REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Réseaux

Par: LAOUBI OUSSAMA

SUJET

Application de la programmation par contraintes au problème de gestion du temps cas: département Informatique

Soutenu publiquement le : / / devant le jury composé de :

……	**Université de M'sila**	**Président**
M. KHETTAF Abdelouahab	**Université de M'sila**	**Rapporteur**
……	**Université de M'sila**	**Examinateur**
……	**Université de M'sila**	**Examinateur**

Promotion : 2016 /2017

Dédicaces

A mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien tout au long de mes études,

A mes chères amis KAMEL Mohamed et LOUNNAS Bilal pour leurs encouragements permanents, et leur soutien moral,

A mes chers frères, hamza et islam pour leur appui et leur encouragement,

A toute ma famille pour leur soutien tout au long de mon parcours universitaire,

Que ce travail soit l''accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible,

A tous ceux qui me connaissent.

Oussama

Remerciements

Je commence d'abord par remercier ALLAH de m'avoir donné le courage et la volonté pour réaliser ce travail. Je tiens remercier Monsieur KHETTAF ABDELOUAHAB : mon encadreur que je ne saurais jamais remercier assez pour ses précieux conseils, sa compréhension et sa confiance en moi et que j'ai bien profité de son savoir et son expérience.

Je remercie tous ceux qui ont m'aidé de loin ou de près pour réaliser ce mémoire.

Merci à tout le monde

TABLE DES MATIERES

CHAPITRE 03: MODELISATION PAR CSP

LISTE DES FIGURES

CHAPITRE 02 : PROGRAMMATION PAR CONTRAINTES

LISTE DES TABLEAUX

CHAPITRE 03: MODELISATION PAR CSP

INTRODUCTION GENERAL

Quel que soit son domaine, l'être humain est confronté à différents problèmes dans toutes les sphères de la société. Un problème donné peut être défini par l'ensemble des propriétés qui doivent vérifier ses solutions. Il peut être un problème de décision ou un problème d'optimisation. Un problème de décision peut se ramener à un problème d'existence de solution. Il consiste à répondre à la question « Est-ce qu'il existe une solution basée sur un ensemble d'énoncés et qui satisfait un ensemble de contraintes ? » par une des réponses : « *Oui il existe* » ou bien « *Non il n'existe pas* ». Par contre, un problème d'optimisation peut se ramener à un problème d'existence de solution de bonne qualité. Il consiste à rechercher une solution de qualité suffisante au regard d'un (des) critère(s) donné(s) et des objectifs à satisfaire.

L'optimisation combinatoire occupe une place très importante en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes dans différents secteurs de l'industrie (télécommunications, électronique, mécanique, chimie, transport, …).

Des exemples de problèmes d'optimisation combinatoire peuvent être retrouvés partout dans des secteurs industriels ou de services publics. La résolution de ces problèmes peut souvent se faire à l'aide de méthodes exactes. Cependant, pour une classe de problèmes dits NP-Difficiles, il n'est pas possible de trouver la solution optimale.

Parmi les problèmes les plus difficiles et les plus étudiés, nous avons le problème de gestion des emplois du temps. 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).

La résolution de ce problème de manière optimale s'avère dans la plupart des cas impossible à cause de son caractère fortement combinatoire. Les méthodes exactes requièrent un effort calculatoire qui croît exponentiellement avec la taille du problème.

Durant les dernières décennies, la Programmation Par Contraintes (PPC) est devenue une approche efficace pour modéliser et résoudre les problèmes de planification. La PPC permet

de séparer le modèle décrit grâce à un langage déclaratif (tâches, ressources, contraintes, objectif) des algorithmes employés durant la résolution. Ces travaux visent à faciliter l'interaction entre les modèles et les décideurs par l'intégration d'outils utiles dans un contexte d'aide à la décision. Ainsi, la flexibilité de modélisation, les méthodes structurelles de reformulation, la gestion dynamique des contraintes permettent d'appliquer des méthodes de simulation, de planification, ou même de diagnostic. Les méthodes de résolution telles que la réduction de domaines et la propagation de contraintes combinées à des algorithmes de recherche efficaces, exhaustifs ou non, ont permis la résolution de nombreux problèmes. De nos jours, ces méthodes sont de plus en plus couplées avec des techniques de Recherche Opérationnelle (RO), telles que la programmation linéaire, entière ou mixte, pour élaborer des algorithmes efficaces dédiés à l'optimisation.

L'organisation du mémoire découle naturellement de cette problématique traitée. Il est structuré en trois chapitres qui permettent un cadrage progressif du sujet.

Dans le premier chapitre de notre travail, 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. Le deuxième point cité dans le premier chapitre, consacre à présenter des techniques d'optimisation capables de résoudre le problème des emplois du temps, trois grandes classes de méthodes sont citées : les méthodes séquentielles, les méthodes basées contraintes et les méthodes méta-heuristiques.

Le deuxième chapitre donne une présentation synthétique des concepts liés à la programmation par contraintes, et des techniques majeures de résolution de problèmes de satisfaction de contraintes et d'optimisation sous contraintes.

Le troisième chapitre va présenter une modélisation de problème de gestion des emplois du temps en basant sur les techniques de la programmation par contraintes avec une description détaillée des contraintes liées au département d'informatique.

CHAPITRE 01

LA PLANIFICATION D'HORAIRES DE TRAVAIL

1. Introduction

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.

2. 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 [1] [2].

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

2.2. 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 [1].

2.3. 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, les congés et les repos [1]. 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 des personnels ou les tâches effectuées par les personnels :

- **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 les personnels 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 [1].

2.4. 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 leurs personnels.

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. [1]

2.5. 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 commerçant, 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 alternatives. L'aspect combinatoire (pour l'élaboration des plannings) rend d'autant plus difficile la négociation, car les opinions sont plus difficiles à formuler [1]. Les moyens informatiques apportent une aide certaine notamment dans l'acquisition et la confrontation des données individuelles [1].

2.6. Qui peut se charger de l'élaboration d'un planning ?

Dans la plupart des entreprises, cette tâche peut être centralisée ou déléguée par 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 plannings joue un rôle primordial.

3. Différents types de plannings :

Dans la construction de plannings d'horaires de travail, 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.

3.2. Types de plannings dans le domaine de la santé :

Les plannings dans les 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.

3.3. 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 [1].

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.

3.4. 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» [1].

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 [1]. 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 [1].

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 [1]. En 1985, De Werra, a décrit les divers problèmes traitant le problème d'emploi du temps d'une façon formelle et il a fourni les différentes formulations dans une tentative de les résoudre. Il a aussi décrit les approches considérées les plus importantes à ce temps là [1]. 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 [1].

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 [1]. 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.

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.

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 les littératures.

4. Les méthodes de gestion de l'emploi du temps

4.2. 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. Par exemple si quelques étudiants doivent suivre 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 graphe 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 prévus tôt. Le raisonnement est que les évènements avec un grand nombre de conflits sont plus difficiles à prévoir et donc doivent être abordé 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.

4.3. 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.

4.4. Les méthodes méta-heuristiques

Pendant les deux dernières décennies une variété d'approches méta-heuristiques 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 méta-heuristiques 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 produire 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 méta-heuristique 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 (la 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 méta-heuristique et une méthode locale afin d'affiner la solution. Cette coopération peut prendre la simple forme d'un passage de relais entre la méta-heuristique et la technique locale, comme elles peuvent être entremêlées de manière plus complexe.

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

4.4.1. Les méthodes évolutives :

L'approche proposée est fondée sur une programmation par contraintes utilisant un algorithme génétique comme moteur d'optimisation.

