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

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET DE l'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Informatique

Par :

Boumegouas younes
Benmestoura toufik

Sur le thème

Gestion des plannings du personnel de la santé par la programmation par contraintes

---

Soutenu publiquement le 19/09/2022 à Tiaret devant le jury composé de :

Mr. Hattab Noureddine	Grade Université MAA	Président
Mr. Bakkar Khaled	Grade Université MAA	Encadrant
Mme. Hamdani Abdia	Grade Université MAA	Examinateur

2021 - 2022

Dédicace

Avec tout respect et amour je dédie ce modeste travail :

- Aux personnes dont j'ai bien aimé la présence dans ce jours, à mes frères **Mestafa** et **zakaria bouderoui** , je dédie ce travail dont le grand plaisir leurs revient en premier lieu pour leurs conseils, aide et encouragements.

A tous mes proches la famille **benmestoura** et **boumegouas :** Que ce travail soit pour vous un témoignage de mon respect et de mon affectation.

- Aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés et qui m'ont accompagnaient durant mon supérieures, mes aimables aimes, mes collègues d'étude

A mon encadreur **bekare khaled :** pour son aide et sa précieuse attention

Remercîments

En préambule à ce mémoire, j'adresse ces quelques mots pour remercier notre **grand Dieu Allah** tout puissant pour exprimer ma reconnaissance envers sa grand générosités. **Allah** m'a donné la volonté, la patience, la sonte et la confiance durant toutes mes années d'étude. Je tiens à exprimer tout d'ordre ma reconnaissance et mes remerciements les plus profonds à mon encadreur **bakar khaled** qui m'a proposé le sujet de ce travail son aide et ses conseils ont été pour moi un soutien très précieux. Je désire à le remercie pour sa compétence, sa rigueur ainsi que pour leur caractère novateur de ses idées. Mes vifs remerciements les plus sincères aux personnes qui m'apport leur aide et qui contribué à la réussite de cette formidable année universitaire. Enfin, mes derniers profonds gratitudes à **mon chère père** et **ma mère** ainsi qu'à tout ma famille pour leur grand soutien.

ملخص

تقدم أطروحة الماجستير هذه نموذجًا لبرمجة القيد بالإضافة إلى استراتيجيات البحث لصياغة وحل مشاكل الجدولة في المستشفيات. تمت بالفعل دراسة هذه المشكلة على نطاق واسع وتم تطوير العديد من الأساليب المختلفة على مر السنين ، لكنها غالبًا ما تكون مرتبطة جدًا بسياقها التنموي. لذلك، فإن تصميم طريقة قوية يمكن تطبيقها على سياقات مختلفة لا يزال يمثل تحديًا.

نحن نصف طريقة مرنة وعامة إلى حد ما لبرمجة القيد التي تدمج استدلال البحث بطريقة أصلية. نقارن بعض الاستراتيجيات على مجموعة بيانات غير متجانسة. لغرض هذه الأطروحة ، ركزنا على الرضا بدلاً من التحسين ، لكننا نشير إلى التوجيهات التي يجب اتباعها لتحويل طريقتنا إلى طريقة تحسين للجداول الزمنية الممكنة.

الكلمات المفتاحية: الجدولة ، إعداد المستشفى ، برمجة القيد ، استدلال البحث .

Résume

Ce mémoire de maitrise présente un modelé de programmation par contraintes ainsi que des stratégies de recherche permettant de formuler et de résoudre des problèmes de confection d'horaires en milieu hospitalier (infirmier). Ce problème a déjà été largement étudié et beaucoup d'approches différentes ont été développées au cours des années, mais elles sont souvent trop liées à leur contexte de développement. Concevoir une méthode robuste pouvant s'appliquer à différents contextes reste donc un défi à relever.

Nous décrivons une méthode flexible et plutôt générale de programmation par contraintes qui intègre des heuristiques de recherche de façon originale. Nous comparons quelques stratégies sur un ensemble de données hétérogènes. Dans le cadre de ce mémoire, nous nous sommes concentrés sur la satisfaction plutôt que l'optimisation, mais nous indiquons les directions à suivre pour transformer notre méthode en une méthode d'optimisation des horaires réalisables .

Mots Clés : planificaton, Milieu hospitalier(infirmier), programmation par contraintes, heuristiques de recherche

Abstract

This master's thesis presents a constraint programming model as well as research strategies for formulating and solving scheduling problems in hospitals. This problem has already been widely studied and many different approaches have been developed over the years, but they are often too tied to their development context. Designing a robust method that can be applied to different contexts therefore remains a challenge.

We describe a flexible and rather general method of constraint programming which integrates search heuristics in an original way. We compare some strategies on a heterogeneous data set. For the purpose of this thesis, we have focused on satisfaction rather than optimization, but we indicate the directions to follow to transform our method into an optimization method of feasible schedules.

Keywords: scheduling , hospital setting , constraint programming , search heuristics.

Liste des tableaux

Tableau 3.1 Week-ends brisés et non brisés autorisés 59
Tableaux 3.2 de description des taches de l'unité de d'analyse des infermières 60

Listes des figures

Figure 2.1 Automate correspondant à un patron courant pour des horaires de personnel. 29
Figure 2.2 Modélisation des charges de travail individuelles comme un problème de mise en boite. Les boites Grises représentent les quarts de travail et la boite noire déjà placée permet de réduire la charge effective de l'employé C.....32
Figure 2.3 Graphe des valeurs pour le filtrage de distribue (Y, V, X).....38
Figure 2.4 Ordre de parcours des feuilles lors d'une recherche en profondeur. 44
Figure 2.5 Ordre de parcours des feuilles lors d'une recherche à divergence limitée. L'ordre apparait à l'intérieur du cercle alors que le nombre de divergences est indiqué en dessous 45
Figure 3.1 exemple du solver de contrainte optaplanner……50
Figure 3.2 Exemple l'horaire satisfaisant le patron J/M/N sur les périodes grisées. 55
Figure 3.3 Horaire découpé en séquences de types. 56
Figure 3.4 Conséquences des vacances sur l'enchaînement des week-ends.....60
Figure 3.5 Exécution de l'application 63
Figure 3.6 Interface de l'application 64
Figure 3.7 Planning Avon l'exécution 65
Figure 3.8 Planning après l'exécution.....65

Liste des Acronymes

CSP: problème de satisfaction de contraints.

PPC: programmation par contraints.

COP: problème d’optimisation sous contraints.

HIBISCUS : Horaire d'Infirmières Basé Sur les Contraintes

Table des matières

Dédicace I

Remercîment II

ملخص III

Résume IV

Abstract V

Liste des tables..... VI

Liste des figures..... VII

Liste des Acronymes VIII

Table des matières IX

Introduction générale 01

Chapitre 1 : Optimisation et planification

1- INTRODUCTION 03

2- DEFINITION 03

3- PROBLEMES D'OPTIMISATION 03

4- CLASSIFICATION DES PROBLEMES D'OPTIMISATION 05

4-1- Les problèmes d'optimisation continue versus les problèmes d'optimisation discrète s05

4-2- Les problèmes d'optimisation avec et sans contrainte 05

4-3 - Les problèmes d'optimisation mono-objectif ou multi-objectif 05

4-4 - Les problèmes d'optimisation déterministe ou stochastique 06

5-DOMAINES D'APPLICATION ET QUELQUES PROBLEMES TYPES 06

6-PLANNINGE ET OPTIMISATION 07

6-1- la problématique de la planification d'horaires de travail 07

6-2- Qu'est ce que la planification ? 08

6-3- Qu'est ce qu'un planning ? 08

6-4- A Quoi sert un planning ? 09

6-5- Comment est évalué un planning ? 10

6-6- Qui peut se charger de l'élaboration d'un planning ? 10

6-7- Différents types de plannings 10

6-7-1- Types de plannings dans le domaine de la santé 10
7-1- Le problème de confection d'horaires 11
7-2- Les méthodes de confection d'horaires 13
7-2-1- Horaire cyclique avec rotations 14
7-2-2- Horaire cyclique sans rotation 14
7-2-3- Horaire non cyclique 14
8-1- Revue de la littérature 14
8-1-1- Travaux précurseurs 15
8-1-2- Programmation mathématique 16
8-1-3- Méta-heuristiques 17
8-2- Programmation par contraintes 18
8-3- Méthodes hybrides 19
8-4- Autres méthodes 20
9- Conclusion 21

