], utilisant la relation de dominance pour affecter des rangs aux individus de la population, faisant apparaître la notion de Front.

- du sharing (niching) qui pour éviter qu'un grand nombre d'individus se concentre autour d'un même point, la valeur d'adaptation est pénalisée en fonction du nombre d'individus au voisinage du regroupement[GOL87].

Parmi les approches représentatives de cette génération, on peut citer :

- M.O.G.A (Multi-Objective Genetic Algorithm).
- N.S.G.A (Non dominated Sorting Genetic Algorithm)
- N.P.G.A (Niched-Pareto Genetic Algorith)

- la deuxième génération est composée généralement des approches élitistes manipulant une population secondaire externe.

Parmi les approches représentatives de cette génération, on peut citer :

- SPEA et SPEA-2 (Strength Pareto Evolutionary Algorithm)
- NSGA-II ( Non dominated Sorting Genetic Algorithm)
- PAES ( Pareto Archived Evolution Strategy)
- PESA et PESA-II ( Pareto Envelope-based Selection Algorithm)
- MOMGA et MOMGA-II ( Multi-Objective Messy Genetic Algorithm)
- MicroGA pour l'optimisation multi-objectif.

## III-5 Métaheuristiques et optimisation multiobjectif :

Les métaheuristiques sont des méthodes très performantes, elles ont été appliquées avec succès à un grand nombre de problèmes. Les principales heuristiques abordées sont : le recuit simulé, la recherche tabou et les algorithmes génétiques.

### III-5-1 Recuit simulé multiobjectif :

L'utilisation du recuit simulé dans le cadre de l'optimisation multiobjectif a premièrement été étudiée par Serafini [SER92]. L'idée principale dans l'application du recuit simulé pour l'optimisation multiobjectif consiste en l'utilisation d'une norme pondérée des composants du vecteur objectif, afin d'accepter ou rejeter une solution de coût inférieur.

Règle de calcul des probabilités de transition :

En fonction de la manière de calculer la probabilité d'acceptation d'une solution, deux approches sont utilisées. Dans la première approche dite à critère d'acceptation fort, seule les solutions dominantes sont sûres d'être acceptées. La deuxième approche dite à faible critère d'acceptation adoucit cette contrainte en acceptant directement les solutions non dominées(une solution non dominée n'est pas obligatoirement dominante).

1. Trie agrégatif :

Ce mécanisme s'apparente aux techniques agrégatives où les différents critères sont combinés au sein de la même fonction objectif globale. Delà le principe d'acceptation classique est utilisé.

- Règle Scalairisation linéaire : la probabilité d'acceptation d'une solution y partant de la solution x à la température T est :

$P(y,T) = e^{\Sigma\omega_{i}( fi(x)-fi(y) )/T}$

Dans [TUY98], l'auteur propose cette technique pour la résolution du problème du sac à dos multiobjectif.
Dans [ULU98], l'approche proposée utilise cette technique pour rechercher la surface de compromis.

- Règle de programmation par but :

$P(y,T) = e^{\max wi(fi(x)-ri) - \max wi(fi(y)-ri)/T}$ avec $i \in [1..n]$

Le vecteur $(r_1, r_2,\ldots r_n)$ représente le vecteur idéal à approcher.

2. Trie Pareto :

Le principe d'acceptation est révisé pour introduire la notion d'optimalité Pareto. Dans les approches à critère d'acceptation fort, le critère d'acceptation peut être formulé sous la forme suivante :

- Règle du produit simple :

$$P(x,T) = \prod \min \left\{ 1, e^{(fi(x)-fi(y))/T} \right.$$

- Règle de Cebicev :

$$P(x,T) = \min \left\{ 1, \min_{i \in [1..n]} e^{(fi(x)-fi(y))/T} \right.$$

Avec l'approche à critère d'acceptation faible, toute solution qui n'est pas strictement dominée serait acceptée.

- Règle de critère faible :

$$P(x,T) = \min \left\{ 1, \max_{i \in [1..n]} e^{(fi(x)-fi(y))/T} \right.$$

III-5-2 Recherche Tabou multiobjectif :

Les premiers travaux portants sur l'utilisation de la Recherche Tabou pour l'optimisation multiobjectif opéraient par transformation vers le mono-objectif. Des approches ultérieures, étendent les principes de la Recherche Tabou afin de produire une bonne approximation de la frontière Pareto.

Dans [HER94], on se base sur une approche lexicographique. la recherche tabou est utilisée pour résoudre le problème de formation de cellules de fabrication dans les ateliers. La méthode est appliquée pour optimiser une séquence de sous-problèmes mono-objectif sous contraintes. Les objectifs sont traités de façon séquentielle suivant l'ordre d'importances des critères. Le premier sous-problème résolu consiste à optimiser la fonction objectif la plus prioritaire. Le deuxième problème revient alors à optimiser la deuxième fonction objectif la plus importante. Ceci en maintenant satisfaite une contrainte supplémentaire qui consiste à ne pas détériorer, avec une certaine marge, la valeur obtenue pour la première fonction. Les autres critères sont ainsi traités en réitérant ce procédé.

Dans [GAN97], se basant sur une approche de programmation par but, ils proposèrent l'application de la recherche tabou pour des problèmes d'optimisation où les préférences de l'utilisateur sont connues dès le départ. Ces préférences sont alors formuler

sous forme d'un vecteur idéal F*($f1$*,$f2$*,...$f$n*). L'objectif est alors de minimiser la fonction objectif globale $f$ suivante :

Minimiser $f(x)= [F(x) - F^*(x)]_{p,\omega} = \left( \sum_{j=1..n} \left[ (\omega_j)^p \ (f_j(x) - f_j^*)^{\,p} \right] \right)^{1/p}$

Les poids $\omega_j$ sont introduits pour opérer un changement d'échelle ou pour signaler l'importance de certains critères par rapport à d'autres.

Lors de chaque itération, la solution voisine du point courant ayant la plus petite valeur de $f$ et qui n'est pas tabou est choisie comme nouvelle solution courante.

L'objectif principal de l'algorithme est de générer une bonne approximation de l'ensemble Pareto optimal. A cet effet, la liste des M meilleurs solutions voisines est considérée ($1 \leq M \leq | V(x) |$ ). A chaque itération, ces M solutions sont comparées aux différentes solutions présentes dans la population Pareto. Les solutions dominées de la population Pareto dans la liste sont éliminées. Si une solution de la liste non dominée dans la population Pareto elle est insérée dans celle-ci.

D'autres versions multiobjectif de la recherche tabou ont été proposées [BEN97], [HAN98], [BAR03].

III-5-3 Algorithmes Génétiques multi-objectif :

Les algorithmes génétiques sont très bien adaptés au traitement des problèmes d'optimisation multiobjectif.

Goldberg [GOL89] indiquait que la notion de recherche génétique est multiobjectif dans sa nature. Cependant, les algorithmes génétiques requièrent une connaissance scalaire sur l'adéquation des solutions. La formulation du problème comme optimisation d'un critère global n'est souvent pas évidente. La conversion du problème par utilisation d'une somme pondérée des objectifs offre l'avantage de la simplicité d'implémentation. En plus, une seule solution est fournie, éliminant le besoin du décideur à intervenir. Cette technique reste inefficace puisqu'un choix inapproprié des poids induira un temps d'exécution additionnel jusqu'à ce qu'une solution convenable soit trouvée. Plusieurs algorithmes génétiques multiobjectif ont été proposés dans la littérature, on peut citer :

- La méthode M.O.G.A (Multiple Objective Genetic Algorithm) :

En 1993 Fonseca et Flemming on proposé une méthode dans laquelle chaque individu de la population est rangé en fonction du nombre d'individus qui le dominent [FON93]. Ensuite , ils utilisent une fonction de notation permettant de prendre en compte le rang de l'individu et le nombre d'individus ayant même rang.

Soit un individu $x_i$ à la génération t, dominé par $p_i(t)$ individus. Le rang de cet individu est :

$$Rang(x_i,t) = 1+p_i(t)$$

Tous les individus non dominés sont de rang 1.

Exemple :
Soient les points 1,3 et 5 ne sont dominés par aucun autre point. Alors que le point 2 est dominé par le point 1, et que le point 4 est dominé par les points 3 et 5.

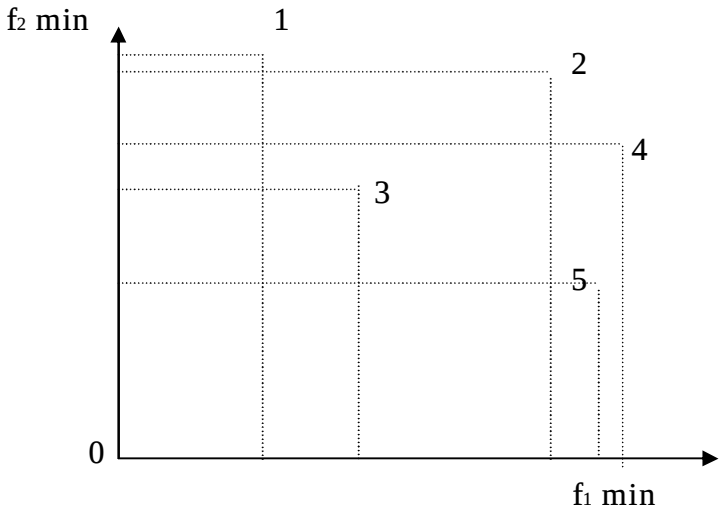



Fig 3.6 Exemple de dominance

Le tableau de dominance est le suivant :

| Point | Dominé par | Rang |
|---|---|---|
| 1 | Aucun | 1 |
| 2 | 1 | 2 |
| 3 | Aucun | 1 |
| 4 | 3 et 5 | 3 |
| 5 | Aucun | 1 |

Tableau 1 : tableau de dominance

Fonseca et Fleming calculent la fitness de chaque individu de la façon suivante :
1- Calcul du rang de chaque individu
2- Affectation de la fitness de chaque individu par application d'une fonction de changement d'échelle (on parle aussi de fonction de scaling) sur la valeur de son rang. Cette fonction est en général linéaire. Suivant le problème. D'autres types de fonction pourront être envisagés afin d'augmenter ou de diminuer l'importance des meilleurs

rangs ou d'atténuer la largeur de l'espace entre les individus de plus fort rang et de plus bas rang.