4.4.2. Les méthodes utilisant les colonies de fourmis:

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 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.

4.4.3. Les méthodes utilisant la recherche tabou

Où Schaerf a utilisé cette technique pour résoudre le problème. Il a employé le codage matriciel $M_{i,j}$ 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 à des modèles de coloration de graphes, de relaxation lagrangienne et de réseaux de neurones.

5. 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.

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.

CHAPITRE : 02

PROGRAMMATION PAR CONTRAINTES

1. Introduction

Les Problèmes de Satisfaction de Contraintes (CSP) sont un formalisme générique destiné à modéliser les problèmes typiques de la Recherche Opérationnelle, notamment ceux dons la nature sont combinatoires tels que les problèmes d'ordonnancement, d'allocation, de configuration ... Les CSP sont caractérisés par la difficulté de trouver une solution admissible, c'est-à-dire qui respecte toutes les spécifications qui le structurent, ou par la difficulté d'obtenir une solution optimale pour un critère donné.

Ces problèmes sont décrits en termes de variables (« inconnues ») et de contraintes (relations entre les variables), leur résolution consiste à affecter une valeur à chacune des variables de telle manière que les contraintes soient respectées [3].

Nous discutons dans ce chapitre des principes fondateurs de la Programmation Par Contraintes (PPC). Dans un premier temps, nous décrivons le modèle de déclaration des problèmes de satisfaction de contraintes et d'optimisation sous contraintes avant de discuter des méthodes de résolution des problèmes à base de contraintes et des techniques permettant d'accélérer le processus de résolution. Avant de conclure, nous présentons les principes de quelques solveurs de contraintes qui mettent à disposition des utilisateurs une bibliothèque pour la modélisation et la résolution de problèmes variés.

2. Modélisation d'un problème par des contraintes (CSP)

Nous définissons dans cette section les différents éléments permettant de modéliser un problème combinatoire par une conjonction de contraintes. Dans une première partie, nous définissons les problèmes de satisfaction de contraintes puis nous présentons les problèmes d'optimisation sous contraintes dont les solutions maximisent ou minimisent la valeur d'une fonction de coût [4].

2.1. Problème de satisfaction de contraintes

Un problème de satisfaction de contraintes (CSP) se définit par la donnée d'un ensemble de variables et d'un ensemble de contraintes. Chaque variable peut prendre une valeur choisie dans le domaine qui lui est associé. Les contraintes, quant à elles, décrivent les combinaisons autorisées de valeurs pour les variables. L'objectif est d'attribuer une valeur à chaque variable de sorte que toutes les contraintes soient satisfaites [4] [5].

2.1.1. Une contrainte

Une contrainte est une relation logique établie entre différentes variables, chacune prend sa valeur dans un ensemble qui lui est propre, appelé domaine. Une contrainte sur un ensemble de variables restreint les valeurs que peuvent prendre simultanément ses variables.

Une contrainte est déclarative et relationnelle puisqu'elle définit une relation entre les variables sans spécifier de procédure opérationnelle pour assurer cette relation.

2.1.2. Un arité d'une contrainte

L'arité d'une contrainte est égale à la cardinalité de son ensemble de variables, appelé son scope. Par exemple, la contrainte binaire $(x, y) \in \{(0, 1), (1, 2)\}$ précise que la variable x ne peut prendre la valeur 0 que si la variable y prend la valeur 1 et que la variable y peut prendre la valeur 2 uniquement si x prend la valeur 1. Une telle contrainte est définie en extension, c'est-à-dire en spécifiant explicitement les tuples de valeurs qu'elle autorise (ou interdit).

Une contrainte peut également être définie en intension par :

- une équation mathématique ($x + y \leq 4$),
- une relation logique simple $(x \leq 1 \Rightarrow y \neq 2)$
- un prédicat portant sur n variables $(allDifferent\ (x_1, \ldots, x_n))$.Une telle contrainte est appelée contrainte globale. Nous reviendrons sur la définition et l'intérêt des contraintes globales tout au long de ce mémoire.

La définition d'une nouvelle contrainte est aisée puisque la seule opération indispensable est le test de consistance qui détermine si une instanciation de ces variables satisfait la contrainte.

2.1.3. Définition d'un CSP formellement

Un problème de satisfaction de contraintes (CSP) est défini formellement par un triple (X, D, C) représentant [7]:

- un ensemble fini de variables X,
- une fonction D affectant à chaque variable $x \in X$ son domaine D(x)
- un ensemble fini de contraintes C.

Le domaine initial d'une variable $D_0(x)$ représente l'ensemble de valeurs auxquelles la variable x peut être instanciée avant le début de la résolution. Chaque contrainte $c \in C$ est une relation multidirectionnelle portant sur un sous-ensemble de variables noté $var(c) \subseteq X$

restreignant les valeurs que ces variables peuvent prendre simultanément. Les problèmes traités dans ce mémoire sont des CSP discrets ($D(x) \in Z$) et statiques, c'est-à-dire l'ensemble des contraintes ne change pas au cours de la résolution [7].

Donc un problème de satisfaction de contraintes (CSP) est un triplet (X, D, C)avec :

- X un ensemble $\{X_1, \dots\dots, X_k\}$de variables ;
- D un ensemble $\{D_1, \dots\dots, D_k\}$ de domaines : $X_i \in D_i, i = 1, \dots., n$;
- C un ensemble$\{C_1, \dots\dots, C_t\}$ de contraintes où chaque Ci définit un sous-ensemble du produit cartésien des domaines des variables sur lesquelles elle porte :

$$C_i\left(X_{i_1}, \dots\dots, X_{i_k}\right) \subseteq D_{i_1} \times \dots.\times D_{i_k}$$

La notion de domaine désigne l'ensemble de valeurs que peut potentiellement prendre une variable. Cette notion générique fait référence à des ensembles de natures potentiellement très différentes [10] :

- un ensemble de valeurs symboliques : on peut vouloir représenter des couleurs, e.g. D = {rouge, bleu, vert}, ou encore des jours de la semaine, par exemple $D = \{lundi, mercredi, samedi\}$;
- un ensemble d'entiers non contigus : il est possible d'utiliser un ensemble d'entiers quelconques. $D = \{5, 8, 12\}$;
- un intervalle d'entiers : on peut également définir un ensemble d'entiers contigus, en compréhension, ne spécifiant que les bornes inférieure et supérieure de l'intervalle, par exemple $D = \{1, \dots, 10\}$;
- un intervalle de réels : de la même façon, on peut définir en compréhension un ensemble de réels, en ne déclarant que ses bornes inférieure et supérieure. $D = [-\pi, \pi]$.

2.1.4. Une instanciation

Une instanciation $x \leftarrow v$ consiste à affecter à une variable x une valeur v appartenant à son domaine $D(x)$. À chaque étape de la résolution,

2.1.5. Une affectation

Une affectation partielle A est définie comme l'ensemble des domaines courants de toutes les variables. Le domaine courant $D(x)$ d'une variable x est toujours un sous-ensemble de son

domaine initial $D_0(x)$ (inclusion non stricte). On note $var\ (A)$ l'ensemble des variables instanciées, c'est-à-dire dont le domaine est réduit à un élément.

2.1.6. Une affectation partielle ou totale

Une affectation est dite partielle si $var\ (A)\ \subsetneq\ X$ ou totale si $var\ (A)\ =\ X$.