Chapitre 2 : Programmation par contraintes

1-Introduction 23
2- Principe général 24
3- Modalisation 24
4- Problèmes de satisfaction des contraintes 25
4-1- Langage de contraintes satisfiable 26
5- Exemples de modèle 26
6- Quelque contraintes globales 27
7- Conseils de modélisation 30
7-1-Choisir les variables de décision 30
8- Modèles possibles 31
9- Ajouter des contraintes redondantes 31
10- L'ajout de la contrainte 32
11- Propagation 33
11-1-Algorithmes de filtrage 35

11-2- Contraintes binaires 35
11-3-Contraintes arithmétiques 36
11-4-Contraintes globales 37
12- Recherche 40
12-1 -Heuristiques de choix de variable 41
12-2- Heuristiques de choix de valeur 43
12-3- Exploration de l'arbre 43
12-4- APPROCHES HYBRIDES 45
13-Conclusion 47

CHAPITRE 3 : ETUDE DE CAS

1- INTRODUCTION 49
2- OUTILS DE DEVELOPPEMENT 49
2-1- Optaplanner 49
2-2- Eclipse 51
3- Presentations du probleme de plannin 52
3-1- Formulation du problème 52
3-1-1 le personnel 52
3-1-2 les taches 52
3-1-3- L'horizon de planification 53
3-1-4- les règles 53
3-1-4-1- définition et notation 53
3-1-4-2- satisfaction de la demandes 54
3-1-4-3- les disponibilité 54
3-1-4-4- respect de la charges de travaille 55
3-1-4-5- régules ergonomique 55
3-1-4-6- règles d'équité 56
3-1-4-7- distribution des quarts 57
3-2- Particularité des unités de soins infirmiers 57
3-2-1difinition de la demande 57

3-2-2- vacances et week-end 58
3-2-3- Exception pour la règle sur les séquences de week-ends 59
4- Modélisation du problème 61
4-1- Variables de décision, paramètres et domaines 61
4-2-Contraintes 62
4-3-Fonctions Objectif 62
Conclusion générale 67
Références Bibliographique 69

Introduction générale

Introduction générale :

Avec l'évolution de la technologie, les tâches humaines deviennent de plus en plus complexes et leur réalisation nécessitent des techniques d'optimisation afin de les réaliser dans les brefs délais et avec les moindres dépenses. Parmi les domaines qui nécessitent une forte utilisation de l'optimisation nous trouvons le domaine de réalisation des plannings. Ce dernier est omniprésent dans tous les secteurs d'activité et son objectif est d'exploiter le réquisitionnent du personnel avec les meilleurs façons.

L'hôpital est l'une des organisations qui ont besoin de cette satisfaction. La spécificité du travail des infirmier qui exerce plusieurs tâches de différentes natures nous oblige de lui fournir un planning de gestion bien optimisé et bien étudié.

Soit il s'agit d'un logiciel distribué a grande échelle ne permettant pas de tenir compte de toutes les spécificités que l'on peut rencontrer. Il reste donc de nombreux hôpitaux pour lesquels aucun logiciel actuel ne convient parfaitement. Nous proposons une approche qui prend en compte un ensemble de règles très riche, per mettant d'apporter des solutions à ces hôpitaux. Nous utilisons pour cela la programmation par contraintes (PPC) qui est un outil issu conjointement des communautés de l'intelligence artificielle et de la recherche opérationnelle. Il existe déjà plusieurs travaux sur la confection automatisée d'horaire utilisant la PPC. Nous apportons tout de même quelques nouveauté

Structuration du mémoire

Ce mémoire et composé de trois chapitre :

Chapitre 1 : on va parler de l'optimisation et de la planification en générale en suit en va introduire quelque recherche sur les méthodes de convection d'horaire au milieux hospitalier.

Chapitre 2 : en parleras sur la programmation par contrainte c'ais principe et les méthodes de cette approche.

Chapitre 3 : ce chapitre et consacré a notre étude opérationnelle.

Chapitre 1 :

Optimisation et planification

1- Introduction :

Avec l'évolution des techniques humaines dans tous les secteurs de la vie quotidienne, les tâches de conception, de mise en œuvre et de maintenance ont devenus de plus en plus complexes et leur réalisation nécessite un mode de travail optimisé afin de minimiser les coûts, le temps de réalisation et les risques. De ce fait, les techniques d'optimisation prennent leur importance et la discipline d'optimisation est de planification imposée dans la quasi-totalité des activités humaines. Le chapitre suivant dévoile les techniques et les aspects de cette discipline en mettant l'accent sur l'optimisation des plannings.

2- Définition :

Définition 1 : L'optimisation est l'action et l'effet d'optimiser. Ce verbe veut dire chercher la meilleure manière de réaliser une activité. Le terme est souvent employé dans le domaine de mathématiques et de l'informatique [1.].

Définition 2 : L'optimisation est l'acte d'obtenir le meilleur résultat possible dans des circonstances données. En conception, construction, maintenance, ..., les ingénieurs doivent prendre des décisions. Le but de toutes ces décisions est soit de minimiser les efforts, soit de maximiser les avantages. L'effort ou le bénéfice peut généralement être exprimé en fonction de certaines variables de conception. Par conséquent, l'optimisation est le processus consistant à trouver les conditions qui donnent la valeur maximale ou minimale d'une fonction [2].

3- Problèmes d'optimisation :

Dans de nombreux cas, des relations de préférence entre les solutions d'un CSP existent au égard aux critères fixés par le décideur. L'optimisation consiste alors rechercher des solutions optimales d'un CSP au regard des relations de préférence du décideur. Nous nous intéresserons uniquement à des problèmes d'optimisation combinatoire c'est-à-dire lorsque l'ensemble des solutions est discret et qu'il y a un critère d'optimalité unique. Ce critère d'optimalité est généralement associé à la maximisation ou 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 objectif. Cette fonction est souvent modélisée par une variable dont le domaine est défini par les bornes supérieures et les bornes inférieures.

En pratique, la résolution d'un problème d'optimisation mathématique consiste à trouver la meilleure solution à un problème qu'on a su préalablement exprimer sous une forme

mathématique particulière qui fait intervenir un ou plusieurs critères. Ces critères sont exprimés sous la forme d'une fonction mathématique, appelée souvent fonction objectif. La solution optimale ou meilleure solution à trouver une valeur extrême, appelée aussi extremum – c'est à dire un maximum ou un minimum – à cette fonction objectif. L'optimisation se rapproche donc des calculs d'extremum de fonctions.

Dans le cadre de ce chapitre, on définira un problème d'optimisation comme la détermination mathématique d'un ensemble de paramètres (ou variables de contrôle) afin d'optimiser (c'est-à-dire minimiser ou maximiser) une quantité mathématique donnée, appelée fonction objectif (ou critère). Cette fonction objectif peut être sujette à des contraintes.

Un problème d'optimisation peut par exemple consister à chercher le maximum du rendement d'une unité de production industrielle, la distance maximale parcourue par un véhicule pour un quantité de carburant donnée, le nombre maximal de clients servis à un guichet dans un intervalle de temps fixé, la résistance mécanique maximale d'une pièce produite, etc..

Un problème d'optimisation peut également consister à chercher le minimum des coûts de production d'une usine, à minimiser les pertes thermiques d'un procédé industriel, à identifier le temps de fabrication minimal, etc.

Dans tous ces exemples, il est nécessaire d'avoir modélisé (c'est à dire mis en équations) la grandeur qu'on cherche à minimiser ou maximiser. En fait, on retrouve des problèmes d'optimisation dans de nombreuses applications telles que les problèmes de conception, d'optimisation des conditions de fonctionnement de procédés, d'ordonnancement, de planification , de transport, de localisation, mais également dans les domaines économiques et financiers, comme la gestion de portefeuilles.