L'utilisation de la sélection par rang a tendance à répartir la population autour d'un même optimum. Or cela n'est pas satisfaisant pour un décideur car cette méthode proposera qu'une seule solution. Pour éviter cette dérive, les auteurs utilisent la fonction du sharing. Ils éspèrent ainsi répartir la population sur l'ensemble de la frontière Pareto.

Cette méthode obtient des solutions de bonne qualité et son implémentation est facile. Mais les performances sont dépendantes de la valeur du paramètre utilisé dans le sharing.

Algorithme de la méthode M.O.G.A :

Initialisation de la population
Evaluation des fonctions objectif
Assignation d'un rang basé sur la dominance
Assignation d'une efficacité à partir du rang

Pour i=1 to genmax (nombre maximal de générations)
    Sélection (aléatoire proportionnelle à l'efficacité)
    Croisement
    Mutation
    Evaluation des fonctions objectif
    Assignation d'un rang basé sur la dominance
    Assignation d'une efficacité à partir du rang
Finpour

- La méthode N.S.G.A (Non dominated Sorting Genetic Algorithm)

Cette méthode est dû à Srinivas et Deb [SRI93] , elle se base sur le calcul d'efficacité des individus comme suit :
Dans un premier temps, on affecte à chaque individu un rang de Pareto. Tous les individus de même rang sont classés dans une catégorie. Chaque catégorie aura une efficacité inversement proportionnelle à son rang.
Pour maintenir la diversité de la population, la valeur de l'efficacité d'une catégorie est partagée entre ses individus de la façon suivante :

1. On calcule, pour chaque individu, le compte de ses voisins

$$m_i = \sum_{j=1}^{K} Sh(d(i,j))$$

$$Sh(d(i,j)) = \begin{cases} 1-(\frac{d(i,j)}{s_{share}})^2 & si \quad d(i,j) < s_{share} \\ 0 & \text{sin } on \end{cases}$$

avec K :nombre d'individus de la catégorie.

$\delta_{share}$ : permet de définir une zone d'influence pour le calcul de l'efficacité de l'individu.

2. L'efficacité $f_i$ de l'individu i est donnée par : $f_i = \frac{F}{m_i}$

La méthode NSGA est une méthode efficace, qui permet d'obtenir une diversité dans la représentation des solutions, cependant elle est très sensible au choix du diamètre de la niche $\delta_{share}$ . Récemment une nouvelle version améliorée de NSGA appelée NSGA-II est apparue [DEB00]. Cette dernière intègre un opérateur de sélection, basé sur un calcul de la distance de " crowding ", très différent de celui de NSGA. En comparaison avec NSGA, NSGA-II obtient de meilleurs résultats, ce qui fait de cet algorithme un des plus utilisés aujourd'hui.

Ø La méthode S.P.E.A (Strength Pareto Evolutionary Algorithm) :

La méthode SPEA réalisée par Zitzler et Thiele [ZIT98] est l'illustration même d'un algorithme évolutionnaire élitiste. Cette méthode est basée sur des techniques afin de générer plusieurs solutions Pareto optimales en parallèle :

- Une population additionnelle est maintenue, elle archive les solutions Pareto trouvées tout au long de la recherche.
- Le concept de dominance est utilisé afin d'associer une valeur d'adéquation scalaire aux individus.
- Un clusturing des solutions archivées est réalisé afin de réduire leur nombre sans perdre la forme de la frontière Pareto.
- Les solutions archivées participent eux aussi à la phase de sélection.

Tout d'abord, la population Pareto (archive) est mis à jour. Ce qui revient à copier l'ensemble des solutions non dominées de la population dans l'ensemble Pareto et à supprimer éventuellement les éléments dominés de celle-ci. Quand le nombre de solutions archivées dans l'ensemble Pareto dépasse une certaine limite, une représentation réduite est déduite par échantillonnage. Après l'évaluation des individus, la sélection par tournoi est effectuée sur les deux populations (population courante et population Pareto) afin de produire la population intermédiaire P'. les opérateurs de croisement et de mutation sont alors appliqués.

Algorithme de la méthode SPEA

Initialiser la population $P_0$ et créer l'archive externe vide $\overline{P} = \phi$
Mise à jour de $\overline{P}$ à partir des individus non dominés de $P_0$
Tant que critère- d'arrêt- non- rencontré faire
* Calcul de la valeur d'adaptation pour tous les individus de P + $\overline{P}$
* Sélection dans $P_t$ + $\overline{P}$ en fonction de la valeur d'adaptation
* Croisement
* Mutation
* Mise à jour de à partir des individus non dominés de P
FinTant que

La population Pareto (l'archive) correspond à l'ensemble des solutions élites. Cet ensemble contient les solutions non dominées trouvées pendant la recherche. L'évaluation d'un individu est déterminée en fonction des individus de la population courante et de la population Pareto.

Dans certains problèmes, l'ensemble Pareto optimale peut être extrêmement large. Dans ce cas la limitation de la taille de cet ensemble est nécessaire à cause des points suivants :

- Du point de vue du décideur, la présentation de toutes les solutions Pareto trouvées est inutile quand leur nombre excède une certaine limite.

- Dans des cas d'optimisation de fonction continue, il est physiquement impossible et non nécessairement désirable de fournir toutes les solutions Pareto.

- Quand le nombre de solutions Pareto archivées est grand, ils tendent à avoir le même rang, entraînant une plus faible pression de sélection et du coup une lenteur de la recherche.

Une nouvelle version améliorée de la méthode SPEA existe sous le nom de SPEA-II [ZIT01].

**Ø** La méthode PAES (Pareto Archived Evolution Strategy) [KNO99] :

Cette méthode a été développée initialement comme une méthode de recherche locale dans un problème de routage d'information off-line. Les premiers travaux de Knowles et Corne ont montré que cette méthode simple objectif fournissait des résultats supérieurs aux méthodes de recherche basées sur une population. Par conséquent, les auteurs ont adapté cette méthode aux problèmes multiobjectifs. Les particularités de cette méthode sont les suivantes :

- § Elle n'est pas basée sur une population. Elle n'utilise qu'un seul individu à la fois pour la recherche des solutions
- § Elle utilise une population annexe de taille déterminée permettant de stocker les solutions temporairement Pareto-optimales.
- § Elle utilise une technique de crowding basée sur un découpage en hypercubes de l'espace des objectifs.

L'algorithme de PAES se présente en trois parties :

1) Génération d'une solution candidate.
2) Fonction d'acceptation de la solution candidate.
3) Archivage des solutions non dominées.

La méthode de génération d'un nouveau candidat ressemble à la méthode de Hill-Climbing. A chaque itération un nouveau candidat est produit par mutation aléatoire.

a) Génération aléatoire d'une solution c et ajout de c à l'archive.
b) Production d'une solution m par mutation de c et évaluation de m.
c) Si (c domine m)
   Alors on écarte m.
   Sinon si (m domine c)
      Alors on remplace c par m et ajout de m à l'archive.
      Sinon si (m est dominé par un membre de l'archive)
         Alors on écarte m
         Sinon on applique une fonction de test (c, m, archive) qui détermine la nouvelle solution courante.
d) On recommence en b).

Fonction de test (c, m, archive)

a. Si l'archive n'est pas pleine
   Alors ajout de m à l'archive.
   Sinon si (m est dans une région moins encombrée qu'une solution x ∈ à l'archive )
      Alors ajout de m et suppression d'un membre de la zone la plus encombrée de l'archive.
b. Si (m est dans une région moins encombrée que c)
   Alors m est acceptée comme solution courante.
   Sinon on conserve c comme solution courante.

Pour mesurer l'encombrement d'une zone, cette méthode utilise un crowding basé sur un découpage en hypercubes de l'espace des objectifs. Cette technique offre deux avantages par rapport aux méthodes de sharing classiques :

- Le temps de calcul est moins important,
- Le découpage étant adaptatif, cela ne nécessite pas de réglage de paramètre.

Cette méthode est relativement simple à mettre en œuvre. De plus , n'étant pas basée sur un algorithme génétique, elle évite à l'utilisateur le réglage de tous les paramètres de

celui-ci. Mais son efficacité va dépendre du choix d'un nouveau paramètre : le paramètre de discrétisation de l'espace des objectifs.
La technique de crowding utilisée dans PAES permet une mise à jour de l'archive plus rapide lors des dépassements de capacité que celle de SPEA.

Ø La méthode PESA (Pareto Envelope-based Selection Algorithm) :

La méthode PESA a été également proposée par Corne et Knowles[COR00]. Elle reprend approximativement le principe de crowding développé dans PAES et définit un paramètre appelé ***squeeze_factor*** qui représente la mesure d'encombrement d'une zone de l'espace. Alors que PAES est basée sur une stratégie d'évolution, PESA est une méthode basée sur les algorithmes génétiques. Elle définit deux paramètres concernant la taille des populations d'individus : $p_I$ (taille de la population interne) et $P_E$ (taille de la population externe ou archive).