Une affectation A_t restreint une affectation partielle *A*, noté $A_t \subseteq A$, si le domaine de n'importe quelle variable *x* dans A_t est un sous-ensemble de son domaine dans *A*. Les valeurs minimum et maximum du domaine d'une variable entière *x* sont notées respectivement $min(x)\ et\ max(x)$. Les notations $min(x) \leftarrow\ v\ et\ max(x) \leftarrow\ v$ représentent respectivement les réductions du domaine à $D(x)\ \cap\ [v, +\infty [\ \ et\ \ D(x) \cap] - \infty, v]$.

2.1.7. Une affectation viole une contrainte

Une affectation viole une contrainte si toutes ses variables sont instanciées et que la relation associée à la contrainte n'est pas vérifiée.

2.1.8. Une affectation est consistante

Une affectation est consistante si elle ne viole aucune contrainte et inconsistante dans le cas contraire.

2.1.9. Une solution d'un CSP

Une solution d'un CSP est donc une affectation totale consistante. Résoudre un CSP consiste à exhiber une unique solution ou à montrer qu'aucune solution n'existe (le problème est irréalisable). Prouver la consistance d'un CSP est un problème NP-complet mais exhiber une solution est un problème NP-difficile dans le cas général. On peut également être intéressé par la détermination de plusieurs ou toutes les solutions d'un problème [4] [8].

2.2. Problème d'optimisation sous contraintes

Dans de nombreux cas, des relations de préférence entre les solutions d'un CSP existent eu égard aux critères fixés par le décideur. L'optimisation consiste alors à rechercher des solutions optimales d'un CSP au regard des relations de préférence du décideur. Ce critère d'optimalité est généralement associé à la maximisation / minimisation d'une fonction objectif d'un sous-ensemble de variables vers les entiers.

Un problème d'optimisation sous contraintes (COP) est un CSP augmenté d'une fonction objective *f*. Cette fonction est souvent modélisée par une variable dont le domaine est défini par les bornes supérieures et inférieures de f [3].

2.3. Exemple du problème des n reines

Le but de ce problème, inspiré du jeu d'échec, est de placer n reines sur un échiquier de dimension n × n de manière à ce qu'aucune ne soit en prise. Deux reines sont en prises si elles sont sur la même ligne, la même colonne ou la même diagonale. Un échiquier classique comporte 8 lignes et 8 colonnes. Ce problème désormais classique est devenu une référence de mesure de performance des systèmes grâce à un énoncé simple masquant sa difficulté.

Un modèle classique sans contraintes globales est défini dans les relations 2.1, 2.2 et 2.3. En observant que deux reines ne peuvent pas être placées sur la même colonne, on peut imposer que la reine *i* soit sur la colonne *i*. Ainsi, la variable l_i de domaine {1, . . . , n} représente la ligne où est placée la reine dans la colonne *i*.

$$l_i \neq l_j \qquad 1 \leq i < j \leq n \qquad (2.1)$$

$$l_i \neq l_j + (j - i) \qquad 1 \leq i < j \leq n \qquad (2.2)$$

$$l_i \neq l_j - (j - i) \qquad 1 \leq i < j \leq n \qquad (2.3)$$

La contrainte (2.1) imposent que les reines soient sur des lignes différentes alors que les contraintes (2.2) et (2.3) imposent que deux reines soient placées sur des diagonales différentes.

les contraintes précédentes montrent plusieurs affectations partielles pour le problème des 4 reines dans lesquelles le placement d'une reine sur l'échiquier est représenté par un cercle dans la case correspondante. Les reines sont ordonnées de gauche à droite et les positions de haut en bas. On peut observer que l'affectation partielle $\{l_1 \leftarrow 1, l_2 \leftarrow 3, l_3 \leftarrow 2\}$ n'est pas consistante car elle viole une des contraintes (2.2). Au contraire, les affectations totales $\{l_1 \leftarrow 2, l_2 \leftarrow 4, l_3 \leftarrow 1, l_4 \leftarrow 3\}$ *et* $\{l_1 \leftarrow 3, l_2 \leftarrow 1, l_3 \leftarrow 4, l_4 \leftarrow 2\}$ sont les deux solutions symétriques des 4 reines.

Si l'on définit un COP à partir du problème des n reines en définissant la fonction objectif $min\ (l_1)$, alors le problème admet une unique solution optimale : $\{l_1 \leftarrow 2, l_2 \leftarrow 4, l_3 \leftarrow 1, l_4 \leftarrow 3\}$.

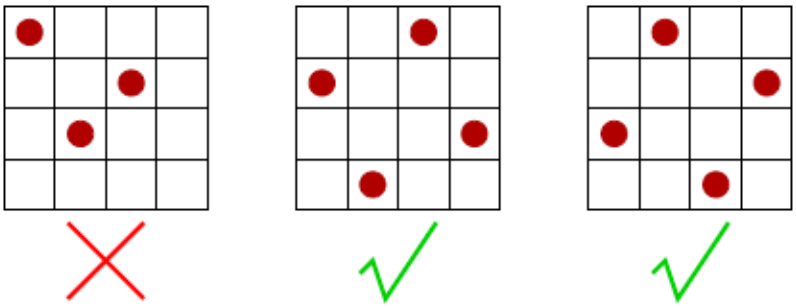

Figure 2. 1 Exemples d'affectation : 4 reines.

3. Méthodes de résolution

Les méthodes de résolution des *CSPs* sont génériques, c'est-à-dire qu'elles ne dépendent pas de l'instance à résoudre. Cependant, des techniques dédiées améliorent la résolution de différentes classes de problèmes. Dans le contexte d'un *CSP* statique et discret, nous discuterons de la réduction de l'espace de recherche par des techniques de consistance couplées si nécessaire à un algorithme de recherche arborescente. [3]

3.1. Algorithmes simples de recherche

3.1.1. L'algorithme generate-and-test

Il énumère les affectations totales et vérifie leur consistance, c'est-à-dire qu'aucune contrainte n'est violée. En général, les affectations sont énumérées grâce à une recherche arborescente qui instancie itérativement les variables. Cet algorithme ne réveille les contraintes que lorsqu'une affectation est totale, mais considère par contre un nombre exponentiel d'affectations le rendant impraticable même pour des problèmes de petite taille. Des résultats de complexité ont montré que prouver la satisfiabilité d'un CSP était un problème NP-complet mais qu'exhiber une solution admissible ou optimale était un problème NP-difficile [5].

3.1.2. L'algorithme simple retour arrière (backtrack)

Il étend progressivement l'affectation vide en instanciant une nouvelle variable à chaque étape. L'algorithme vérifie alors que l'affectation partielle étendue est consistante avec les contraintes du problème. Dans le cas contraire, la dernière instanciation faite est remise en cause et l'on effectue une nouvelle instanciation. De cette manière, l'algorithme construit un arbre de recherche dont les nœuds représentent les affectations partielles testées. Cet algorithme teste l'ensemble des affectations possibles de manière implicite, c'est-à-dire sans les générer toutes. Le nombre d'affectations considéré est ainsi considérablement réduit, mais en revanche, la consistance des contraintes est vérifiée à chaque affectation partielle.

La figure 2.2 illustre la résolution du problème des 4 reines par l'algorithme backtrack qui place à chaque point de choix la première reine libre en partant de la gauche à la première position disponible en partant du haut. À chaque point de choix, l'affectation partielle courante est étendue en instanciant les variables et les valeurs selon l'ordre lexicographique. Dans ce cas précis, toutes les solutions symétriques sont éliminées en imposant que la première reine (à gauche) soit placée dans la partie haute de l'échiquier. Remarquez que la recherche continue après la découverte de la première solution puisqu'on cherche toutes les solutions à une symétrie près. L'algorithme generate-and-test consiste à déployer l'arbre entier [4].