Face à un problème d'optimisation compliqué, soit en raison du grand nombre d'inconnues et/ou soit en raison de la formulation mathématique complexe, il est rare de pouvoir trouver une solution analytique exacte.

Il est alors nécessaire de résoudre « numériquement » le problème.

La plupart des algorithmes numériques d'optimisation sont itératifs et permettent de s'approcher pas à pas vers la solution. Selon le problème d'optimisation à résoudre, des algorithmes peuvent être efficaces ou au contraire complètement inopérants.

4- Classification des problèmes d'optimisation

Face à la résolution d'un problème d'optimisation, il est important de bien identifier à quelle catégorie ce problème appartient. En effet, les algorithmes développés sont conçus pour résoudre un type de problème donné et sont peu efficaces pour un type différent. La classification des problèmes d'optimisation change d'un auteur à l'autre. Par exemple, on distingue [3]:

4-1- Les problèmes d'optimisation continue versus les problèmes d'optimisation discrètes :

Dans certains cas, les variables de décision sont discrètes, le plus souvent sous la forme d'entiers ou de binaires. Le problème d'optimisation est dit discret. Au contraire, dans les problèmes d'optimisation continue, les variables peuvent prendre n'importe quelle valeur, ce sont des réels. Les problèmes d'optimisation continue sont généralement plus simples à résoudre. Un problème d'optimisation mêlant variable continue et variable discrètes est dit mixte.

4-2- Les problèmes d'optimisation avec et sans contrainte :

Il est important de bien distinguer les problèmes où des contraintes existent sur les variables de décision. Ces contraintes peuvent être simplement des bornes et aller jusqu'à un ensemble d'équations de type égalité et de type inégalité. Il est parfois possible d'éliminer une contrainte égalité par substitution dans la fonction objective. Naturellement, les problèmes avec contraintes sont plus compliqués à résoudre et utilisent des algorithmes dédiés.

4-3- Les problèmes d'optimisation mono-objectif ou multi-objectif

Les problèmes mono-objectifs sont définis par une unique fonction objectif. Les problèmes multi-objectifs existent quand un compromis est à rechercher entre plusieurs objectifs contradictoires. Il est éventuellement possible (mais pas nécessairement efficace) de reformuler un problème multi-objectif avec une seule fonction objectif sous forme d'une combinaison des différents objectifs ou en transformant des objectifs sous forme de contraintes.

4-4 - Les problèmes d'optimisation déterministe ou stochastique :

Les problèmes d'optimisation déterministe considèrent que les données sont connues parfaitement, alors que dans les problèmes d'optimisation stochastique, ce n'est pas le cas ; par exemple une approche stochastique peut être pertinente dans le cas où les variables d'un problème sont les ventes futures d'un produit. Dans ce cas, l'incertitude peut être introduite dans le modèle.

5- Domaines d'application et quelques problèmes types :

L'optimisation est une discipline scientifique qui vise à obtenir le meilleur résultat dans certaines circonstances. Beaucoup d'applications difficiles dans les affaires, l'économie et l'ingénierie peuvent être modélisées comme des problèmes d'optimisation. En fait, la résolution de problèmes dans la vie réelle est souvent complexe et difficile à résoudre, et l'optimisation joue un rôle clé dans la recherche de solutions réalisables à ces problèmes [4]. Au niveau général, l'optimisation peut se réaliser dans des différents domaines, toujours avec le même objectif : améliorer le fonctionnement de quelque chose au moyen d'une gestion perfectionnée des ressources. L'optimisation peut avoir lieu à n'importe quelle étape quoiqu'il est conseillé de la mener à bien jusqu'à la fin du processus visé. Plusieurs domaines vise à utiliser l'optimisation chacun à sa façon.

Parmi ces domaines nous citons :

Mathématiques : dans le cadre des mathématiques, l'optimisation cherche à apporter des réponses à un type général de problèmes qui consistent à sélectionner le meilleur élément parmi plusieurs appartenant au même ensemble.

Software : L'optimisation des logiciels a pour but d'adapter les programmes informatiques pour qu'ils accomplissent leurs taches, le plus rapidement possible. Grace à l'optimisation des logiciels, par exemple, les programmes n'ont pas besoin d'autant de mémoire pour fonctionner car ils utilisent leurs ressources d'une façon plus efficace.

L'optimisation des requêtes, par ailleurs, consiste à améliorer les temps de réponse dans la gestion des bases de données. Dans le langage des requêtes SQL, qui est l'un des plus utilisés par les développeurs, l'optimisation vise à simplifier les requêtes les plus complexes pouvant être plus longues à résoudre.

Planification : Une personne qui souhaite optimiser son temps de travail, par exemple, peut changer l'organisation de ses activités, faire appel à la technologie ou bien travailler avec quelqu'un qui puisse l'aider. Si l'optimisation est plutôt réussie, la personne pourra travailler ou produire davantage en moins de temps.

Plusieurs domaines d'application peuvent être cités tels que :

• calcul d'une trajectoire de rentrée dans l'atmosphère d'une navette.

• Gérer un planning d'un hôpital

• voyageur de commerce…etc. [5].

6- le planning et l'optimisation (on prend l'exemple de notre mémoire la gestion du personnel infirmier) :

Un planning est un calendrier de travail, où figure à la fois le temps, l'affectation du personnel par un horizon (un intervalle du temps). La gestion des planning est une tâche très complexe, cette complexité vient du fait qu'on a un ensemble de contraintes à respecter, et des conflits à résoudre. Le problème de planning 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. Pour cela nous définissons les différents types de plannings dans différents domaines de travail et plus particulièrement dans le domaine pédagogique.

6-1- 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 publique 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 [6].

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

6-2- Qu'est ce que la planification ?

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 [6].

6-3- 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, et les congés et repos.

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..

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 :

- Le 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.

6-4- 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 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 au court, moyen et à long terme.

6-5- 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 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 des solutions alternatives. L'aspect combinatoire (pour l'élaboration des plannings) rend d'autant plus difficile la négociation, car les opinions sont plus difficiles a formuler .

Les moyens informatiques apportent une aide certaine notamment dans l'acquisition et la confrontation des données individuelles.

6-6- Qui peut se charger de l'élaboration d'un planning ?

Dans la plupart des entreprises, cette tâche peut être centralisée ou déléguée a 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 du planning joue un rôle primordial.

6-7- Différents types de plannings

Dans la construction de plannings d'horaires de travail, si créer un planning optimisé d'une journée est pas assez, 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.

6-7-1- 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

7-1 Le problème de confection d'horaires :

La gestion du personnel hospitalier est une recherche perpétuelle d'un équilibre entre trois objectifs contradictoires: la qualité des soins prodigués, les conditions de travail du personnel et les coûts financiers. D'après Warner [9] , trois niveaux de décision interviennent dans le processus de gestion du personnel infirmier :

i. Le niveau stratégique ("The staffing décision"). C'est l'étape du dimensionnement des unités de soins en réponse à l'évaluation de la demande. Au cours de cette étape, on détermine pour chaque catégorie du personnel les effectifs nécessaires pour pouvoir couvrir la demande.

ii. Le niveau tactique ("The scheduling décision") . C'est l'étape de la confection d'horaires à proprement parler, au cours de laquelle on détermine la nature, le nombre et les dates des tâches affectées a chaque membre du personnel.

L'horizon de planification peut s'étaler sur une période allant d'une semaine a plusieurs mois.

iii. Le niveau opérationnel ("The allocation décision") . Ce sont des décisions prises au jour le jour pour réagir aux événements imprévus tels que les absences ou fort hausse de la demande de cette étude, nous nous intéressons seulement au second niveau de décision, ce qui simplifie beaucoup le problème car le nombre d'unités de soins, leurs effectifs ainsi que la demande ont été estimés au premier niveau de décision. Les coûts financiers étant principalement déterminés lors du dimensionnement des unités de soins , il ne reste vraiment que deux objectifs pour l'étape de confection des horaires : la qualité des soins et les conditions de travail du personnel.