L'algorithme de PESA se présente comme suit :

a) Génération aléatoire et évaluation de la population $P_I$, et initialisation de $P_E$ ($P_E = \emptyset$ ).
b) Transfert de tous les individus non dominés de $P_I$ dans $P_E$
c) Si le critère d'arrêt est réalisé
   Alors on retourne $P_E$ comme ensemble de solutions
   Sinon on supprime tous les individus de $P_I$ et on recrée $P_I$ de la façon suivante :
      On sélectionne deux parents dans $P_E$ avec la probabilité $p_c$, on produit un enfant par croisement puis on le mute. Avec la probabilité (1- $p_c$), on sélectionne un parent et on le mute pour produire un autre enfant.
d) On recommence au b)

Une solution courante de $P_I$ peut entrer dans l'archive $P_E$ si elle est non dominée dans $P_I$ et si elle est non dominée dans $P_E$. Une fois, la solution insérée dans l'archive, on supprime tous les membres de l'archive qu'elle domine. Si l'ajout crée un dépassement de capacité de $P_E$ alors le membre de l'archive ayant le paramètre ***squeeze_factor*** le plus élevé est supprimé.
Le paramètre ***squeeze_factor*** est égal au nombre d'individus qui appartiennent au même hypercube. Il est utilisé comme fitness des individus qui appartiennent à cette zone.

D'autres méthodes basées sur les algorithmes génétiques multiobjectifs existent, exemple :

* La méthode MOMGA ( Multi-objective Messy Genetic Algorithm)[VAN00],
* La méthode MSGA (Algorithme Génétique Multi-sexuelle) [LIS96],...

En effet, les algorithmes génétiques semblent être les plus adaptés aux problèmes d'optimisation multiobjectif. Ceci est dû aux mécanismes pertinents utilisés pour approcher la surface de compromis. Ces mécanismes son les suivants :

- Un mécanisme de séléction Pareto :

La sélection Pareto utilise la relation de dominance pour affecter des rangs aux individus de la population, faisant apparaître la notion de front. Par exemple la technique de ranking[GOL89] : où tous les individus non dominés de la population reçoivent le rang 1 et sont retirés temporairement de la population. Puis, les nouveaux individus non dominés reçoivent le rang 2 avant d'être à leur tour retirés. Le processus s'itère tant qu'il reste des individus dans la population. La valeur d'adaptation de chaque individu correspond à son rang dans la population. Ainsi, l'évaluation d'un individu ne dépend pas uniquement de lui même, mais aussi de la population. Ce principe a l'avantage de ne pas hiérarchiser les objectifs entre eux. Mais l'augmentation de la taille de l'espace de recherche peut influencer la performance de cette sélection.

- L'élitisme :

L'élitisme permet de conserver les meilleurs individus dans les générations futures. Il est introduit pour conserver les bonnes solutions lors du passage de la génération courante à la prochaine génération. Conserver ces solutions pour les générations futures permet d'améliorer les performances des algorithmes sur certains problèmes. Réaliser un algorithme élitiste dans le cadre des problèmes multiobjectifs est plus difficile que dans le cadre mono objectif. En effet, la meilleure solution n'est plus un individu unique mais tout un ensemble dont la taille peut même dépasser celle de la population. Deux adaptations du mécanisme élitiste sont considérées : la première approche regroupe les algorithmes qui conservent pour les générations futures les k meilleurs individus. Mais comment avec cette approche sélectionner seulement k individus des solutions non dominées sans perdre une partie du front de Pareto. Les approches récentes, comme SPEA par exemple, utilisent une population externe d'individus dans laquelle est stocké le meilleur ensemble des point non dominés découverts jusqu'ici. Cet ensemble est mis à jour continuellement pendant la recherche, et les individus stockés continuent à pouvoir être choisis par l'opérateur de sélection. Ils peuvent ainsi se reproduire et transmettre leurs caractéristiques aux générations suivantes.

- Maintenir la diversité :

Dans le cas des problèmes multiobjectif, converger prématurément rend impossible la découverte de l'intégralité du front Pareto. En effet, les individus de la population se focaliseront sur une partie du front Pareto, et ne se répartiront pas sur la totalité de ce front Pareto.

Pour remédier à ce problème, de nombreuses techniques ont été développées afin de maintenir un certain degré de diversité, on peut citer :

- Le sharing :

Le sharing consiste à modifier la valeur de coût d'un individu. C'est cette nouvelle valeur qui sera utilisée comme valeur d'adaptation par l'opérateur de sélection.

Pour éviter qu'un grand nombre d'individus ne se concentrent autour d'un même point, il faut pénaliser la valeur d'adaptation en fonction du nombre d'individus au voisinage du regroupement : plus les individus sont regroupés, plus leur valeur d'adaptation est faible, et les individus proches les uns des autres doivent partager leur valeur d'adaptation. Dans la pratique, on estime ce taux de concentration en déterminant un domaine autour d'un individu, puis on calcule les distances entre les individus contenus dans ce domaine.

Pour déterminer les bornes du domaine, appelé aussi niche, on définit une distance maximale, appelée $s_{share}$, au-delà de laquelle les individus ne seront plus considérés du domaine. La distance séparant deux individus i et j est calculée grâce à la fonction d(i,j). la valeur d'adaptation F(i) d'un individu i ∈ P(population) est égale au coût F'(i) divisé par sa valeur de niche :

$$F(i) = \frac{F'(i)}{\sum_{j \in P} Sh(d(i,j))}$$

Où la fonction Sh est définie par :

$$Sh(d(i,j)) = \begin{cases} 1-(\frac{d(i,j)}{s_{share}})^2 & si \quad d(i,j) < s_{share} \\ 0 & \text{sinon} \end{cases}$$

- La réinitialisation :

La réinitialisation est une technique largement utilisée par toutes les métaheuristiques. Cette technique consiste à introduire de manière régulière certains individus générés aléatoirement. Ainsi de nouvelles zones de l'espace de recherche inexplorées peuvent être explorées.

- Le crowding :

L'approche par crowding consiste à désigner des représentants pour les niches. A la différence du sharing où tous les individus sont susceptibles d'être sélectionnés et de participer aux phase de croisement, mutation et sélection, avec le crowding , seulement les représentants des différentes niches peuvent participer aux différentes phases de l'algorithme.

- Le clustering

Généralement, on fait appel à la technique de clustering Lorsque la taille de l'archive externe dépasse une certaine limite, car dans ce cas les performances de l'algorithme peuvent se dégrader significativement. Le clustering utilisé par SPEA, par

exemple, consiste à considérer initialement chaque individu comme son propre groupe, puis fusionne deux à deux les groupes les plus proches en terme de distance. Cette étape est itérée jusqu'à obtention du nombre de groupes désiré. Ensuite un représentant par groupe est choisi. Ce représentant peut être, par exemple, le barycentre du groupe. Enfin, chaque groupe est réduit à son représentant.

III-5-4 Les méthodes hybrides multiobjectifs :

Le principe général de l'hybridation consiste à combiner un algorithme avec une autre méthode dont l'objectif est d'améliorer ses performances. Généralement, on combine un algorithme génétique avec une méthode de recherche locale où on substitue souvent la mutation par une méthode de recherche locale. Parmi les méthodes hybrides multiobjectif, on cite :

- La méthode MOTS[HAN97]
- La méthode PSA[CZY98]

III-5-5 Autres métaheuristiques :

Plusieurs autres métaheuristiques on été adapté au POMO telles que :
Les colonies de fourmis [IRE01], les réseaux de neurones artificiels[CUI01], les systèmes immunitaire [COE02],...

III-6 Evaluation des méthodes d'optimisation multiobjectif :

Comme les méthodes d'optimisation multiobjectif sont en nombre important, il est indispensable de disposer des outils nécessaire pour leur comparaison (évaluation des surfaces de compromis). Plusieurs métriques sont utilisées :

- L'hypervolume et le rapport d'hypervolume [ZIT99] : elle permet de mesurer le volume compris sous la courbe formée par les points de l'ensemble à évoluer. Plus la valeur de cette métrique est proche de 1, plus la surface de compromis est meilleure.

- La métrique C: elle est introduite par Zitzler permettant de comparer deux surfaces de compromis en transformant ces deux surfaces en un nombre réel compris entre 0 et 1[ZIT99].

- La métrique d'espacement [SCH95b] : elle permet de mesurer l'uniformité de la répartition des points sur la surface de compromis. Plus le résultat de la mesure est proche de 1, meilleure est la répartition des points sur la surface de compromis.

- D'autres métriques sont aussi utilisées [COL02].