La découverte redondante d'inconsistance locale due à la perte d'information sur l'inconsistance d'affectation partielle dégrade les performances de l'algorithme backtrack. Nous discuterons donc de l'utilisation active des contraintes pour supprimer les inconsistances, des algorithmes prospectifs qui anticipent les prochaines affectations pour réduire les domaines, et rétrospectifs qui utilisent les conflits pour remettre en cause les choix antérieurs.

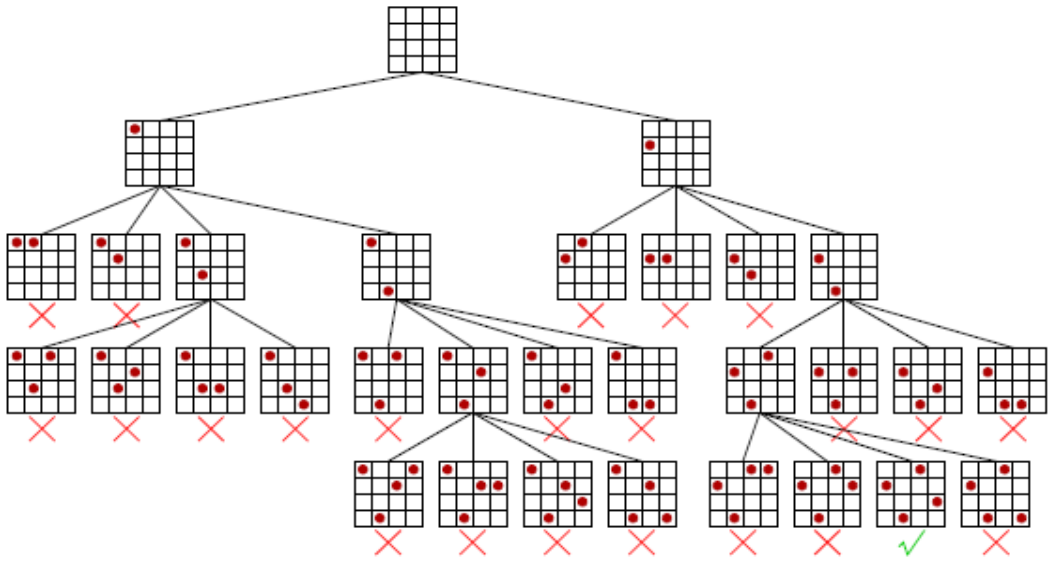

Figure 2. 2 L'algorithme simple retour arrière (4 reines)

3.2. Filtrage et propagation des contraintes

Le filtrage et la propagation des contraintes permettent d'améliorer les capacités de résolution des CSP lors de la construction d'un arbre de recherche. Une fonction revise() associée à chaque contrainte réalise le filtrage des domaines, c'est-à-dire qu'elle supprime les valeurs inconsistantes des domaines de ces variables. Différents niveaux de consistance locale existent pour une même contrainte. La propagation calcule un point fixe global qui est atteint

lorsque les techniques de consistance locale ne peuvent plus réaliser aucune inférence. En général, les algorithmes de consistance sont incomplets, c'est-à-dire qu'ils ne retirent pas toutes les valeurs inconsistantes des domaines [6].

Les opérations (2.4) et (2.5) définissent une fonction revise () pour la contrainte arithmétique $x < y$. Supposons que les domaines de x et y sont égaux à {1, 2, 3}, alors l'opération (2.4) réduit le domaine de *x* à {1, 2} tandis que l'opération (2.5) réduit le domaine de *y* à {2, 3}. La fonction revise () a atteint son point fixe puisqu'aucune règle ne peut plus produire d'inférence. Dans de nombreux cas, la fonction revise () doit être appelée plusieurs fois avant d'atteindre son point fixe [4].

$$max(x) \leftarrow max(y) - 1 \qquad (2.4)$$

$$min(y) \leftarrow min(x) + 1 \qquad (2.5)$$

Nous rappelons les principales techniques de consistance.

Un CSP est dit consistant de nœud si pour toute variable x et pour toute valeur $v \in D(x)$, l'affectation partielle $x \leftarrow v$ satisfait toutes les contraintes unaires, c'est-à-dire dans lesquelles x est l'unique variable non instanciée.

Une contrainte est dite arc- consistante si pour chaque valeur de chaque variable x, il existe une affectation des autres variables telle que la contrainte soit satisfaite.

Un CSP est dit arc-consistant lorsque toutes ses contraintes sont arc- consistantes. Néanmoins, il est insuffisant d'effectuer une révision unique par contrainte pour atteindre le point fixe global. Par conséquent, l'algorithme AC-1 révise toutes les contraintes jusqu'à ce qu'aucun domaine n'ait changé. L'algorithme AC-3 utilise une file de contraintes à réviser dans laquelle il ajoute les contraintes portant sur une variable dont le domaine a changé. En pratique, les algorithmes à partir d'AC-3 affinent le réveil des contraintes en fonction d'évènements sur les domaines définis par le solveur (réduction, suppression, instanciation).

Lorsque les domaines des variables sont trop grands, la mémoire requise pour stocker l'appartenance ou non de chaque valeur devient problématique. De la même manière, l'application des techniques de consistance pour chaque couple de variable et de valeur peut dégrader considérablement la vitesse de propagation par rapport à la réduction effective des domaines. On utilise alors la consistance de bornes qui consiste à raisonner sur la valeur minimale et maximale que les variables peuvent prendre. Pour certaines contraintes, la

consistance de borne est très proche, voire égale à la consistance d'arc, notamment pour la contrainte $x < y$ discutée ci-dessus.

Ainsi, à chaque nœud de l'arbre de recherche, c'est-à-dire à chaque instanciation d'une variable, l'algorithme de recherche applique des techniques de consistance locale, propre à chaque contrainte, qui retirent des valeurs inconsistantes des domaines. À leur tour, les modifications des domaines peuvent inférer de nouvelles inconsistances. Ce mécanisme, appelé propagation, continue jusqu'à ce qu'un point fixe global soit atteint. L'algorithme fait alors un nouveau choix de variable et de valeur et teste la nouvelle instanciation. Si durant le filtrage d'une contrainte, le domaine d'une variable devient vide, alors l'instanciation courante est inconsistante. Le nœud est fermé et l'algorithme passe au nœud suivant s'il existe. Cette opération permet d'agir sur les domaines de toutes les variables d'un CSP durant la construction des affectations partielles réduisant ainsi le nombre de nœuds composant l'arbre.

3.3. Contraintes globales

La modélisation des problèmes complexes est facilitée par l'utilisation de contraintes hétérogènes agissant indépendamment sur de petits ensembles de variables. Toutefois, la détection locale des inconsistances affaiblit la réduction des domaines. Les contraintes globales corrigent partiellement ce comportement en utilisant l'information sémantique issue de raisonnements sur des sous-problèmes. Elles permettent généralement d'augmenter l'efficacité du filtrage ou de réduire les temps de calcul.

Par exemple, l'arc-consistance ne détecte pas l'inconsistance globale du CSP présenté dans la contrainte suivante.

$x \neq y ; x \neq z \ ; \ z \neq y$

$D_0(x) = D_0(y) = D_0(z) = \{0, 1\}$

En effet, l'inconsistance est issue des trois contraintes qui imposent que les trois variables prennent des valeurs distinctes alors que l'union de leurs domaines ne contient que deux valeurs. La contrainte globale $allDifferent\,(x_1, \ldots, x_n)$ qui impose que ses variables prennent des valeurs distinctes détecte cette inconsistance triviale.