Plus concrètement, la confection d'horaires est un problème d'affectation d'un ensemble de ressources (le personnel) à un ensemble de tâches (appelés aussi quarts) sujet à certaines contraintes (règles). Concevoir un horaire, c'est donc trouver une affectation des ressources aux différentes tâches qui satisfait les contraintes. Souvent, il en existe beaucoup, et on s'intéresse alors a trouver un ensemble de bons horaires relativement à certains critères.

L'une des particularités du milieu hospitalier est que l'on rencontre souvent des problème sur-contraint (à cause de l'hétérogénéité des règles, surtout celles concernant les préférences individuelles) . Il faut alors résoudre un problème supplémentaire :

- Le choix des contraintes que l'on va se permettre de violer afin d'obtenir une solution acceptable. La plupart des approches définissent un ensemble de contraintes dures, qu'il est impératif de respectes, et un ensemble de contraintes souples, que l'on se permet de violet.

Dans la littérature, chaque problème étudié se base sur un contexte particulier. Il en résulte une multitude de modèles, chacun ayant ses propres ensembles de contraintes dures et souples ainsi que son propre objectif. Il y a plusieurs raisons à cela la législation propre à chaque état, les habitudes propres à chaque hôpital, voire chaque département ou unité de soins, ainsi que des conditions socio-économiques particulières comme la forte pénurie de personnel que subissent actuellement les hôpitaux d'Algérie. Malgré cela, on trouve beaucoup de similarités dans tous ces modèles. Quelques travaux récents , dont celui présenté dans ce mémoire, cherchent à spécifier des modèles généraux pouvant s'appliquer à plusieurs contextes en s'appuyant sur une nomenclature précise des règles et des objectifs. Après examen des travaux de Carter et Lapierre [28] , de Forget [11] et de nombreux autres , voici la nomenclature qui a été retenue :

- respect de la demande,

- respecter les charges de travail individuelles.

- disponibilités.

-règles ergonomiques.

-règles d'équité.

La liste des objectifs rencontrés est assez longue. Voici les principaux (dans la majorité des cas, plusieurs sont pris en compte simultanément)

-minimiser l'effectif de personnel .

-minimiser les couts économique .

-minimiser les période de sur effectif et de sous-effectif .

-minimiser les écarts entre la charge de travail voulue et la charger de travail réelle .

-équilibrer les quarts d'un certain type de façon équitable.

-maximiser la satisfaction des préférence individuelles.

7-2 :Les méthodes de confection d'horaires :

Warner [9] donne une liste de critères permettant de comparer des méthodes de confection d'horaires :

1- couverture : mesure de la qualité de la couverture des taches. Cett indice est di- récemment relié à la qualité des soins prodigués. Si les horaires générés ne couvrent pas toutes les taches, l'hôpital est obligé de faire appel à des personnel supplémentaire afin d'assurer un service minimum.

2- Qualité : mesure de la qualité des horaires individuels générés des conditions de travail du personnel. Elle doit tenir compte de la législation en vigueur, de certaines règles propres à l'hôpital ou au service, et des préférences et aversions du personnel.

3- Stabilité : mesure du degré avec lequel le personnel peut connaitre ses affectations futures, en particulier les jours de congé et les vacances, et du degré de confiance dans la méthode. Plus la méthode est stable, plus le personnel est capable de prévoir son horaire avec suffisamment de précision pour pouvoir organiser son temps.

4- Flexibilité : mesure de la capacité de la méthode à faire face aux changements d'une période à l'autre (choix des vacances, préférences individuelles, passage de temps complet à temps partiel, changement de personnel.

5- Equité : mesure du degré de confiance dans l'équité de la répartition des tâches par le système.

6- Coût : mesure de l'ensemble des ressources mises en œuvre pour la conception de l'horaire.

Les méthodes de confections d'horaires utilisées dans les hôpitaux diffèrent suivant le type d'horaire génère. Il existe trois grands types d'horaires : les horaires cycliques avec rotations, les horaires cycliques sans rotation et les horaires non cycliques.

7-2-1 Horaire cyclique avec rotations :

Un horaire cyclique avec rotation est construit à partir de patrons d'horaires sur une période courte. que les membres du personnel suivent à tour de rôle. Ainsi, après une rotation complète du personnel, chacun des membres a fait exactement les mêmes Quarts (tache) et dans le même ordre, ce qui garantit une équité parfaite. Cette méthode est en outre très stable, car l'horaire peut se répéter sur une longue durée. Cependant, elle n'est pas du tout flexible car elle considère que le personnel est homogène et ne permet pas de prendre en compte des préférences individuelles (disponibilités, vacances, règles ergonomiques, etc.).

7-2-2 Horaire cyclique sans rotation :

Un horaire cyclique sans rotation est également construit à partir de patrons d'horaires sur une courte période mais, cette fois, chaque membre du personnel répète son propre horaire. L'équité peut être conservée à condition que les patrons d'horaires soient bien construits avec une bonne distribution de tous les types de quart. Le grand avantage par rapport à la méthode précédente est la possibilité de personnaliser les horaires, ce qui donne déjà un peu plus de flexibilité à la méthode. Mais le fait que le même horaire soit répété en boucle ne permet pas de prendre en compte de la préférence épisodique comme des périodes de vacances ou des indisponibilités occasionnelles.

7-2-3 Horaire non cyclique :

Les horaires non cycliques sont reconstruits au début de chaque période de planification. C'est la méthode la plus flexible car elle permet de prendre en compte beaucoup de préférences individuelles. L'équité est un peu plus difficile à respecter, mais peut néanmoins l'être si certaines règles de distribution sont respectées. De plus, si le personnel est hétérogène (des personne plus anciennes, plus expérimentées, etc.), l'équité est souvent difficile à définir.

8-1 Revue de la littérature :

Nous nous intéressons uniquement aux méthodes fournissant des horaires non cycliques permettant de prendre en compte les requêtes individuelles du personnel. Dans ce contexte particulier, essayer de déterminer l'effectif minimum du personnel pouvant couvrir la demande est trop complexe. Les travaux de Burne et Carter [31] ne s'appliquent qu'aux horaires cycliques et au prix d'hypothèses bien trop restrictives pour le projet actuel. Les

premiers travaux datent des années 70 avec Warner [9] Parmi les approches récentes. on trouve des méthodes utilisant la programmation mathématique des méta-heuristiques et la programmation par contraintes ainsi que quelques approches hybrides D'autres méthodes, moins utilisées.

8-1-1 Travaux précurseurs :

Pour Warner [9], les solutions réalisables sont celles qui respectent la demande et les charges de travail. Un certain nombre de règles souples servent à mesurer la qualité des horaires individuels. Le personnel doit déterminer individuellement des poids pour chacune de ces règles, il s'agit Principalement de préférences ou d'aversions pour les quarts et les congés isolés, les séquences de travail de plus de six jours et pour certains enchainements de quarts. L'équité se traduit par une règlementation stricte du nombre de points que chaque membre du personnel peut affecter à l'ensemble des critères avant la planification.

La résolution est heuristique et comporte deux phases : une phase d'initialisation pour trouver une solution réalisable et une phase ressemblant à une recherche locale au cours de laquelle l'objectif est amélioré itérativement. La méthode obtient des résultats pour des unités de soins allant jusqu'à 47 infirmières sur des horizons de planification de 42 jours (découpes périodes de 14 jours) Il faut noter que la politique d'affectation des week-ends est prédéterminée, ce qui réduit significativement la complexité du problème et permet aussi de découper l'horizon de planification en sous-périodes de 14 jours sans que le couplage entre les sous-périodes ne soit trop fort.

Au même moment, Miller et al [32]. s'est intéressé au problème de l'affectation des jours de repos (dans lequel on cherche à dissocier les périodes de repos et les périodes de travail sans s'intéresser la leur nature). L'approche proposée ici est à la base des méthodes plus récentes de programmation mathématique généralisée basées sur la technique de génération de colonnes. Les variables ne sont pas des quarts, mais des horaires individuels au complet. Pour chaque membre du personnel, on énumère l'ensemble des horaires réalisables respectant la charge de travail, les disponibilités et quelques contraintes ergonomiques dures un poids est associé à chacun des horaires en fonction de diverses règles.