## III-7 Conclusion :

Dans ce chapitre, nous avons présenté les concepts de base de l'optimisation multiobjectif, puis une classification des techniques d'optimisation multiobjectif qui fait intervenir trois aspects. D'une part, le type d'interaction entre algorithme de recherche et décideur (aspect aide à la décision). D'autre part, l'objectif de la recherche elle-même : recherche exacte de l'ensemble des solutions ou recherche approchée de cet ensemble. Le troisième aspect, concerne l'utilisation ou non de la notion d'optimalité Pareto lors de la recherche. Les approches agrégatives ainsi que les approches non Pareto semblent être moins efficaces, car elles sont très sensibles à la forme du front Pareto (concavité, non uniformité,...). Cependant les méthodes Pareto issues des métaheuristiques pour l'optimisation multiobjectif semblent bien se prêter à ce type de problèmes. Parmi les métaheuristiques utilisées, les techniques de recherche à base de population telles que les algorithmes génétiques semblent les plus adéquates. En effet, l'utilisation d'une population permet d'approximer par une seule exécution l'ensemble des solutions Pareto contrairement aux techniques de recherche locale (recuit simulé et recherche tabou) où il faut exécuter la recherche plusieurs fois afin d'obtenir un ensemble de solutions Pareto. Nous avons aussi fait la distinction entre les techniques élitistes et les techniques non élitistes dans les approches Pareto. Les techniques non élitistes ne conservent pas les individus Pareto-optimaux trouvés au cours du temps, elles maintiennent difficilement la diversité sur la frontière Pareto et la convergence des solutions vers la frontière Pareto est lente. Pour résoudre ces difficultés, de nouvelles techniques ont été appliquées : c'est les techniques élitistes qui permettent l'introduction d'une population externe ou archive permettant de stocker les individus Pareto-optimaux, l'utilisation de techniques de niching et clustering pour répartir efficacement les solutions sur la frontière Pareto et la préférence pour les solutions non dominées.
Nous avons aussi présenter les métriques les plus utilisées pour mesurer la qualité de l'ensemble des solutions non dominées trouvées.

Notre intérêt à porté sur l'étude de l'application des algorithmes génétiques dans le domaine de l'optimisation multiobjectif, vu leur grande efficacité face aux problèmes d'optimisation combinatoire afin de proposer une algorithme génétique multiobjectif pour le problème d'emploi du temps.

# CHAPITRE IV

# PROPOSITION D'UN ALGORITHME GENETIQUE MULTIOBJECTIF POUR LA RESOLUTION DU PROBLEME D'EMPLOI DU TEMPS

Cette partie du travail est consacrée à la présentation du problème d'emploi du temps multiobjectif. Nous discutons les différents paramètres à optimiser ainsi que les contraintes à satisfaire. Nous présentons les choix de codage, de fonctions objectifs, d'opérateurs génétiques et des stratégie de sélection.

## IV-1 Introduction :

Le problème de l'emploi du temps est un problème représentatif d'une famille de problèmes combinatoires discrets. Il renferme un ensemble d'objectifs conflictuels, un ensemble de contraintes non linéaires et un nombre de combinaisons potentielles très élevé.

Dans ce chapitre, le problème de l'emploi du temps sera prouvé que lui aussi est un problème d'optimisation multiobjectif. Jusqu'à maintenant ce sont principalement les modèles mono-objectifs qui sont utilisés lors de la résolution de ce type de problèmes. Ainsi, nous allons concentrer notre travail dans ce qui suit sur l'utilisation d'une approche évolutive multiobjectif. Pour la résolution de ce problème.

## IV-2 Justification du choix d'une approche évolutive multiobjectif :

Les problèmes d'optimisation multiobjectif (POMO) sont une généralisation des problèmes d'optimisation. Le but est de trouver l'ensemble de Pareto optimal comprenant des solutions non dominées. Le problème de l'emploi du temps est fondamentalement un problème d'optimisation multiobjectif. Tout comme la plupart des problèmes d'optimisation, il doit traiter avec plusieurs objectifs. Les techniques d'optimisation mono-objectif quant à elles, assemblent tous les objectifs dans une même fonction. Si l'on suppose un problème d'optimisation défini par minimiser $\{f_1, f_2,\ldots f_m\}$ avec $m > 1$, la fonction d'objectif utilisé par un algorithme mono-objectif sera $\Sigma f_i w_i$, $i=1...m$. le vecteur $W = \{w_1,w_2,\ldots w_m\}$ est le poids qui est accordé à chaque objectif. Ce vecteur de poids ne représente qu'une mesure de préférence. Par contre, si les critères de préférence changent, un nouveau vecteur W doit être identifié et une nouvelle recherche doit être faite. Une autre différence est que dans un problème d'optimisation multiobjectif l'objectif est double. Contrairement aux techniques d'optimisation mono-objectif où le seul objectif à atteindre est la minimisation (ou la maximisation) de la fonction objectif, pour un problème multiobjectif l'obtention de l'ensemble Pareto optimal du problème et la diversité des solutions qui en font partie sont deux objectifs indissociables. Cette diversité est un objectif important, car elle permet d'effectuer une meilleure décision à posteriori. Donc en considérant que le problème de l'emploi du temps est un problème multiobjectif, l'application d'une méthode de résolution multiobjectif semble être logique. Les méthodes évolutives offrent aussi l'avantage d'utiliser une population d'individus. Dans une application multiobjectif il sera donc possible de faire converger l'ensemble de la population vers un front Pareto en effectuant un seul lancement de l'algorithme. Ceci éliminera du même coup toute pondération d'objectifs appliquée à priori.

## IV-3 Description du problème à résoudre :

Dans un établissement éducatif, un ensemble d'étudiants groupés sous une structure hiérarchique (filières, promotions, sections, groupes,…) sont sensés avoir un ensemble d'enseignements qui se répètent périodiquement, chacun de ces enseignements s'étend sur une durée de temps, dont l'unité élémentaire est la période.

Résoudre le problème de l'emploi du temps revient à affecter à chacun de ces enseignements un nombre de périodes consécutives égal à la durée qu'il exige, un local dont le type et la capacité sont convenables, et un enseignant apte à assurer le module concerné par l'enseignement de façon à prévenir les conflits sur les enseignants, sur les étudiants et sur les locaux..

Le problème de l'emploi du temps se manifeste en plusieurs différentes formes dont chacune des formes est spécifique à l'environnement ou à l'institution qui en a besoin. Dans notre cas, le problème de l'emploi du temps étudié est celui d'université (département d'informatique) où les responsables pédagogiques ont besoin chaque année d'établir une nouvelle planification des différentes promotions en essayant au mieux de satisfaire les contraintes « humaines » des enseignants et des étudiants, les contraintes pédagogiques imposées par la progression des enseignements et en tenant compte des contraintes « physiques » liées aux ressources matérielles ( les locaux, les équipements etc…).

Le département d'informatique regroupe différentes formations qui ont une durée qui varie entre trois ans (DEUA) et cinq ans (Ingénieur d'état).
Le programme pédagogique de chaque formation est connu à priori. Ce programme précise les modules à suivre, leurs volumes horaires et quelques informations pédagogiques (répartition en cours, travaux dirigés, travaux pratiques etc…). selon les besoins pédagogiques et les conditions physiques des ressources, chaque formation est structurée en promotions, en sections, et en groupes, le nombre d'étudiants par groupe en travaux dirigés(TD) est limité à 30 pour préserver un meilleur suivi des étudiants. Les enseignants interviennent selon leur discipline et leur domaine de compétence. Administrativement, les enseignants doivent assurer un nombre minimal d'heures qui est défini dans leur statut. Lorsqu'un un enseignant est chargé d'un enseignement donné, il est tenu d'en respecter le volume horaire prévu par le responsable pédagogique. En cas d'absence, l'enseignant doit prévoir des séances de rattrapage. Il doit donc connaître précisément la disponibilité des ressources de sa séance. Cette organisation garantit que tous les étudiants qui suivent une même formation auront eu le même volume horaire d'enseignement.

En résumé, les données du problème à résoudre sont constituées par :

1- un ensemble de créneaux horaires étalés sur une semaine de six jours, du samedi au jeudi avec un nombre six périodes (du samedi au mercredi) et un nombre de trois périodes le jeudi. La durée d'une période est d'une heure trente minutes.
2- Un ensemble de promotions ou groupes d'étudiants.
3- Un ensemble de cours, TD ou TP à programmer dans la semaine.
4- Un ensemble de locaux (salles, amphis et labos).

L'affectation des modules, enseignants, locaux à des périodes est soumise à des contraintes qui diffèrent selon leurs priorités (l'intérêt que recouvre la satisfaction d'une contrainte ou sa violation).

Une contrainte ne revêt pas nécessairement un aspect absolu (soit elle est vérifiée ou violée) mais peut être formulée sous forme d'un objectif qui doit être approché autant que possible, selon ce critère, les contraintes peuvent être réparties en deux grandes classes : les contraintes dures (absolues) et les contraintes de préférences.

IV-3-1 Les contraintes dures :

Ce type de contraintes doit être obligatoirement satisfait dans toutes les situations car la violation de l'une de ces contraintes rend l'emploi du temps inefficace dans la réalité. On distingue dans notre cas cinq contraintes dures :

1- un enseignant ne peut pas être affecté à deux séances différentes à la même période.
2- Une salle ne peut pas accueillir deux séances différentes à la même période.
3- Chaque enseignant doit enseigner un module qui entre dans ses compétences.
4- Un module doit respecter le nombre de séances hebdomadaires de ce dernier c'est à dire si un module est enseigné trois fois par semaine, alors il doit apparaître 3 fois dans le chromosome de la promotion.
5- Un emploi du temps doit comporter tous les modules d'une promotion.
6- La charge journalière d'un enseignant ne doit pas être dépassée.

IV-3-2 Les contraintes de préférence :

Contrairement au type de contraintes précédent, les contraintes de préférences n'exigent pas la vérification stricte, mais d'approcher au maximum de l'objectif voulu. Dans notre cas, on distingue :

1- essayer d'éviter aux (enseignant ou étudiants) des pertes de temps par de trop longs espacements entre deux séances d'une même journée (pas de trous).