Un second modèle classique pour le problème des reines défini dans la figure 2.2 utilise des contraintes globales *allDifferent* pour représenter les cliques d'inégalités binaires. On

introduit généralement les variables auxiliaires x_i et y_i par le biais des contraintes de liaison (2.6). Les variables auxiliaires correspondent aux projections sur la première colonne de la reine i en suivant les diagonales. La contrainte (2.6) impose que les reines soient sur des colonnes différentes alors que les contraintes (2.7) et (2.8) imposent que deux reines soient placées sur des diagonales différentes [4].$allDifferent\ (y_1, \ldots, y_n)$

$$x_i = l_i - iy_i = l_i + i \qquad 1 \leq i \leq n \qquad (2.6)$$

$$allDifferent\ (l_1, \ldots, l_n) \qquad (2.7)$$

$$allDifferent\ (x_1, \ldots, x_n) \qquad (2.8)$$

4. Solveurs de contraintes

Nous citons quelques outils de programmation par contraintes libres ou commerciaux.. Dans un souci de concision, nous ne citerons que les solveurs dédiés aux problèmes booléens (SAT) et à la programmation mathématique mentionnés. Les outils présentés diffèrent en plusieurs points : les entités qu'ils sont capables de traiter (entiers, ensembles, objets . . .) ; les contraintes et algorithmes de recherche implémentés ; le langage hôte sur lequel il s'appuie. Historiquement, les premiers solveurs étaient des extensions du langage déclaratif de programmation logique Prolog dont l'algorithme d'unification est une clé. De nos jours, les extensions de Prolog les plus populaires sont ECLiPSe et SICStus Prolog qui proposent toutefois des interfaces vers des langages objet tels que Java, .NET, C et C++ ainsi qu'une compatibilité avec les langages d'autres solveurs. Les langages C et C++ ont connu beaucoup de succès grâce à leur rapidité et la gestion optimisée de la mémoire. Il existe quelques solveurs libres implémentés en Java comme choco et *koalog* malgré des critiques concernant la gestion de la mémoire, notamment le ramasse-miettes (garbage collector), et les performances des programmes. Pour répondre à ces critiques, on peut d'abord remarquer que les performances des solveurs dépendent bien plus des technologies embarquées que du langage hôte. Cependant, cet effort d'uniformisation des contraintes globales et des langages de description de CSP ne permettent pas encore de déclarer un problème sans aucune considération du solveur sous-jacent. Simultanément, il est encore nécessaire de développer des contraintes globales dédiées à la résolution d'un problème précis [4].

5. Conclusion

Nous avons décrit dans ce chapitre les principes de la Programmation Par Contraintes (PPC). Cette approche déclarative permet de résoudre des problèmes combinatoires variés. Les utilisateurs décrivent leur problème en posant des contraintes sur les valeurs que peuvent

prendre les différentes variables composant le problème. Un solveur de contraintes est alors chargé de calculer les solutions du problème. Le processus de résolution exacte de problèmes de satisfaction de contraintes permet de générer efficacement les solutions d'un problème grâce à la combinaison des techniques de filtrage et des algorithmes de recherche.

CHAPITRE : 03

MODELISATION PAR CSP

1. Introduction

La programmation par contraintes est un paradigme générique de résolution de problèmes dont l'une des principales caractéristiques est de distinguer deux grandes tâches principales :

- modéliser le problème à résoudre : cette première tâche consiste à formaliser le problème à l'aide de concepts dédiés, dans le but de le reformuler de manière plus abstraite.
- résoudre le modèle ainsi obtenu : cette seconde étape vise à appliquer des techniques de résolution qui vont permettre de trouver une (des) solution(s) s'il en existe [11].

L'objectif de cette chapitre est d'adopter le formalisme de CSP pour modéliser le problème des emplois du temps de département d'informatique de l'université de M'sila, Ce problème se caractérise par une combinatoire élevée (il est NP-Complet) et il n'y a pas de résolution efficace dans le cas général.

Pour la simplicité de travail on a décomposé notre modélisation en 3 parties. Puisque l'augmentation de la taille du problème rend la tâche très difficile. Pour chercher d'une solution dans une tempe raisonnable. De cette façon, le problème global d'emplois du temps peut être représenté comme une union de trois sous-problèmes:

1. L'affectation des séances aux jours (dimanche,…, jeudi).
2. L'affectation de chaque séance un intervalle de temps(8 :00 - 9 :30,…. ,15 :30 - 17 :00) pour chaque jour
3. L'affectation des salles.

Pour libéré tous les groupe de même section pendant les séances de cours, on applique les deux premiers parties de cette model sur les séances de cours avant les deux type de séance TD et TP. En autre façon, Nous commençons l'affectation des séances de *cours* avant les séances de *TD* et *TP* dans la semaine.

Dans les problèmes des emplois du temps, 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.

2. 1ère partie : Affectation des séances aux jours

Les séances sont allouées aux jours en considérant des contraintes suivantes :

2.1. Les contraintes dures :

1. Toutes les séances devraient être assignées à un jour.
2. La capacité quotidienne de la salle ne doit pas être dépassée.
3. la charge maximale et minimale des heures d'un enseignant devrait être prise en compte
4. Le temps de repos de l'étudiant et l'enseignant devrait être pris en compte

2.2. Les contraintes de préférences :

1. rassembler les séances de l'enseignant dans deux jours consécutifs
2. la séance de cours doit être planifie avant les séances de TD et TP
3. quatre séances de cours par jour c'est interdit

Si toutes les contraintes sont considérées, on verra que ce problème est un problème de satisfaction des contraintes (CSP) où les séances sont des variables et les jours sont les domaines des variables.

- **Les Variables :**

  $\boldsymbol{J_{s_j}}$: Le jour de séance s_j dans la semaine.

- **Le domaine de variable :**

  $\boldsymbol{J_{s_j}} \in \{0, \dots, 4\}$

- **Les contraintes** :

L'objectif est de trouver une solution satisfaisant les contraintes suivantes :

- **Les contraintes dures :**

C1: $J_{s_j} \in \{0, \dots, 4\}, \qquad \forall s_j \in S$

C2: $\sum_{s_j \in s_{sl_k}} f\left(J_{s_j}, t\right) \leq d_{sl_k}, \qquad \forall sl_k \in Sl, t \in \{0, \dots, 4\}$

C3: $h_{E_{min}} \leq 1.5 \times \left[\sum_t \sum_{s_j \in s_{E_i}} f\left(J_{s_j}, t\right) \times cof_{s_j}\right] \leq h_{E_{max}}, \quad \forall E_i \in E, t \in \{0, \dots, 4\}$

C4: $\sum_{s_j \in s_{g_i}} f\left(J_{s_j}, t\right) \leq 5, \qquad \forall g_i \in G\ , t \in \{0, \dots, 4\}$

C5: $\sum_{s_j \in s_{E_i}} f\left(J_{s_j}, t\right) \leq 5, \qquad \forall E_i \in E, t \in \{0, \dots, 4\}$

- **Les contraintes de préférences :**

C6: $\forall s_j \in s_{E_i},\ J_{s_j} \in \{a_{E_i}, b_{E_i}\},$ $\forall\ E_i \in E$

C7: $J_c \leq J_t \ \wedge\ J_c \leq J_v \forall\ (s_c, s_t, s_v) \in T$

C8: $\sum_{s_j \in \{s_{g_i} \cap s_{sl_4}\}} f\left(J_{s_j}, t\right) \leq 3,$ $\forall g_i \in G\ , t \in \{0, \dots, 4\}$

Les définitions et notations suivantes sont nécessaires pour décrire le modèle.