L'algorithme de résolution est une heuristique en deux phases : une phase d'initialisation et une phase d'amélioration itérative d'une somme pondérée du poids global de l'horaire et de pénalités traduisant la non-satisfaction de la demande (périodes de sur- et sous-

effectif). La méthode a obtenu des résultats sur des problèmes composés d'une douzaine d'infirmières sur des horizons de 28 jours. Le point faible de cette méthode est certainement l'énumération initiale des horaires individuels ainsi de leur stockage explicite qui prend beaucoup de place. De plus, même si le système des poids peut sembler très flexible, il est néanmoins fastidieux en pratique car les horaires individuels sont nombreux.

8-1-2 Programmation mathématique :

Forget[11] et Beaulieu[12] proposent un modèle PLNE (Programmation Linéaire en Nombres Entiers) prenant en compte des contraintes ergonomiques complexes Les contraintes souples sont traduites par des pénalités dont la somme pondérée constitue un objectif a minimiser. Malgré cela, leurs problèmes sont sur-contraints dans la plupart des cas. Ils proposent donc une procédure automatique qui relaxe successivement certaines contraintes dures.

L algorithme de résolution utilise SIMPLEX. Cette approche fournit des horaires

pour des salles d'urgence compostes d'environ 20 infermière sur des horizons de planification d'un mois.

Jaumard et al [13] proposent un modèle général et flexible de Programmation mathématique généralisée utilisant la technique de génération de colonnes. La décomposition utilisée est semblable à celle de Miller et al. mais, dans ce cas, les horaires individuels sont génères au cours de la recherche. Le problème maitre est un PLNE qui cherche la configuration optimale des horaires individuels déjà générés relativement à un objectif tenant compte de critères tels que la satisfaction de la demande, la qualité et l'équité de l'horaire. À chaque itération un problème auxiliaire génère un ou plusieurs horaires individuels (colonnes dans la matrice des contraintes du problème maître) susceptibles d’améliorer la solution globale. L‘algorithme de résolution du problème maître utilise CPLEX tandis que le problème auxiliaire utilise un algorithme de programmation dynamique combiné à un brunch-and-bound.

Cette méthode fournit des résultats pour des unités de soins d'environ 50 infermières sur des horizons de planification de six semaines.

Caprara, Monaci [14] et Toth Proposent plusieurs méthodes de programmation mathématique dont une basée sur la génération de colonnes mais dans un contexte bien plus

restrictif que le précédent. D'une part, le personnel est indifférencié et, d'autre part, la modélisation choisie interdit tout enchaînement de quarts différents (les blocs de travail étant nécessairement homogènes) . De plus la charge de travail du personnel est exprimée en nombre de blocs de chaque type, ce qui est assez peu conventionnel.

La situation étant considérablement simplifié, la méthode fournit des horaires pour des exemplaires de problème impliquant plus de mille personnes sur des horizons allant jusqu'à 182 jours.

8-1-3 Méta-heuristiques :

Une méta-heuristique est un schéma algorithmique de haut niveau pouvant se spécialiser pour résoudre des problèmes d'optimisation de façon approchée. Certaines s'inspirent de concepts scientifiques provenant de domaines aussi divers que la biologie (colonies de fourmies), la génétique (algorithmes génétiques) ou la thermodynamique (recuit.simulé).

Thompson [15] propose une méthode de recuit simulé pour résoudre un modèle basé sur celui de Dantzig [16] . La charge de travail et les disponibilités sont des contraintes dures. Les autres règles sont souples et sont exprimées par un ensemble de poids associés aux affectations potentielles. L'objectif est composé de pénalités de sur- et sous-effectif ainsi que de la somme des poids des affectations. Le principal défaut de cette approche réside dans la difficulté de déterminer les poids des affectations. Il est en outre impossible avec ce modèle d'exprimer des règles ergonomiques complexes.

Dowland [17] s'intéresse au problème d'affectation des jours de repos et utilisé à l'instar de Miller [32] un modèle dans lequel les inconnues sont des horaires individuels réalisables. Les horaires individuels sont énumérés et évalués dans une phase d'initialisation. L'algorithme utilisé est une heuristique de recherche tabou particulière faisant intervenir deux phases en alternance : l'une visant à rendre la solution réalisable en terme de satisfaction de demande, l'autre visant à minimiser le poids global de l'horaire.

Comme la méthode de Miller, la modélisation utilisée est assez flexible car elle permet de prendre en compte n'importe quelle règle ergonomique. Cependant l'énumérations des horaires coûte cher et limite la taille des horizons de planification à une semaine ou deux au plus. De ce fait, il est plus difficile de prendre en compte des contraintes d'équilibrage à long terme.

Buzòn [18] propose également une heuristique de type recherche tabou utilisant diverses structures de voisinages. Le modèle permet de prendre en compte des contraintes ergonomiques complexes. Cette approche fournit des résultats pour des salles d'urgence d'environ 20 médecins sur des horizons de planification de trois mois.

8-2- Programmation par contraintes :

Beaucoup de travaux récents se sont tournés vers la programmation par contraintes (PPC). Dans sa thèse, Heuse [19] modélise le problème sous forme d'un CSP (Problème de Satisfaction de Contraintes). Il discute de la façon de briser certaines symétries dans le modèle et propose deux stratégies de recherche, l'une d'elle utilisant la décomposition affectation des jours de repos affectation des tranches horaires. Ces travaux ont aidé à élaborer le logiciel commercial Gymnaste.

-La méthode conçoit assez Facilement des horaires pour environ 30 personnes sur des horizons de deux semaines. Le défaut de cette approche est de ne pas modéliser des règles souples. Ce n'est pas gênant dans le contexte fourni en exemple car les problèmes sont tous cohérents.

- Abdennadher ET Schenker [22] proposent un modèle sous forme de PCSP (Problème de satisfaction de contraintes partiel) et distinguent un ensemble de contraintes dures et un ensemble de contraintes souples. La stratégie est similaire a celle de Heus [19] et comporte trois phases: l'affectation des jours de repos l'affectation des quarts de nuits puis l'affectation des quarts de jour. La minimisation se fait sur le maximum des coûts des horaires individuels, ce qui permet d'éviter les solutions avec une trop grande variance Les auteurs mettent l'accent sur l'interactivité de leur approche qui permet de voir rapidement l'impact des changements effectués par l'utilisateur sur le modèle. Les horaires sont générés pour 20 infirmières sur des horizons de la planification d'un mois.

Les méthodes suivantes modélisent le problème sous forme de CSP ou de PCSP, comme dans les approches précédentes, mais utilisent en outre les contraintes souples sous forme d'heuristiques pour guider la recherche. Darmoni et al [20]. utilisent des compteurs mis à jour dynamiquement au cours de la recherche, Caseau et al [21]. se basent sur un calcul d'entropie pour chaque paire quart/jour et enfin Meisels et al utilisent des règles d'affectation. Les approches de Darmoni et Caseau fournissent des horaires pour des effectifs d'une dizaine

de personnes et pour des horizons d'une semaine, celle de Meisels pour des effectifs de 20 personnes et des horizons de deux semaines.

Cheng et Lee [23] utilisent une modélisation double pour améliorer le filtrage. Dans leur approche, les contraintes souples sont traitées comme des disjonctions dans l'arbre de recherche un point de choix binaire par contrainte souple est ajouté à la racine de l'arbre, la contrainte devant être respectée uniquement dans la première branche. Le premier défaut de cette approche est le bais introduit par l'ordre de ces points de choix dans l'arbre, car les contraintes correspondant à des points de choix élevés seront rarement remises en cause. Pour régler le problème de l'équité, les auteurs proposent un ordonnancement faisant intervenir des paramètres aléatoires pour les contraintes souples exprimant les souhaits individuels. L'explosion combinatoire de l'arbre de recherche lorsqu'on ajoute des contraintes souples est un entre défaut majeur de l'approche. L'approche est validée sur un exemple comprenant des règles assez complexes et surtout un grand nombre de types de quarts différents. Elle fournit des horaires pour des effectifs allant jusqu'à 27 infirmières sur des horizons d'une semaine avec 11 quarts différents.