2- Eviter que certains jours se trouvent surcharger alors que d'autres le sont moins.
3- Eviter d'affecter une période jugée non convenable à un enseignant, sauf si cela est inévitable
4- Les modules de coefficient minimal ne doivent pas occuper les séances de la matinée d'un journée donnée, au détriment des modules de coefficients élevés.
5- Minimiser les déplacements des étudiants dans l'établissement.
6- Libérer quelques après-midi pour les enseignants

Afin de modéliser l'ensemble des contraintes régissant notre problème, on doit d'abord définir les structures de données nécessaires.

## IV-4 Structures de données :

Dans notre cas, nous proposons les structures de données suivantes :

- Un tableau d'activités pédagogiques ACT comportant les différentes activités pédagogiques à programmer pour une promotion donnée. Le contenu de chaque élément est un enregistrement représentant l'activité pédagogique numéro i et rassemblant l'enseignant i1 et l'évènement i2 (un événement est composé d'un module en terme de cours, TD ou TP et le groupe (ou section) d'étudiants concerné par cet enseignement).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Ens1 even1 | Ens1 even1 | Ens1 enev2 | Ens2 even2 | .... | Ensi evenj | .... | | | |

- Un tableau événement EVEN comportant les différents enseignements à programmer pour une promotion donnée. Le contenu de chaque élément est un enregistrement représentant l'évènement i et rassemblant l'enseignement et les différents groupes d'étudiants concerné par cet enseignement.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| M1g1 | M1g1 | M2g1 | M2g3 | …. | migj | | ….. | | |

- Un vecteur binaire ETATE associé à chaque enseignant indiquant l'état de l'enseignant à la période i, chaque élément du vecteur est une valeur binaire : 0 signifie que l'enseignant est libre, et 1 signifie que l'enseignant est occupé.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | | | … | 1 |

- Un vecteur binaire ETATL associé à chaque local indiquant l'état du local à la période i, chaque élément du vecteur est une valeur binaire : 0 signifie que le local est libre est libre, et 1 signifie que le local est occupé.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | | | … | 0 |

- Une matrice emploi du temps EPDT représentant toutes les séances de la semaine (en colonnes), qui sont au nombre de 33 périodes allant du samedi au jeudi et en ligne les différents locaux. EPDT(i,j)=k si l'activité pédagogique k est programmée à la période j dans la salle i.

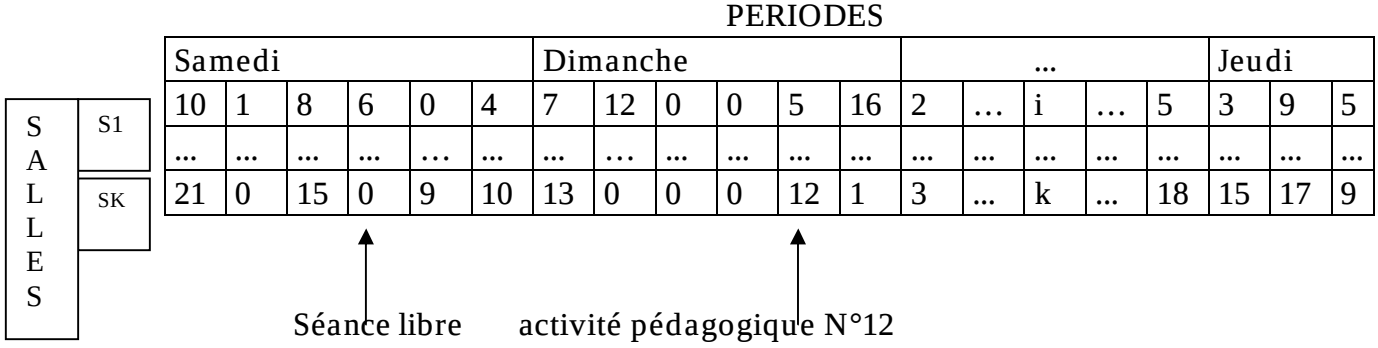
Chaque élément de l'EPDT est un enregistrement comportant, l'enseignant, la salle et l'évènement impliqué dans la période correspondante. Ainsi pour concevoir l'emploi du temps, il faut remplir tous ses enregistrement tout en respectant l'ensemble des contraintes.

## IV-5 Présentation de l'algorithme génétique :

### a- Représentation chromosomique :

Le codage dépend du problème à résoudre. En effet, sa définition permet de cerner l'espace des solutions possibles. Ce codage doit, de plus, être aussi compact que possible pour permettre une évolution rapide.

Principalement, le codage binaire est le plus commun, car les premiers travaux, appliquant les algorithmes génétiques, ont utilisé ce type de codage. Pour notre problème, l'utilisation du codage binaire est d'une difficulté qui complique le décodage des solutions. Dans cette optique, on choisit un codage entier. Chaque gène correspond au numéro de l'activité pédagogique programmée à une période donnée dans une salle donnée. Voir figure 4.1




| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 1 | 8 | 6 | 0 | 4 | 7 | 12 | 0 | 0 | 5 | 16 | 2 | … | i | … | 5 | 3 | 9 | 5 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 21 | 0 | 15 | 0 | 9 | 10 | 13 | 0 | 0 | 0 | 12 | 1 | 3 | … | k | … | 18 | 15 | 17 | 9 |

Fig 4.1 : représentation chromosomique des solutions

Dans cette représentation, la taille du chromosome, sera calculée comme suit :
N = nombre de jours d'enseignement par semaine
X nombre de périodes d'enseignement par jour
Dans notre cas :
Le nombre de jours d'enseignement par semaine = 6 (du samedi au jeudi)
Le nombre de périodes d'enseignement par jour = 5 (du samedi au mercredi) et
= 3 le jeudi
Donc Le nombre de colonnes est de = (5*6) + 3 = 33.
Le nombre de lignes est égal au nombre de salles qui est fixé au départ par exemple egal à nbsal.

Les chromosomes des individus qui vont constituer la population à chaque génération sont des matrices edpt(n,m) avec n= nbsal et m=33.

b- Modélisation des contraintes :

Les contraintes auxquelles nous nous intéressons sont donc subdivisées en deux grandes catégories :

- des contraintes dures ou de faisabilité de l'emploi du temps, qui doivent être respectées dans chaque instanciation. Donc la violation d'une seule contrainte dure, viole la faisabilité de l'emploi du temps.
- Des contraintes de préférences qui seront modéliser comme des objectifs. Leurs utilité réside dans la sélection de l'emploi du temps le plus performant. Donc, le chromosome gagnant est celui qui approche le plus ces objectifs.

b-1 les contraintes dures :

A chaque contrainte i est associée une mesure de sa violation V(i).

1. Un enseignant ne peut pas être impliqué dans plus d'une séance à la fois. Nous devons donc vérifier qu'il est libre pendant la séance d'affectation. D'où la mesure de violation :

$$V(1) = \sum_{i=1}^{nbsal} \sum_{j=1}^{33} DisponibleE(X(i,j), I) \qquad (4.1)$$

Avec X(i,j) retourne l'enseignant affecté à la séance qui se déroule à la salle i pendant la période j.

$$\text{Et DisponibleE}(X(i,j), I) = \begin{cases} 0 & \text{si l'enseignant en question est libre pour l'instanciation I} \\ 1 & \text{sinon} \end{cases}$$

2. Une salle ne peut pas être impliqué dans plus d'une séance à la fois. Nous devons donc vérifier qu'elle est libre pendant la séance d'affectation. D'où la mesure de violation :

$$V(2) = \sum_{i=1}^{nbsal} \sum_{j=1}^{33} DisponibleS(Y(i,j), I) \qquad (4.2)$$

Avec Y(i,j) retourne la salle affectée à la période j.

Et DisponibleS(Y(i , j), I) = { 0 si la salle en question est libre pour l'instanciation I ; 1 sinon

3. chaque enseignant doit enseigner les modules qui entrent dans ses compétences. D'où :

$$V(3) = \sum_{i=1}^{nbsal} \sum_{j=1}^{33} Apte(ENS(i,j), MOD(i,j), I) \qquad (4.3)$$

Avec ENS(i,j) retourne l'enseignant affecté à la séance qui se déroule à la salle i pendant la période j.
MOD(i,j) retourne le module enseigné à la séance qui se déroule dans la salle i pendant la période j.

Et Apte(X, Y, I) = { 0 si le module Y entre dans les compétences de l'enseignant X pour l'instanciation I. ; 1 sinon

4. chaque module doit respecter le nombre de séances hebdomadaires prévus. D'où :

$$V(4) = \sum_{i=1}^{nbsal} \sum_{j=1}^{33} Hebd_k(i,j) = nbsh_k \qquad (4.4)$$

Avec $nbsh_k$ = nombre de séances hebdomadaires du module k.

5. L'emploi du temps doit comporter tous les modules d'une promotion donnée. D'où :

$$V(5) = \sum_{i=1}^{nbsal} \sum_{j=1}^{33} Occur_k(i,j) > 0 \qquad (4.5)$$

Avec $Occur_k(i,j)$ retourne le nombre d'occurrence du module k.

6. la charge journalière d'un enseignant ne doit pas être dépassée. D'où :

$$V(6) = \sum_{i=1}^{nbsal} \sum_{j \in m} Charge_k(i,j) \leq charg\_max_k \qquad (4.6)$$

Avec $charg\_max_k$ est le nombre maximal de séances par jour pour l'enseignant k.