S	L'ensemble des séances, $\{s_1, \dots \dots \dots, s_n\}$
cof	L'ensemble des coefficients $\{cof_1, cof_2, cof_3\}$
cof_{S_j}	Le coefficient de séance s_j
E	L'ensemble des enseignants $\{E_1, \dots \dots \dots, E_m\}$
S_{E_i}	L'ensemble des séances données par enseignant E_i
$h_{E_{max}}, h_{E_{min}}$	Le nombre maximum et minimum des heures d'enseignant peut enseigner
SC	L'ensemble des sections
G	L'ensemble des groupes d'étudiants $\{g_1, \dots \dots \dots, g_k\}$
G_{SC_i}	L'ensemble des groupes d'étudiants de même section
S_{g_i}	L'ensemble des séances données par groupe d'étudiant g_i
Sl	L'ensemble des types de salles, $\{sl_1, \dots., sl_4\}$
S_{sl_k}	Ensemble de séances qui nécessitent de salle $sl_k (sl_k \in Sl)$
d_{sl_k}	Capacité maximale de salle sl_k par jour $(sl_k \in Sl)$
a_{E_i}, b_{E_i}	Deux jours choisis par l'enseignant peuvent enseigner
T	L'ensemble des triples des séances s_c, s_t, s_v de même module
$f(t_1, t_2) = \begin{cases} 1, & si\ t_1 = t_2. \\ 0, & sinon. \end{cases}$	Fonction pour vérifier l'égalité

Tableau 3. 1 définitions et notations nécessaires pour décrire le modèle (Partie 1)

2.3. Description des contraintes :

Dans le formalisme CSP, la recherche d'une solution requiert de satisfaire toutes les contraintes. Dans la mesure ou certains contraintes doivent être obligatoirement satisfaites, on les qualifie généralement des contraintes dures, cependant pour certains problèmes réels, certains contraintes (dites molles) ne traduisent, dans la réalité qu'une préférence d'une possibilité…le formalisme valu peut jouer sans doute un rôle très important dans l'optimisation de la qualité des emploi du temps pour répondre au mieux au besoin des

étudiants et des enseignants avec une très bonne gestion des ressources tel que le problème d'affectation des salles. Les contraintes dures traduisent une interdiction des empiétements des ressources, alors que pour qu'une contrainte dure soit satisfaite il faut que l'implication suivante soit valide [4].

C1: $J_{s_j} \in \{0, \dots, 4\}, \qquad \forall s_j \in S$

Cette contrainte vérifie que la séance affectée dans les jours de semaine. On notera que 0 signifié le première jour de la semaine est dimanche et 1 est lundi, 2 est mardi …etc.

C2: $\sum_{s_j \in s_{sl_k}} f\left(J_{s_j}, t\right) \leq d_{sl_k}, \qquad \forall sl_k \in Sl, t \in \{0, \dots, 4\}$

Au C2, il a été déclaré que chaque séance nécessite une salle d'un type spécifique, Dans ce problème, il existe quatre types de salle *{*salle *TD, TP, Multimédia, amphis}*

Les capacités de salle quotidiennes (sl_k) dépendent du nombre de salle k appartenant au type de salle représenté par cette salle. Par exemple, avoir 15 salles de TD, signifie que la capacité quotidienne de salle de TD est 15×6=90 séances (nombre des salles × nombre des intervalles de temps dans une journée).

$$\underbrace{\text{[8 :00 - 9 :30]},\dots., \text{[15 :30 - 17 :00]}}_{6}$$

C3: $h_{E_{min}} \leq 1.5 \times \left[\sum_t \sum_{s_j \in s_{E_i}} f\left(J_{s_j}, t\right) \times cof_{s_j}\right] \leq h_{E_{max}}, \quad \forall E_i \in E, t \in \{0, \dots, 4\}$

Au cours de l'affectation des séances dans les jours, il convient de rappeler que le nombre total des heures auxquelles l'enseignant par semaine peut être au minimum **9 heures** et maximum **18 heures**, pour calculer La durée d'une séance de l'enseignant :

La durée de séance = la durée unitaire (généralement 1 heure 30 minutes) × **le coefficient** de type de séance.

On rappelle que **le coefficient** de séance de cours est 1.5 et la séance de Travaux dirigés (TD) est 1 et pour la séance de Travaux pratiques (TP) est de 0.75.

C4: $\sum_{s_j \in s_{g_i}} f\left(J_{s_j}, t\right) \leq 5, \qquad \forall g_i \in G\ , t \in \{0, \dots, 4\}$

C5: $\sum_{s_j \in s_{E_i}} f\left(J_{s_j}, t\right) \leq 5, \qquad \forall E_i \in E, t \in \{0, \dots, 4\}$

Les deux contraintes C4 et C5 concerne le temps de repos de l'étudiant et de l'enseignant. Cette séance s'appelle séance de « **break time**» pour faire le déjeuner. Pour satisfaire cette contrainte il faut limiter le nombre de séances par jour à cinq séances au maximum.

C6: $\forall s_j \in s_{E_i},\ J_{s_j} \in \{a_{E_i}, b_{E_i}\}, \qquad \forall\ E_i \in E$

Cette contrainte concerne les deux jours de disponibilité de l'enseignant, il est préférable de rassemblé leurs séances dans deux jours consécutifs, pour cette raison. Il est possible de modifier la limite des séances en jour de l'enseignant à sa demande (la contrainte C5) pour satisfaire cette contrainte.

C7: $J_{s_c} \leq J_{s_t} \wedge J_{s_c} \leq J_{s_v} \forall\ (s_c, s_t, s_v) \in T$

Cette dernière contrainte concerne le jour entre les séances de cours et TD, et TP de même module. Cette contrainte assurée que le jour des séances de TD et TP ne peuvent pas être programmés avant le jour de cours, On notera que s_c est une séance de cour, s_v est une séance de TD, s_t est une séance de TP.

C8: $\sum_{s_j \in \{s_{g_i} \cap s_{sl_4}\}} f\left(J_{s_j}, t\right) \leq 3, \qquad \forall g_i \in G\ , t \in \{0, \dots, 4\}$

Cette dernière contrainte de cette étape concerne le nombre maximum des séances de cours pédagogiques par jour, afin de les comprendre par les étudiants et pour ne pas perdre la concentration, il faut prendre au maximum trois cours par jour.

3. 2ème partie : Construire le programme quotidien

Dans cette partie, les séances qui sont assignées à un jour spécifique par l'étape précédente ont un intervalle de temps spécifique du jour (8 :00 - 9 :30,…. ,15 :30 - 17 :00), pour satisfaire des contraintes :

3.1. Les contraintes de dures :

1. Chaque séance devrait être programmée à un intervalle de temps en veillant à ce qu'elle se termine à 17h00 au plus tard.
2. À chaque intervalle de temps, l'exigence de capacité totale des séances pour un type de salle spécifique ne peut pas dépasser la capacité disponible de ce type de salle.
3. Deux séances enseignées par le même d'enseignant ne peuvent être planifiées sur de même intervalle de temps.
4. Deux séances d'un groupe d'étudiants ne peuvent être planifiées sur de même intervalle de temps.
5. Le temps de repos de l'étudiant devrait être pris en compte

3.2. Les contraintes de préférences

6. Le temps de repos d'enseignant devrait être pris en compte
7. la séance de cours doit être planifie avant les séances de *TD* et *TP*

Les définitions et notations dans le tableau 3.2 sont nécessaires pour décrire le modèle.