. Saadie [24] propose un modèle CSP pour résoudre les problèmes des salles d'urgence. L'heuristique de recherche utilisée est semblable à celle de Heus [19] et Abdennadher [22] Enfin, Feuilletin [26] ont modélisé le même problème que Jaumard et Al sous forme de CSP. Leur travail a servi de point de départ a cette maitrise.

8-3-Méthodes hybrides :

Plus récemment, des approches hybrides entre la PC et diverses heuristiques de recherche locale ont été développées, la PC étant utilisée pour parcourir implicitement des voisinages de grande taille.

Rousseau, Pesant et Gendreau [33] présentent un algorithme génétique dont les populations à chaque itération sont constituées de solutions partielles qui sont ensuite complétées par le branch-and-bound de la PPC. Pour modéliser les contraintes complexes, les auteurs utilisent deux contraintes géniériques : la contrainte DISTRIBUTE et une nouvelle contrainte appelée « PATTERN constraint » modélisée avec prollog OPL Studio. Les expérimentations ont été réalisées sur les mêmes problèmes que ceux de Forget [11] et montrent la supériorité de la méthode hybride sur une méthode de PC pure et une méthode de recherche locale pure.

Meyer [10] propose un modèle sous forme de HCOP (Hierarchical Constraint Optimisation Problem) . En plus d'associer un poids à chacune des contraintes, un modèle HCOP définit une hiérarchie parmi les contraintes. Le but est de trouver une affectation qui satisfait toutes les contraintes dures et qui minimise le poids des violations des contraintes de niveau inférieur, en tenant compte de la hiérarchie. Voici la hiérarchie des contraintes proposées par les auteurs.

1 : vacances et temps de repos minimal entre deux quarts (11h dans le contexte étudie) ,

2 : respect de la demande minimale,

3 : contraintes sur le nombre de jours de repos,

4 : respect de la demande,

5 : charges de travail individuelles spécifiées dans les contrats,

6 : préférences parmi un ensemble de patrons d'horaires de 15 jours, 7 : préférences et aversions individuelles.

La méthode proposée est une recherche locale utilisant la PPC pour parcourir des voisinages de grande taille. La plupart des contraintes étant souples, l'auteur propose en outre une méthode de propagation spéciale basée sur les ensembles flous.

8-4- Autres méthodes :

D'autres méthodes moins répandues sont également mentionnées dans la littérature. On peut trouver quelques références dans les travaux de Darmoni [20]et Heus [19]

Il s'agit de méthodes basées sur des systèmes experts ou des réseaux neuronaux.

9-Conclusion :

Dans ce chapitre nous Avons cité la planification à quoi sert un planning et quelque notion sure la planification d'horaire de travail et quelque type de plannings dans le domaine de la santé et on suit on a parlé de l'optimisation et les classifications des problèmes d'optimisation et ces domaines et on suite au a parlé sure quelque travaux précurseurs les méthodes de confection d'horaires et sure la programmation mathématique et les méta-heuristique et la programmation par contraintes et les méthodes hybrides .

CHAPITRE 2

LA PROGRAMMATION PAR CONTRAINTES (PPC)

1-Introduction :

La programmation par contraintes (PPC) est un paradigme de programmation qui permet de dissocier la représentation du problème de sa résolution. En PPC, on ne décrit pas l'algorithme qui permet de trouver une solution mais plutôt les solutions elles-mêmes, en exprimant les propriétés et les relations entre les variables du problème . Une fois le problème défini, on utilise un algorithme appelé solveur de contraintes pour le résoudre.

Un solveur de contraintes est spécifique à un ensemble de contraintes définies sur un domaine particulier que l'on appelle système de contraintes . Il existe des solveurs pour des systèmes de contraintes aussi divers que les contraintes sur les arbres finis, les contraintes linéaires de type égalité sur les réels, les contraintes linéaires sur les réels, les contraintes non-linéaires sur les réels, les contraintes sur les variables de domaine fini, etc... Pour chacun d'entre eux, le solveur de contraintes utilise des algorithmes différents. Le système de contraintes sur les variables de domaines finis est celui qui donne lieu au plus grand nombre d'applications pratiques. C'est d'ailleurs celui que nous utilisons pour modéliser notre problème de confection d'horaires. Les problèmes exprimés sur les domaines finis s'appellent aussi CSP (Contraint Satisfaction problême). Ce sont des problèmes combinatoires souvent NP difficiles .

Le paradigme de programmation par contraintes est orthogonal aux autres paradigmes de programmation que sont la programmation impérative , la programmation fonctionnelle la, programmation logique ou la programmation orientée objet : il existe des langages de programmation par contraintes logiques (comme chip, eclipse. Prolog ou Gnu-Prolog) , fonctionnels (comme oz), impératifs (comme optaplanner ou programme orientés objet (comme les bibliothèques C++ et Java de llog Solver).

La programmation par contraintes (PPC) est, au sens large, l'étude des systèmes de calcul basés sur les contraintes. Ses origines mixtes se retrouvent en interfaces personne-machine dès les années soixante, en intelligence artificielle dans les années soixante-dix et en langages de programmation dans les années quatre-vingt. Son application sérieuse et répandue au domaine de la recherche opérationnelle est plus récente ; elle date des années quatre-vingt-dix. Dans ce chapitre, nous traitons une partie seulement de la programmation par contraintes, celle qui s'intéresse àla résolution de problèmes combinatoires. Nous présentons les concepts centraux regroupés selon les trois thèmes suivants : modélisation du problème . propagation

des contraintes et recherche de solutions Le lecteur souhaitant une introduction plus générale à la PPC pourra consulter quelques ouvrages (Apt, [1] ; Dechter, [2] ; Marriot et Stuckey [3]).

2-Principe General :

L'approche de la programmation par contraintes pour résoudre des problèmes combinatoires consiste à les modéliser par un ensemble de variables prenant leur valeur dans un ensemble fini et liées par un ensemble de contraintes mathématiques ou symboliques L'efficacité de ce paradigme repose sur de puissants algorithmes de propagation de contraintes qui éliminent du domaine des variables les valeurs qui engendrent des solutions irréalisables. Si la propagation de contraintes ne suffit pas à elle seule à établir une solution réalisable, une recherche arborescente est entreprise afin de réduire davantage le domaine des variables définissant le problème. A chaque nœud de l'arbre de recherche une variable est d'abord choisie selon une certaine politique (statique ou dynamique) puis fixée à une des valeurs possibles de son domaine. On a une solution lorsque le domaine de chaque variable ne contient qu'une seule valeur et une impasse lorsque le domaine d'une variable devient vide. Il est généralement possible de guider la recherche de solutions réalisables en incorporant de l'information sur la nature du problème dans les politiques de sélection de variables et de valeurs.

3-Modelisation :

3-1- Variables, domaines, contraintes

La résolution de problèmes combinatoires en PPC passe par une modélisation basée sur des variables à domaine fini. à chaque variable x est associé un ensemble fini de valeurs D_x, appelé son domaine, au sein duquel elle prend nécessairement sa valeur. Ce domaine est donné tantôt en extension :

$x \in \{1, 2, 4, 6, 7, 11\}$

Tantôt en compréhension :

$1 \leq x \leq 5$

Un aspect très important de ces domaines est qu'ils ne sont pas simplement statiques, mais plutôt misa jour au fur et à mesure que des décisions sont prises.

Par exemple, $D_x = \{1, 2, 4, 6, 7, 11\}$ deviendra $D_x = \{1, 2, 4\}$ s'il appert que $x \leq 5$ dans le sous-problème considéré. Chaque domaine est donc maintenu dans une structure de données. Un domaine ne peut qu'être réduit : aucune valeur n'y est jamais ajoutée.