$Charge_k(i,j) =$ { 0 si l'enseignant k est affecté à la séance qui se déroule dans la salle i à la période j. ; 1 sinon }

et m = { [1..6] pour le samedi ; [7..12] pour le dimanche ; ….. ; [31..33] pour le jeudi }

b-2 les fonctions objectifs :

A chaque emploi du temps possible S est associé un vecteur objectif F(S) qui représente l'évaluation de la solution. Avec F = ($f_1$,$f_2$,$f_3$,$f_4$,$f_5$,$f_6$), chaque $f_i$ correspond à la contrainte de préférence i

b- Génération de la population initiale :

La génération de la population initiale, consiste en la création aléatoire de cette population de N chromosomes en d'autres termes, N matrices d'emploi du temps seront générés. L'aspect combinatoire nous garantit normalement une bonne répartition des individus dans l'espace de recherche.

c- Opérateurs génétiques :

L'application des algorithmes génétiques à un problème donné nécessite aussi le choix des opérateurs génétiques qui serviront à faire évoluer la recherche. Pour une application standard de la mutation, on fait intervertir un ou plusieurs bits au niveau des chromosomes des enfants. Or le code utilisé dans notre cas n'est pas binaire ( c'est un entier), l'inversion d'un allèle du chromosome n'est pas bien évidente, une suggestion consiste à remplacer le nombre en question par un nombre aléatoire pris entre 0 et le nombre des séances à condition qu'il soit différent de ce nombre choisi pour la mutation.

L'opérateur de croisement nommé aussi croisement à un point consiste au choix aléatoire d'un point de croisement et de permuter tous les gènes d'un côté de ce point entre deux individus.

## IV-6 Opérateur de sélection :

L'absence d'une relation d'ordre totale entre les différentes solutions possibles d'un problème multiobjectif fait que la notion d'optimalité tel que conçu pour le cas mono-objectif n'est plus utilisable. La notion d'optimalité Pareto tirée de la notion de dominance mathématique des vecteurs, jouit d'une grande adhésion de la part des chercheurs.

### IV-6-1 Dominance et optimalité Pareto :

La notion de dominance représente une relation d'ordre sur l'ensemble des points de l'espace de recherche.

$S_i$ domine $S_j$ si et seulement si :

$\forall k \in [1..n]\ f_k(S_i) \leq f_k(S_j)$ et $\exists\ k \in [1..n] / f_k(S_i) < f_k(S_j)$

$S_i$ est Pareto optimale si et seulement si :

$\forall\ S_j / S_j \neq S_i$ on a $S_j$ ne domine pas $S_i$

où n désigne le nombre d'objectifs à optimiser.

De façon intuitive, une solution $S_i$ est dite « dominant » la solution $S_j$ si et seulement si $S_i$ est au moins meilleure que $S_j$ sur tous les plans (par rapport à l'ensemble des objectifs d'optimisation) et qu'elle soit préférable à $S_j$ au moins pour un objectif.

Le but de l'optimisation multiobjectif est d'aboutir à un ensemble de solutions non dominées qui approxime au mieux l'ensemble des solutions Pareto du problème.

### IV-6-2 Stratégie de sélection :

Dans les stratégies de sélection multiobjectifs, les différences principales résident dans la façon dont les individus de la population sont ordonnés, ainsi que l'expression permettant le calcul des probabilités de sélection. La sélection élitiste dans notre cas, consiste à maintenir une sorte de population archive qui contiendra les meilleurs solutions non dominées rencontrées tout au long de la recherche. Cette population participera aux étapes de sélection et de reproduction.

## IV-7 Maintien de la diversité :

Les algorithmes génétiques classiques sont réputés pour être très sensibles quant au choix de la population initiale ainsi qu'aux mauvais échantillonnages lors de la sélection. Cette fragilité est observable sur le plan de perte de diversité ou ce qu'on appelle aussi la dérive génétique. Pour palier à cet inconvénient, plusieurs approches visant à maintenir la diversité dans la population ont été proposées dans la littérature. Parmi les plus utilisées est le sharing qui consiste à modifier la valeur du coût d'un individu. C'est cette nouvelle valeur qui sera utilisée comme valeur d'adaptation par l'opérateur de sélection. Dans ce cas l'objectif est la dégradation de l'adéquation des individus qui appartiennent à des espaces de recherche de forte concentration de solutions ce qui aura pour effet de favoriser la dispersion des solutions de la population dans l'espace de recherche. L'étape du sharing suit directement l'étape d'évaluation et précède celle de la sélection.

## IV-8 Hybridation avec la recherche locale :

Les chercheurs ont prouvé que l'utilisation de la recherche locale (comme hybridation avec les algorithmes génétiques) est un moyen d'accélération et de raffinement de la recherche génétique. Pour cela, nous suggérons l'utilisation de la recherche locale au niveau de l'opérateur de mutation.

## IV-9 conclusion :

L'algorithme génétique proposé consiste en une introduction progressive de concepts multiobjectifs tels que la sélection, la maintenance de la diversité. Seuls les tests pratiques peuvent prouver l'efficacité d'une approche donnée.

# Conclusion générale

L'objectif de ce travail a été l'étude des approches de résolution des problèmes d'optimisation combinatoire et plus particulièrement le problème de l'emploi du temps afin de proposer une approche qui peut contribuer à la résolution de ce problème.

L'accent a particulièrement été mis sur l'utilisation de l'optimisation multiobjectif. Un choix justifié par la nature même du problème qui en réalité formuler sous forme d'un ensemble d'objectifs qui peuvent éventuellement être contradictoires.

L'optimisation multiobjectif revêt deux aspects. Le premier concerne la manière dont le décideur coopère avec l'algorithme de résolution. Trois approches principales sont alors possibles :

- L'utilisateur formule ces préférences à priori. La recherche est alors lancé, le résultat est une solution unique
- La recherche est lancée en premier, conduisant à la production de plusieurs solutions. L'utilisateur choisit une solution parmi celles obtenues.
- Une coopération est installée entre le décideur et l'algorithme de recherche. L'algorithme est lancé sur la base des préférences du décideur. Les connaissances acquises lors de la résolution sont alors utilisées par le décideur afin de reformuler ses préférences.

Le deuxième aspect se rapporte à la signification donné à la notion d'optimalité. Selon cette distinction, les approches multiobjectifs peuvent être classées en trois catégories : les approches opérant par transformation du problème en un problème mono-objectif, les approches non Pareto et les approches Pareto. Parmi celles-ci, les approches Pareto semblent être les plus utilisées dans le domaine de l'optimisation.

L'objectif de cette optimisation multiobjectif est de produire un ensemble de solutions efficaces, généralement Pareto optimales. Pour cette raison, l'algorithme proposé stocke les meilleurs solutions trouvées durant la recherche. Dans les algorithmes génétiques ceci est réalisé en maintenant une population supplémentaire dite population Pareto ou archive. L'élitisme consiste à faire participer la population Pareto lors de la phase de sélection. Ce mécanisme sert généralement comme moyen d'intensification de la recherche.

Un volume important de travail reste à faire. Premièrement l'implémentation de ce travail. Ce qui permettra d'analyser les performances de la méthode. Les AGs sont souvent critiqués pour leur lenteur, pour cela les mécanismes d'hybridation sont à étudier. Les AGs sont aussi connus pour leur sensibilité quant au choix de la population initiale. L'utilisation d'heuristiques pour la génération de cette population pour améliorer les performances de l'AG peuvent être étudiées afin d'assurer de bonnes solutions de départ.

# Abstract

The problems of scheduling notably timetabling problem received a big attention. The timetabling problem which is subjected to a variety of constraints sometimes contradictory, it is a problem NP-complete. Therefore some heuristic methods must be used to solve it. Most works in this sense treats it in its shape mono objective. In this study, we were interested in the survey of this problem in its version multi objective while using the meta-heuristics. A survey retailed of the combinatorial optimization problems mono objective and multi objective is led. All tentative of mathematical modeling as the representation by the graphs (coloration of summits or of stop), the linear programming, the theory of the wholes, etc. failed and the examples showed that these exact methods are not sufficient anymore in most real cases. The reason is that the topology of such problems is of very elevated complexity. The meta-heuristic, on the other hand, are endowed with general mechanisms their permitting a good investigating of the research space. So the techniques of research approached are discussed, their fundamental principles are described. The variants multi objectives of these methods are also presented. A particular interest is carried to the genetic algorithms seen their big efficiency facing the problems of combinative optimization. The extension of this method to the case multi objective is also exposed. An evolutionary algorithm multi objective is proposed for the resolution of the timetabling problem ..

**Keywords:** timetabling problem, planning, meta-heuristic, evolutionary algorithms, combinatorial optimization, multiobjective optimization.

# Résumé

Les problèmes de planification d'horaires de travail notamment le problème de l'emploi du temps ont reçu une grande attention. La plupart des travaux dans ce sens les traitent dans leur forme mono-objectif. Dans ce mémoire, nous nous sommes intéressés à l'étude de ce problème dans sa version multi objectif en utilisant les méta heuristiques.

Une étude détaillée des problèmes d'optimisation combinatoire mono-objectif et multi objectif est menée.

Toutes les tentatives de modélisations mathématiques telles que la représentation par graphes (coloration de sommets ou d'arrêtes), programmation linéaire, théorie des ensembles … ont échouées et les exemples ont montré que ces méthodes exactes ne suffisent plus dans la plupart des cas réels. La raison est que la topologie de tels problèmes est de complexité très élevée. Les méta heuristiques, par contre, sont dotées de mécanismes généraux leurs permettant une bonne investigation de l'espace de recherche.