S	L'ensemble des séances, $\{s_1, \dots \dots \dots, s_n\}$
E	L'ensemble des enseignants $\{E_1, \dots \dots \dots, E_m\}$
S_{E_i}	L'ensemble des séances données par enseignant E_i
G	L'ensemble des groupes d'étudiants $\{g_1, \dots \dots \dots, g_k\}$
S_{g_i}	L'ensemble des séances données par groupe d'étudiant g_i
Br_{g_i}	Le temps de repos de groupe g_i (break time)
Br_{E_i}	Le temps de repos d'enseignant E_i (break time)
Sl	L'ensemble des types des salles-, $\{sl_1, \dots., sl_4\}$
S_{sl_k}	Ensemble de séances qui nécessitent desalles$sl_k (sl_k \in Sl)$
e_{sl_k}	Capacité horaire maximale de salle sl_k $(sl_k \in Sl)$
$f(t_1, t_2) = \begin{cases} 1, & si\ t_1 = t_2 \\ 0, & sinon. \end{cases}$	Fonction pour vérifier l'égalité

Tableau 3. 2 définitions et notations nécessaires pour décrire le modèle (Partie 2)

- **Les variables :**

  T_{s_j} Le début de séance

- **Le domaine de variable :**

  $T_{s_j} \in \{0,1, \dots .,5\}$

- **Les contraintes :**

C1: $T_{s_j} \in \{0, \dots ,5\}, \forall s_j \in S.$

C2: $\sum_{s_j \in s_{sl_k}} f\left(T_{s_j}, t\right) \leq e_{sl_k},\ \forall sl_k \in Sl, t \in \{0, \dots ,5\}.$

C3: $\sum_{s_j \in s_{E_i}} f\left(T_{s_j}, t\right) \leq 1,\ \forall E_i \in E, t \in \{0, \dots ,5\}.$

C4: $\sum_{s_j \in s_{g_i}} f\left(T_{s_j}, t\right) \leq 1,\ \forall g_i \in G, t \in \{0, \dots ,5\}.$

C5: $\sum_{s_j \in s_{g_i}} f\left(T_{s_j}, Br_{g_i}\right) = 0,\ \forall g_i \in G.$

C6: $\sum_{s_j \in s_{E_i}} f\left(T_{s_j}, Br_{E_i}\right) = 0,\ \forall E_i \in E$

C7: $T_{s_c} \leq\ T_{s_t}\ \wedge\ T_{s_c} \leq T_{s_v} \forall\ (s_c, s_t, s_v) \in T$

3.3. Description des contraintes :

C1: $T_{s_j} \in \{0, \dots ,5\}, \forall s_j \in S.$

La première contrainte garantit que si une séance est attribuée à un jour par la première étape, elle devrait être programmée sur un intervalle de temps ce jour-là, On notera que 0 signifié la première séance dans le jour 8 :00 - 9 :30 et 1 est deuxième séance 9 :30 - 11 :00…etc.

C2: $\sum_{s_j \in s_{sl_k}} f\left(T_{s_j}, t\right) \leq e_{sl_k},\ \forall sl_k \in Sl, t \in \{0, \dots ,5\}.$

La capacité horaire maximale de salle est utilisées, ce qui correspond au nombre de salle dans ce type de salle. Pour n'importe quel intervalle de temps, les séances utilisant un type de salle spécifique ne peuvent pas dépasser le nombre de salles dans ce type de salle, par exemple il y a 10 salles de TP, donc la capacité horaire maximal des salles TP est 10 séances, En d'autres termes, impossible d'affecter au plus 10 séances de TP dans un même intervalle de temps.

C3: $\sum_{s_j \in s_{E_i}} f\left(T_{s_j}, t\right) \leq 1,\ \forall E_i \in E, t \in \{0, \dots ,5\}.$

C4: $\sum_{s_j \in s_{g_i}} f\left(T_{s_j}, t\right) \leq 1,\ \forall g_i \in G, t \in \{0, \dots, 5\}$.

Les deux contraintes C3 et C4 concerne les conflits dans les séances, les enseignants ne peuvent pas enseigner plus d'une séance et un groupe ne peut pas assister au plus d'une séance à un intervalle de temps.

C5: $\sum_{s_j \in s_{g_i}} f\left(T_{s_j}, Br_{g_i}\right) = 0,\ \forall g_i \in G$.

C6: $\sum_{s_j \in s_{E_i}} f\left(T_{s_j}, Br_{E_i}\right) = 0,\ \forall E_i \in E$

Les deux dernières contraintes de cette étape concernent le temps de repos de l'étudiant et d'enseignant, les deux contraintes C4 et C5 de première partie garantissent une séance libre, mais ces contraintes ont garantis l'affectation d'une séance dans un intervalle de temps $Br_{g_i}\ et\ Br_{E_i}$, (11:00 − 12:30 ou 12:30 − 14:00), On notera que Br_{g_i}est le temps de reposde groupe g_i choisir par d'administration, et Br_{E_i}est le temps de repos d'enseignant E_i choisir par lui-même.

C7: $T_{s_c} \leq T_{s_t} \wedge T_{s_c} \leq T_{s_v} \forall\ (s_c, s_t, s_v) \in T$

La dernière contrainte de cette étape concerne le temps entre les séances de cours et TD, et TP de même module. Cette contrainte assure que le début des séances de TD et TP ne peuvent pas être programmés avant le début de cours, On note que s_c est une séance de cour, s_v est une séance de TD, s_t est une séance de TP.

4. 3ème parie : Affectation des salles :

Il existe quatre types de salle, *salle TD*, *TP*, *Multimédia*, *amphis* avec des capacités des étudiants déférents, En fait, chaque séance peut être affectée à une salle ayant une capacité suffisante à des étudiants. Par exemple, une séance avec 35 étudiants peut être affectée à une salle de 35 au plus au lieu d'une salle de 30. Cela est possible grâce au modèle actuel. Le problème est de trouver des salles appropriées pour les séances prévues ce jour-là satisfaisant aux contraintes :

C1: Chaque séance devrait être assignée à une salle.

C2: Il peut y avoir au plus une séance dans une salle pendant un intervalle de temps.

Les définitions et les notations dans le tableau 3.3 sont nécessaires pour décrire le modèle.

S	L'ensemble des séances, $\{s_1, \dots \dots \dots, s_n\}$
Sl	L'ensemble des types dessalles, $\{sl_1, \dots, sl_4\}$
S_{sl_k}	Ensemble de séances qui nécessitent desalles$l_k (sl_k \in Sl)$
C_{sl_k}	Ensemble des salles qui sont dans le type sl_k
T_{s_j}	Le début de séance s_j

Tableau 3. 3 définitions et notations nécessaires pour décrire le modèle (Partie 3)

- **Variable :**

$$As_k(s_j, i, t) = \begin{cases} 1, & si\ la\ séance\ s_j\ qui\ se\ déroule\ \text{à}\ la\ salle\ i^{ème}\ de\ C_{sl_k}\ \text{à}\ partir\ de\ t. \\ 0, & sinon \end{cases}$$

- **Domaine de variable :**

$As_k \in \{0,1\}, \quad \forall\, k \in \{1,..,4\}.$

- **Les contraintes :**

1. $\sum_{i \in C_{sl_k}} As_k\left(s_j, i, T_{s_j}\right) = 1 \qquad \forall\, k \in \{1,..,4\}, \forall\, s_j \in S_{sl_k}$
2. $\sum_{s_j \in S_{sl_k}} As_k(s_j, i, t) \leq 1 \qquad \forall\, k \in \{1,..,4\}, \forall\, i \in C_{sl_k}, \forall\, t \in \{0,..,5\},$

4.1. Description des contraintes :

C1: $\sum_{i \in C_{sl_k}} As_k\left(s_j, i, T_{s_j}\right) = 1, \quad \forall\, k \in \{0,..,4\}, \forall\, s_j \in S_{sl_k}$

La première contrainte C1 assure que chaque séance devrait être affectée à une salle dont vous avez besoin, pour les quatre type des salles *TD*, *TP*, *Multimédia*, *amphis*, la fonction *affectation des salles* (*As*) garantie une salle i de types l_k ayant une capacité suffisante pour les étudiants pour une séances s_j à partir de période T_{s_j}.