Une contrainte impose des restrictions sur les combinaisons de valeurs que peuvent prendre les variables. La portée d'une contrainte est le sous-ensemble des variables sur lesquelles elle est définie. Par exemple, la contrainte c :

$(x_2 - 2x_5 + 3x_8 < 4)$ → équation a pour portée :

$\{x_2, x_5, x_8\}$

Une contrainte est définie comme un sous-ensemble strict du produit cartésien des domaines des variables de sa portée, restreignant ainsi les combinaisons de valeurs possibles. Dans l'exemple précédent, supposons que les domaines initiaux des variables sont :

$D_{x2} = D_{x5} = D_{x8} = \{1, 2\}$

La contrainte ce peut alors être définie par :

$\{(1, 1, 1), (1, 2, 1), (1, 2, 2), (2, 1, 1), (2, 2, 1)\} \subset D_{x2} \times D_{x5} \times D_{x8}$

Pour une permutation lexicographique naturelle (x2, x5, x8) de sa portée. On écrit $(1, 1, 1) \in c$ puisque cette combinaison de valeurs satisfait la contrainte, mais $(2, 2, 2) \notin c$. Il s'agit ici d'une définition en extension ; une définition en compréhension, comme équation $(x2 - 2x5 + 3x8 < 4)$. est tout aussi acceptable et même souvent préférée.

4-Problème de satisfaction de contraintes

On appelle instanciation d'un ensemble de variables X une application $(I : x \in X \rightarrow v \in D_x)$. On dit que I satisfait une contrainte c si la restriction de I a la portée de c donne un n-uplet appartenant à c. Les problèmes combinatoires que la PPC s'efforce de résoudre sont généralement définis comme problèmes de satisfaction de contraintes, dont voici un exemple.

Etant donne :

- un ensemble fini de variables, $X=\{x_1, \ldots , x_n\}$,
- un domaine fini de valeurs possibles pour chaque variable

$D = \{D_{x1}, \ldots, D_{xn}\}, x_i \in D_{xi}\ (1 \leq i \leq n)$,

- un ensemble fini de contraintes sur les variables, C= {c_1 , ... , c_m},

trouver une instanciation de X satisfaisant simultanément toutes les contraintes dans C. S'il n'existe pas une telle instanciation, on dit que le problème n'est pas satisfiable.

4-1- Langage de contraintes satisfiable.

La définition d'une contrainte donnée à l'article 2.1 laissait déjà entrevoir sa richesse expressive. La méthode de résolution en PPC, que nous décrivons en détail a la section 1.3, n'exige pas une forme aussi rigide que 1'a1gorithme du simplexe, par exemple, pour les contraintes exprimées dans un modèle. Il suffit qu'on puisse décrire les combinaisons de valeurs permises par la contrainte.

Il demeure quand même qu'une formulation en compréhension des contraintes est privilégiée. Parmi ces contraintes, les contraintes arithmétiques peuvent être construites à partir des opérateurs relationnels habituels (=, <, >, ≤ , ≥), mais aussi à partir de l'opérateur de différence (≠), qui est fréquemment sollicité.

Ces contraintes sont linéaires ou encore non linéaires. Des opérateurs logiques

(∧ , ∨ , →, ↔) sont également souvent utilises.

A' ces différents types de contraintes s'ajoutent d'autres contraintes, dites

globales, exprimant des sous-structures courantes de problèmes combinatoires.

5-Exemple de modèle

Afin d'illustrer la façon de modéliser en PPC, prenons un problème simplifié d'emploi du temps, celui d'assurer trois quarts de travail différents (notés α, β, γ) par jour sur un horizon d'une semaine (soit sept jours ordonnés : lu, ma, . . . , di) à l'aide de quatre employés (A, B, C, D) travaillant au plus un quart par jour. Voici un modèle possible en PPC :

$$e_{ij} \neq e_{kj} \qquad lu \leq j \leq di, \qquad \alpha \leq i < k \leq \gamma \qquad (1.1)$$

$$e_{ij} \in \{A, B, C, D\} \qquad lu \leq j \leq di, \qquad \alpha \leq i \leq \gamma \qquad (1.2)$$

Selon la contrainte (1.2) chaque variable e_{ij} prend pour valeur l'employé qui assurera le quart i au jour j. En comparaison, les formulations classiques de programmation en nombres entiers auraient défini une variable de décision binaire pour chaque paire employé-quart ou employé-trame . La contrainte (1.1) évite que deux quarts de travail d'un même jour

soient assurés par le même employé. Restreignons davantage le problème en exigeant que chaque employé travaille au moins cinq quarts pendant la semaine :

$t_{jk} = 1 \leftrightarrow e_{ij} = k \quad lu \leq j \leq di, \quad A \leq k \leq D$ (1.3)

$t_{jk} \geq 5 \quad A \leq k \leq D$ (.4) (1.4)

$t_{jk} \in \{0, 1\} \quad lu \leq j \leq di, \quad A \leq k \leq D$ (1.5)

Pour ce faire, on définit les variables 0-1 t_{jk} de la contrainte (1.5) [1] et on les lie au fait que l'employé k travaille le jour j selon la contrainte (1.3). La contrainte (1.4) assure que chaque employé travaille suffisamment.

Ce problème demeure encore très simple à résoudre. Son intérêt réside dans sa capacité d'illustrer plusieurs des notions importantes en PPC. Nous le reprendrons tout au long de ce chapitre et ajouterons à sa complexité. Remarquons déjà la forme particulière du modèle : des variables discrètes prenant une valeur parmi un grand nombre de possibilités (1.2), des contraintes de différence (1.1) et des opérateurs logiques qu'on exprime directement (1.3).

6-**Quelques contraintes globales**

Cette section présente quelques contraintes globales parmi les plus usuelles et notamment celles que le lecteur trouvera dans les autres chapitres de ce mémoire.

La contrainte d'exclusion mutuelle pour un ensemble de variables X, généralement notée all différent(X), impose que les valeurs prises par chacune des variables de l'ensemble soient toutes distinctes.

Par exemple, on pourrait remplacer la contrainte pour le problème d'emploi du temps par :

all différent ((e_{ij}) $\alpha \leq i \leq \gamma$) $lu \leq j \leq di$ (1.6)

La contrainte de distribution des valeurs pour un ensemble de variables X, également appelée contrainte de cardinalité généralisée (gcc) et souvent notée

distribue (Y, V, X), lie la suite de variables $Y = (y_1, y_2, \ldots, y_m)$ au nombre de répétitions de chacune des valeurs de la suite $V = (v_1, v_2, \ldots, v_m)$ prises par les variables de X. Plus précisément, la variable yi a pour valeur le nombre de fois qu'une variable de X prend la

valeur v_i. La contrainte d'exclusion mutuelle all différent correspond en somme au cas particulier ou` $y_i \leq 1$ $(1 \leq i \leq m)$.

Cette nouvelle contrainte permet d'exprimer plus directement, et surtout plus globalement que par les contraintes (1.3)–(1.5), la restriction de l'exemple sur le nombre de jours travaillés par chaque employé :

Distribue $((t_A, t_B, t_C, t_D), (A, B, C, D), (e_{ij}))$ $lu \leq j \leq di$, $\alpha \leq i \leq \gamma$) (1.7)

$t_k \in \{5, 6, 7\}$ $A \leq k \leq D$ (1.8)

On remarque qu'ici, t_k ne peut pas prendre la valeur 7 puisqu'on a 21 quarts assurés par 4 employés en prenant chacun au moins 5, pour un minimum de 20.

L'article 1.3 traite des algorithmes de filtrage associés à la contrainte distribue qui sauraient retirer cette valeur.

La contrainte de relation fonctionnelle entre deux variables, qu'on note

élément(x, V, y), fait correspondre la variable x à la valeur indexée par la variable y dans la suite $V = (v_1, v_2, \ldots, v_m)$. Plus précisément, la variable x a pour valeur le y^e élément de V , v_y.

Le problème d'emploi du temps était jusqu'`a maintenant un problème de satisfaction et non d'optimisation. Supposons, pour les besoins de l'illustration, que le cout de couverture [2] d'un certain quart de travail varie selon l'employé qui l'assure et que l'on veuille minimiser la somme des couts pour assurer le quart β sachant qu'ils sont de 3, 1, 3 et 2 respectivement pour les employés A, B, C et D. On peut écrire :

$$\min \quad \sum_{lu \leq j \leq di} = cj \qquad (1.9)$$

Elément $(c_j, (3, 1, 3, 2), e_{\beta j})$ $lu \leq j \leq di$ (1.10)

La fonction objectif est une somme de variables cj représentant le cout associé au quart β chaque jour de la semaine tandis que la contrainte (1.10) lie ces variables de cout aux variables de décision correspondantes.