Ainsi les techniques de recherches approchées sont discutées, leurs principes fondamentaux sont décrits. Les variantes multi objectifs de ces méthodes sont aussi présentées.

Un intérêt particulier est porté aux algorithmes génétiques vu leur grande efficacité face aux problèmes d'optimisation combinatoire. L'extension de cette méthode au cas multi objectif est aussi exposée.

Un algorithme évolutionnaire multi objectif est proposé pour la résolution du problème de l'emploi du temps.

# Références bibliographiques


[ABR91] : D. Abramson, « Constructing school timetabling using simulated annealing : sequential and parallel algorithms », Management science 37, pp. 98-113, 1991.

[ABR92] : D. Abramson, J. Abela, « A parallel etic algorithm for solving the school time tabling problem », 15 A computer science conference, Fevrier 1992.

[BABOU] : M. Babes et H. Ounas, « Elaboration d'u n algorithme génétique pour résoudre le problème d'emploi du temps d'université », Université Badji Mokhtar Annaba, Algerie.

[BAR03] : V. Barichard and J. Hao, « Genetic Tabu Search for Multiobjective knapsach problem », Angres 2003.

[BAT94] : Battiti R., and Tecchiolli G., « The reactive tabu search », in ORSA Journal on Computing, 6(2), pp. 126-140, 1994.

[BEN97] : F. Ben Abdelaziz and S. Krichen, « A tabu search heuristic for multiple objective knapsack problems », Ructor Reasearch Report, RR 28-67, 1997.

[BLU03] : C. Blum and Roli, « Metaheuristics in combinatorial optimization : overview and conceptual comparison », ACM Computing survey, vol 35, N°3, september 2003.

[BRAEF] : N. Brauner, R. Echahed, G. Finke, F. Prost, W. Sewe, « Intégration des méthodes de réécriture et de recherche opérationnelle pour la modélisation et la résolution de contraintes : application à la planification de personnel médical », Laboratoire Leibniz-IMAG, Grenoble.

[BUR02] : Edmund Kieran Burke et Sanja Petrovic, " Recent Research Directions in Automated Timetabling", ASAP Resarch Group, school of computer science and information technology,

university of Nottingham, EJOR, 2002.

[BUR94a] : E. Burke, D. El and R. Weare, « A genetic algorithm for university timetabling »,Proceeding of the "AISB" Workshop on evolutionary computing , University of leeds, UK, 11th-13th April 1994.

[BUR94b] : E. Burke, D. El and R. Weare, « A genetic algorithm based university timetabling system», Proceeding of the 2nd East-West international conference on computer technologies in education ( Crimea, Ukraine, 19th –23rd sept 1994), vol 1, pp 35-40, 1994.

[BUR95] : E. K. Burke, J. P. Newell and R. F. Weare, « A memetic algorithm for university exam timetabling », proceedings of the first international conference on the practice and theory of automated timetabling (ICPTAT-95), 496-503, 1995.

[BUR97] : Burke E. ,kingston j., Jackson k., Weare R., "Automated university Timetabling : the state of the art", the Computer Journal 40 (9) 565-571, 1997.

[BUR98] : E.K Burke, J.P. Newall, R.F. Weare, "A Simple Heuristically Guided Search for the Timetable Problem", Proceeding of the International ICSC Symposium on Engineering og Intelligent Systems, pp. 574-579, 1998.

[CAN02] : G. Cangini, « une méthode hybride pour la confection d'horaires de médecins pour l'hôpital Côte-des Neiges », Mémoire de maîtrise, CRT, GERAD & Ecole Polytechnique de Montréal, Département de Génie Electrique et de Génie Informatique,2002.

[CAO91] : V. Cao, O. Du Merle, J.P. Vial , « Un système de confection automatisé des horaires d'examens à la faculté des SES de l'université de Genève », cahier de recherches N°91.4, Dép. d'économie commerciale, Université de Genève,1991.

[CAR90] : R. Carraway, T. L. Morin, H. Moskowitz, «Generalized dynamic programming for multicriteria optimization », european journal of operational research, vol 44, page 95, 1990.

[CHA02] : Chan Yew Chéong, Peter, « La planification du personnel : acteurs, actions et termes multiples pour une planification opérationnelle des personnes », Thèse de doctorat, Institut IMAG, Université Joseph Fourier-Grenoble, 1 octobre 2002.

[COE02] : C. Coello, A. Carlos and N. Cortes, « An approach to solve multiobjective Optimization problems based on an artificial immune system », in Jonathan Timmis and Peter J. Betley, editors, first international Conference on artificial immune systems (ICARIS'2002), pages 212-221. University of Kent at Canterbury, UK. ISBN 1-902671-32-5, September 2002.

[COE98] : C. Coello Coello, « An updated survey on G.A. based multiobjective optimization techniques », Rapport technique Lania-RD-98-08, Xalapa Veracruz, Mexico, décembre 1998.http://www.lania.mx/~ccoello/EMOO

[COL02] : Y. Collette et P. Siarry, « Optimisation multiobjectif ». Eyrolles, 2002.

[COL88] : Collins .E., r.w. Eglese and B. L. Golden, « Simulated annealing : an annotated bibliography », American journal of mathematical and management sciences, 8, p. 209-307, 1988.

[COR00] : D. W. Corne, J. D. Knowles and M. J. Oates, « The Pareto-Envelope based Selection Algorithm for Multiobjective Optimization », in Proccedings of the sixth International Conference on Parallel Problem Solving from nature (PPSN VI°, pages 839-848, Berlin, September 2000.

[COS94] : D. Costa, « A tabu search algorithm for computing an operational timetable», european journal of operational research, vol 76, page 98,1994.

[CUI01] : Cui, Xunxue, M. Li and T. Fang, « Study of population diversity of multiobjective evolutionary algorithm based on immune and entropy principes », in proceedings of the congress o evolutionary computation 2001, volume 2, pages 1316-1321,

Piscataway, New Jersey, IEEE Service Center, May 2001.

[CZY98] : P. Czyzak and A. Jaszkiewicz, « A Pareto simulated annealing- a metaheuristic for multiple objective combinatorial optimization ». Journal of multi-criteria decision ana lysis, 7(1):34-47, 1998.

[DAUMA] : Daumas Authman et associés, « j'Road Planner ».

[DEB00] : K. Deb, S. Agrawal, A. Pratapand and T. Meyarivan, « A Fast and Elitist multiobjective genetic algorithm : NSGA-II, Technical Report, Indian Institute of Technology, (Kan GAL),2000.

[DEL95] : D.Delahaye, « optimisation de la sectorisation de l'espace aérien par algorithme génétique »,1995.

[DOR96] : Dorigo M., Maniezzo V., and Colorni A., « Ant system : optimization by a colony of cooperating agents », IEEE Transactions on systems, Man and Cybernetics- part B, 26(1), pp. 29-41, 1996.

[DUE90] : Dueck G. and T. Scheuer, « Threshold accepting : a general purpose optimization algorithm appearing superior to simulated annealing », Journal of computational physics, 90, p. 161-175, 1990.

[FEA05] : Paul Feautrier, « Recherche opérationnelle », 16 novembre 2005.

[FEO95] : T.A. Feo et M. G. C. Resende, « Greedy randomized adaptive search procedures », Journal of global optimization, 6: 109-133, 1995.

[FON93] : Carlos M. Fonseca and Peter J. Fleming, « Genetic algorithm for multiobjective optimization: formulation, discussion and generalization », in proceeding of the fifth internetional conference on genetic algorithms, San Mateo, California, p. 416-423, 1993.

[FRA04] : Francillette Remy, «Optimisation combinatoire », Université des Antilles et Guyane, Faculté des sciences exactes et naturelles, Rapport de T.E.R. , 2004.

[GAN97] : X. Gandibleux, N. Mezdaoui, A. Freville, « A multiobjective tabu search procedure to solve combinatorial optimization problems » in : R. Ca. Ballero, F. Ruiz, R. Steuer (eds), Advances in Multiple Objective and Goal Programming. Lecture Notes in Econom, 1997.

[GEN01] : B. Gendron, « Adaptation d'un modèle mathématique de génération de colonnes pour les horaires de médecins », MIC2001- 4th Metaheuristics Internationa conference,2001.

[GLO89] : F. Glover, « Tabu search- part I », ORSA Journal on Computing, 1: 190-206, 1989.

[GLO90] : F. Glover, « Tabu search- part II », ORSA Journal on Computing, 2: 4-32, 1990.

[GLO94] : F. Glover, « Genetic algorithms and scatter search », statistics and computing, 4 : 131-140,1994.

[GOL87] : D.E. Goldberg and J. Richardson, « Genetic algorithms with sharing for multimodal function optimization ». In Genetic Algorithms and their applications: Proceedings of the second International Conference on Genetic Algorithms, pages 41-49. Lawrence Erlbaum, 1987.

[GOL89] : D.E. Goldberg, «Genetic algorithms for search, optimization, and machine learning ». Reading, MA: Addison-Wesley, 1989.

[HAN97] : M. P. Hansen, « Tabu search for multiobjective optimization », in proceedings of 13 th International conference on MCDM, 1997.

[HAN98] : M.P. Hansen, « Metaheuristics for multiple objective combinatorial optimization », Ph.D. thesis, IMM-PHS-1998-45, Tchnical University of Danmark, Lyngby, 1998.

[HER91] : A Hertz, « Tabu search for large scale timetabling problems », european journal of operational research, vol 54, page 39, 1991.

[HER94] : A. Hertz, B. Jaumard, C. C. Ribero, W. P. Filho, « A multi-creteria tabu search approch to cell formation problems in group technology with multiple objectives », Recherche opérationnelle, vol 28, N°3, page 303, 1994.