C2: $\sum_{s_j \in S_{sl_k}} As_k(s_j, i, t) \leq 1, \ \forall\, k \in \{1,..,4\}, \forall\, i \in C_{sl_k}, \forall\, t \in \{0,..,5\}.$

La dernière contrainte de cette étape assure que chaque intervalle de temps {0,….5}, et quel que soit le type de salle, la salle ***i*** ne peut pas accueillir deux séances différentes à la même période ***t.***

5. Conclusion

Le problème de planification des emplois du temps est considéré comme étant le problème académique, le plus contraignant dans le système de gestion universitaire, Dans ce chapitre nous avons essayé proposé notre modélisation de problème des emplois du temps, basées sur la notion de programmation par contraintes avec description détaillée du processus de modélisation du problème.

CONCLUSION GENERALE

Le problème des emplois du temps universitaires est l'un des problèmes de planification intensivement étudiés. C'est un problème extrêmement complexe. Il est classé parmi les problèmes combinatoires difficiles au sens fort. Cette complexité est due à l'explosion combinatoire du nombre de solutions qui croît exponentiellement avec la taille du problème, Parmi les méthodes utilisées pour résoudre ce genre de problème, il s'impose le paradigme de la programmation par contraintes comme une approche très prometteuse.

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 à ce type de problème est représentable par un ensemble de propriétés. Le but est d'obtenir 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 lieus. Aussi l'estimation du degré de satisfaction des contraintes de préférence est souvent difficile à formuler. En plus, du fait qu'elles peuvent être parfois contradictoires.

La programmation par contraintes, en plus de leur adaptabilité aux différents problèmes combinatoires, ont l'avantage de ne parcourir qu'une faible fraction de l'espace de solutions pour parvenir à une solution acceptable.

Dans ce travail, nous avons traité la problématique de gestion des emplois du temps. Un cadrage théorique de notre thème a consisté de présenter ses trois volets : L'état de l'art de problèmes de planification d'horaires de travail, L'état de l'art de programmation par contraintes et problème de satisfaction des contraintes. La troisième partie du travail s'est attachée à modéliser notre problème de gestion des emplois du temps en se basant sur des principes de la programmation par contraintes.

A l'issue du travail présenté dans ce mémoire, différentes pistes formant les directions futures à cette recherche, sont envisagées :

- Nous recommandons de compléter l'implémentation de notre mobilisation et obtenir des résultats satisfaisants.

- Essayer d'appliquer une hybridation avec une autre méthode de type méta-heuristique.
- Modélisation tous les contraintes liées au problème d'emploi du temps de département de l'informatique de l'université de M'sila. par exemple, les qualifications des enseignants, ses préférences, ses compétences, et les contraintes des ressources matérielles, disponibilité des appareils (data show...).etc.

BIBLIOGRAPHIE

[1] T. Fatiha Résolution du problème de l'emploi du temps, Proposition d'un algorithme évolutionnaire multi objectif, Mémoire de Magister, Université Mentouri Constantine, 2006.

[2] H. Hicham, C. Belgacem, Conception et développement d'une application d'optimisation basée sur les systèmes multi-agents cas d'étude « le problème de l'emploi du temps », thèse de master, Université kasdimerbahouargla, 2013.

[3] N. Barnier, Application de la programmation de contraintes à des problèmes de gestion de trafic aérien, Thèse de doctorat, France, 2002

[4] A. Malapert, Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes, thèse de doctorat, Université de nantes, 2011.

[5] C. Terrioux « Approches structurelles et coopératives pour la résolution des problèmes de satisfaction de contraintes », Thèse ne l'Université de Marseilles 2002.

[6] M. Belaissaoui, Problèmes de satisfaction des contraintes valués : Technique de résolution des emplois du temps universitaires, Conférence : Conférence maghrébine sur les technologies de l'information MCSEAI 2006, À l'Université Ibn Zohr, Agadir, Maroc

[7] T. Moyaux, Satisfaction distribuée de contraintes et son application à la génération d'un emploi du temps d'employés.Article de doctorat, Université Laval, QuébecCanada

[8] EmmanueRachelson, OPL et l'environnement CPLEX Studio, http://emmanuel.rachelson.free.frconsulté le : 06/05/2017

[9] D. Christoph, Programmation par contraintes et programmation mathématique,Ecole Polytechnique, France, 2015.

[10] P. Jégou. Contribution à l'étude des problèmes de satisfaction de contraintes : Algorithmes de propagation et de résolution – Propagation de contraintes dans les réseaux dynamiques. PhDthesis, Université des Sciences et Techniques du Languedoc,1991.

[11] B. Nicolas, Modélisation et résolution en programmation par contraintes de problèmes mixtes continu/discret de satisfaction de contraintes et d'optimisation, thèse de doctorale, l'université de nantes.

ملخص

مشكلة الجدول الزمني هي عملية معقدة، فهي من مشاكل التحسين التوافقي التي يصعب حلها، لأن حل هذه المشكلة تكون ممثلة من قبل مجموعة من الخصائص. والهدف هو تحقيق أفضل مزيج من هذه الخصائص.

و الهدف من هذه المذكرة هو تصميم نموذج لمشكلة الجداول الزمنية في قسم الإعلام الآلي في جامعة المسيلة بالاعتماد على مفاهيم البرمجة المقيد، تمت صيغة هذه المشكلة باستعمال مشكلة تحقيق القيود (CSP).

الكلمات المفتاحية:

مشكلة تحقيق القيود (CSP)، مشكلة الجدول الزمني، مشاكل التحسين التوافقي، البرمجة المقيد.

Abstract

The problem of the Timetabling is a complex process, it is a combinatorial optimization problem that is very difficult to solve, because a solution of this problem is represented by a set of properties. The goal is to obtaining the best combination of these properties.

The purpose of this work is to modeling the problem of time tabling of Department of Informatics of the University of M'sila in based on the constraint programming, these problems are formulated by problems of constraint satisfaction (CSP).

Key words:

Timetabling, combinatorial optimization, Constraint programming, problems of satisfaction of constraints.

Résumé

Le problème de l'emploi du temps est un processus complexe, c'est un problème d'optimisation combinatoire très difficile à résoudre, car une solution ce type de problème est représentable par un ensemble de propriétés. Le but est d'obtenir la meilleure combinaison de ces propriétés.

Le but de ce mémoire est de modéliser le problème de l'emploi du temps du département d'informatique de l'université de M'sila en basant sur la programmation par contraintes. Ce problème est formulé par un problème de satisfaction de contraintes (CSP).

Mots clés :

l'optimisation combinatoire, la programmation par contraintes, problèmes de satisfaction de contraintes, l'emploi du temps..