[HOL75] : J.H. Holland, “Adaptation in natural and artificial systems”, University of Michigan Press, 1975

[IRE01] : S. Iredi, D. Merkle, M. Middendorf, « Bi-criterion optimization with multi colony algorithms », in proceeding of the first international conference on evolutionary multi-criterion optimization. Lecture notes in computer science, Springer, Berlin, 2001.

[ISK91] : MW Isken, WM. Hancock, “ A heuristic approach to nurse scheduling in hospital units with non-stationary, urgent demand, and a fixed straff size “, journal of the Society for Health systems, 1991, 2 : 24-41.

[JAU00] : B. Jaumard, P. Galinier, « Méthode tabou pour l’organisation des soins à domicile », CRT, GERAD & Ecole Polytechnique de Montréal, Département de Génie Electrique et de Génie Informatique,2000.

[KEN95] Kennedy J. and Eberhart R. C., « Particle swarm optimization », proceeding of the IEEE conference on neural networks, IV, Piscataway, NJ, pp. 1942-1948, 1995.

[KIR83] : S. Kirkpatrick, C.D. Gellat et M.P. Vecchi, « Optimization by simulated anealing », Science, 220: 671-680, 1983

[KNO99] : J. Knowles and D. Corne, « The Pareto archived evolution strategy : a new baseline algorithm for multiobjective optimization », in 1999 Congress on Evolutionary Computation, Piscataway, NJ, IEEE Service Center,98-105, 1999.

[LIS96] : J. Lis, A. E. Eiben, « A multi-sexual genetic algorithm for multi-objective optimization », international Conference on genetic algorithms ICGA, Pages 59, Nagova, Japon, 1996.

[LOU02] : Lourenço H. R., Martin O. and Stutzle T. , « Iterated local search », in F. Glover and G. Kochenberger, editors, Handbook of metaheuristics. Kluwer Academic Publishers, Norwell, MA, pp. 321-353, 2002.

[METAH] : http://www.metaheuristics.org

[MIN92] : S. Minton, M. Johnston, A. Philips, P. Laird, « Minimizing conflicts : a heuristic repair method for constraint satisfaction and scheduling problems », Artificialk intelligence, volume 58, pages 160-205, 1992.

[MLA97] : N. Mladenovic et P. Hansen, « Variableneighbourhood decomposition search », Computers and operations research, 24 : 1097-1100,1997.

[MOS99] : P. Moscato, « Memetic algorithms : A short introduction », in D. Corne M. Dorigo et F. Glover, editors, new ideas in optimization, pages 219-234, McGraw-Hill, New York, 1999.

[MSKS] : M. Sevaux[1] and K. Ssorensen[2], « MA/PM: une nouvelle métaheuristique hybride », [1]université de valenciennes-CNRS, UMR 8530, LAMIH/SP, [2]université d'Anvers- Faculty of Applied Economics.

[OST82] : R. Ostermann, D. de Werra, « Somme experiments with a timetabling system », OR Sepektum 3, 1982.

[PAR98] : Marc Parizeau, « Introduction à la NP-complétude », Département de génie électrique et génie informatique, Université Laval, 1998.

[PES00] : G. Pesant, P. Galinier, « un modèle de programmation par contraintes pour la confection d'horaires d'infirmières », European Journal of Operational Research,234 :156-167,2000.

[REM03] : Remy-Robert, Alexandre Joseph, « Systèmes interactifs d'aide à lélaboration de plannings de travail de personnel », Thèse de

doctorat, Laboratoire TIMC, Institut IMAG , Université Joseph Fourier-Grenoble, 07 novembre 2003.

[ROS89] : F. Rossi, C. Pettrie, V. Dahr, « On the equivalence of constraint satisfaction problem », Act –ai-222-89, MCC corporation, Austin, Texas, 1989.

[ROS95] : P. Ross, D. Corne, « Improving evolutionary time tabling with delta evaluation and directed mutation », Departement of artificial intelligence, University of Edinburhg, 1995.

[SAN01] : Sandhu k., "Automating class schedule generation in the context of university timetabling information systém" , School of Management, Nathan Campus, Griffith University, 21 september 2001.

[SCH85] : David Schaffer, «Multiple objective optimization with vector evaluated genetic algorithm», in genetic algorithm and their applications: Proceedings of the first international conference on genetic algorithm, p. 93-1000, 1985.

[SCH95b] : J. R. Schott, « Fault tolerant design using single and multicriteria genetic algorithm optimization », master's thesis, Departement of Aeronautics and Astronautics, Massachutts Institute of Technology, 1995.

[SCH95a] : Schaerf A., et Schaerf M. "Local search techniques for large high school timetabling" , in proceeding of the 1st international conferenceon the practice and theory of automated timetabling,pp. 313-323, 1995.

[SCH96] : Schaerf A., "Tabu Search technique for large high school timetabling problems", In Proceeding of the 13th national conference on artificial intelligence,USA,1996.

[SER92] : Serafini, « Simulated annealing for multiple objective optimization problems », in: Proceeding of the Tenth International Conference on Multiple Criteria Decision Making, Taipei 19-24.07, vol. 1, 87-96, 1992.

[SOC02] : Krzysztof Socha, Joshua Knowles and Michael Sampels, “A MAX-MIN Ant System for the University Course Timetabling Problem”, IRIDIA, Université libre de Bruxelles, 2002.

[SRI93] : N. Srinivas, K. Deb, « Multiobjective optimization using non-dominated sorting in genetic algorithms », Technical Report, Department of technical engineering, Indian Institute of Technology, Kanput, India, 1993.

[STE91] : B. S. Stewart, C. C. White, «Multiobjective A*», journal of association for computing machinery, vol 38, N°4, page 775, 1991.

[SUR95] : P. Surry and al, « A multiobjective approach to constrained optimisation of gas supply networks : the COMOGA method », evolutionary computing AISB workshop selected papers, lecture notes in computer science, p. 166-180, 1995.

[TAI91] : Taillard E. D., « Robust tabu search for the quadratic assignment problem », in Parallel Computing, 17, pp. 443-455, 1991.

[TAL01] : E. Talbi, « Métaheuristiques pour l’optimisation combinatoire multi-objectif : Etat de l’art », Lille : LIFL, 2001

[TAN95] : H. Tanaki and al, « Multicriteria optimization by genetic algorithms : a case of scheduling in hot rolling process », in proceeding of the third apors, p. 374-381, 1995.

[TRI80] : Tripathy A., “ A Lagrangian Relaxation Approach to Course Scheduling”, Journal of the Operational Research Society 31, 1980.

[TUY98] : D. Tuyttens, « An improved MOSA method for solving multi-objective combinatorial optimization problems », rapport interne, laboratory of mathematics operational research, faculté plytechnique de Mons, Belgique, 1998.

[ULU98] : E. Ulungu , J. Teghem, C. Ost, « Efficiency of interactive Multiobjective simulated annealing through a case study »,

Journal of the operational research society, Vol 49, numéro 10, pages 1044, 1998.

[VAN00] : D. Van Veldhuizen, Jesse B. Zydallis and Gary B. Lamont, « Messy Genetic Algorithm Based Multi-Objective Optimization : A Comparative Statistical Analysis », in PPSN/SAB Workshop on Multiobjective Problem Solving from Nature (MPSN), Paris, France, September 2000.

[VOU95] : C. Voudouris et E. Tsang, « Guided local search », technical report TR CSM-247, University of Essex, UK, 1995.

[WEI94] : Georges Weil, Kamel Heus, Patrice François, « Gymnaste : Aide à l'élaboration des roulements infirmiers . Du traitement des absences au management participatif », Laboratoire TIMC, SILM , CHU de Grenoble , Université Joseph Fourier-Grenoble,1994.

[WER85] : De Werra D., "An Introduction to Timetabling", European Journal of Operational research 19, 151-162.1985

[WHI01] : George M. White and Bill S. Xie, "Examination Timetables and Tabu search with longer-term memory", E. BURKE and W.Erben (Eds): PATAT 2000, LNCS 2079, pp. 85-103, 2001. Springer-Verlag Berlin Heidelberg, 2001.

[WKIP] : http://fr.wikipedia.org/wiki/MÃ©taheuristique

[YAN99] : Yann le Fablec, « prévision de trajectoires d'avions par réseaux de neurones », Thèse, Laboratoire d'optimisation globale CENA/ENAC, Toulouse, 1999.

[ZED04] : A. Zedairia, M. Babes, « Une heuristique hybride pour résoudre un problème de planification dans un milieu hospitalier », CISC'04 the International Conference on Complex Systems, September 6-8, 2004, Jijel, Algeria.

[ZIT01] : E. Zitzler, M. Laumanns and L. Thiele, « SPEA2 : Improving the performance of the Strength Pareto evolutionary Algorithm »,

Technical Report 103, Computer engineering and Communication Networrks lab (TIK), Swiss Federal Institute of Technology (ETH) Zurith, Gloriastrasse 35, CH-8092 Zurith, May 2001.

[ZIT98] : E. Zitzler and L. Thiele, « An evolutionary algorithm for multiobjective optimization : the Strenght Pareto Approach », Technical Report, Swiss Federal Institute of technology (Zurich), 1998.

[ZIT99] : E. Zitzler and L. Thiele, « Comparison of multiobjective optimization : evolutionary algorithms : empirical results », Computer engineering and networks laboatory (TIK), Swiss Federal Institute of technology (ETH) Zurich, december 1999.