La contrainte d'ordonnancement avec ressource cumulative, aussi appelée contrainte de mise en boîte et notée cumulative(S, P, R, k), agit sur la suite de variables $S = (s_1, s_2, \ldots s_n)$ représentant le moment où débute chacune de n taches de durée respective $P = (p_1, p_2, \ldots,$

p_n) et de consommation respective de ressource R = (r_1, r_2, . . . , r_n). Entant donné une quantité k de ressource disponible à tout moment, les taches doivent être planifiées de telle façon qu'on ne dépasse jamais cette capacité.

La contrainte de longueur de bloc pour une suite X = (x_1, x_2, . . . , x_n) de variables, notée stretch(X, L^{min}, L^{max}), contraint le nombre de variables consécutives de même valeur dans X. Soit V = {v_1, v_2, . . . , v_m}, l'ensemble des valeurs pouvant être prises par les variables de X et appelons v_i-bloc, toute sous suite maximale de variables prenant la valeur vi dans X. Les suites d'entiers L^{min} = (ℓ_1^{min}, ℓ_2^{min}, . . . , ℓ_m^{min}) et L^{max} = (ℓ_1^{max}, ℓ_2^{max}, . . . , ℓ_m^{max}) précisent, pour chaque valeur v_i, la longueur minimum ℓ_i^{min} et maximum ℓ_i^{max} de tout v_i –bloc dans X

La contrainte d'appartenance à un langage régulier pour une suite de variables X = (x_1, x_2, . . . , x_n), notée réguler(X, A), impose que la suite de valeurs prises par les variables de X corresponde à l'automate fini déterministe A.

Un mot du langage régulier décrit par La figure 1.1 donne un exemple d'automate correspondant à un patron courant pour des horaires de personnel. Chaque état est représenté par un cercle, le plus à gauche étant l'état initial. Chaque arc marque une transition possible, son étiquette représentant la valeur dont la présence permet cette transition.

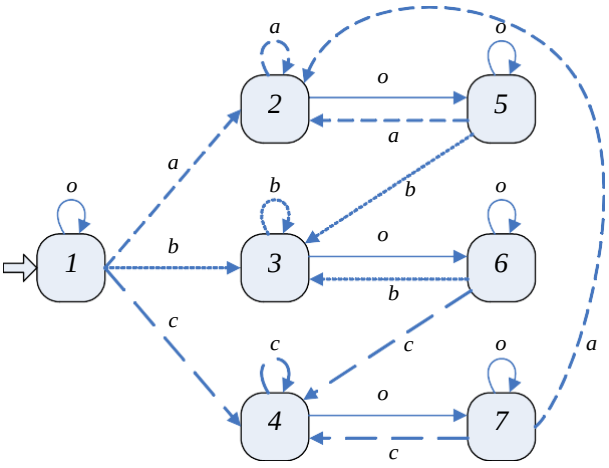


Figure 2.1 Automate correspondant à un patron courant pour des horaires de personnel.

Une telle contrainte sert typiquement à préciser des patrons de valeurs pour X, bien que sa puissance expressive dépasse ce cadre : par exemple, la contrainte de longueur de bloc décrite précédemment est un cas particulier.

7- Conseils de modélisation

Pour terminer cette section, voici quelques conseils utiles pour la modélisation.

7-1- Choisir les variables de décision

La première étape de modélisation consiste à définir les variables. En règle générale, il faut chercher à réduire le nombre de variables qu'on utilise, et ce, pour deux raisons. Tout d'abord, l'espace de recherche des solutions croit de façon exponentielle avec le nombre de variables. Ensuite, la taille des domaines est le plus souvent inversement proportionnelle au nombre de variables. Comme nous le verrons , les domaines des variables sont fort utiles pour guider la résolution du problème : à l'extrême, des domaines ne contenant que deux valeurs donnent très peu d'information.

Par conséquent, le choix de variables (0-1) est rarement heureux, pour les raisons que nous venons d'évoquer. Ce conseil est d'autant plus important pour le lecteur familier avec la programmation en nombres entiers et pour qui la modélisation en variables 0-1 est peut-être un réflexe. Prenons par exemple l'affectation d'un élément d'un ensemble J à chacun des éléments d'un autre ensemble I. On aurait comme une modélisation typique de programmation en nombres entiers :

$x_{ij} \in \{0, 1\} \quad \forall \quad i \in I, \; j \in J$

$\sum_{j \in J} xij = 1 \quad \forall \quad i \in I$

Ou $x_{ij} = 1$ si et seulement si l'élément j est affecté à i, alors qu'en programmation par contraintes on écrirait plutôt :

$y_i \in J \quad \forall \quad i \in I$

ou` yi prend pour valeur l'élément de J qui lui est affecté.

Une fois les variables choisies, la définition des domaines suit naturellement.

8- Modèles possibles

Les dimensions d'un problème apparaissent dans le modèle sous la forme de domaines des variables et d'indices de celles-ci. Ces rôles sont souvent interchangeables et l'exploration des différentes possibilités permet de faire un choix plus judicieux en fonction des contraintes qu'il faudra exprimer. Par exemple, dans le modèle(1.1),(1.2) ou encore(1.6),(1.2) ,les employés se retrouvent dans les domaines alors que les jours et les quarts jouent le rôle d'indices des variables de décision. Un modèle dual pouvant le remplacer place les quarts comme domaines (avec la valeur additionnelle δ pour indiquer un congé) et les jours et employés comme indices :

Distribue ((1, 1, 1), (α, β, γ), (p_{jk}) A ≤ k ≤D) lu ≤ j ≤ di (1.11)

P_{jk} ∈ {α, β, γ, δ} lu ≤ j ≤ di, A ≤ k ≤ D (1.12)

Ce modèle a l'avantage de pouvoir exprimer plus aisément des contraintes sur les horaires individuels des employés, donnés par la suite de variables (pjk)j=lu...di pour un employé k.

Utiliser des contraintes globales Bien que cela demande une certaine dose d'expérience, l'utilisation de contraintes globales mène habituellement a une résolution plus efficace du problème. Comme leur nom l'indique, ces contraintes prennent un point de vue plus global qui se traduit par des raisonnements plus sophistiqués. La contrainte en est un bon exemple.

9- Ajouter des contraintes redondantes

Supposons dans l'exemple que les durées des quarts de travail peuvent varier selon le type de quart et le jour de la semaine et qu'elles sont données par (d_{ij}) lu ≤ j ≤ di , α ≤ i≤ δ. Supposons aussi une semaine de 35 heures précisément pour les employés A, B, D et de 20 heures pour l'employé C, travaillant à temps partiel.

On peut modéliser ainsi cette situation :

Elément (h_{jk} , (d_{ij}) α ≤ i ≤ δ , p_{jk}) lu ≤ j ≤ di, A ≤ k ≤ D (1.13)

$\sum_{lu \le j \le di} h_{jk} = 35$ k ∈ {A, B, D} (1.14)

$\sum_{lu \le j \le di} h_{jc} = 20$ (1.15)

Malheureusement, une telle modélisation n'est pas très efficace parce que les contraintes (1.14) et (1.15) implantent généralement la cohérence de bornes Une option intéressante prend le point de vue d'un problème de mise en boite ou, pour chaque quart de travail, on crée une petite boîte de largeur unitaire et dont la hauteur est égale à sa durée dij. Ces boites doivent toutes être placées dans une plus grande, de largeur quatre (le nombre d'employés) et de hauteur 35 (le nombre d'heures travaillées par semaine). On peut visualiser cette grande boite comme un ensemble de bandes de largeur unitaire, une par employé (fig. 2.2). La bande dans laquelle se trouve une petite boite s'interprète alors comme l'employé effectuant le quart de travail correspondant. Pour modéliser le fait que l'employé C ne travaille que 20 heures par semaine, on ajoute une boite fictive déjà placée dans la bande C et de hauteur 15, laissant une hauteur résiduelle de 20 heures pour les autres taches.

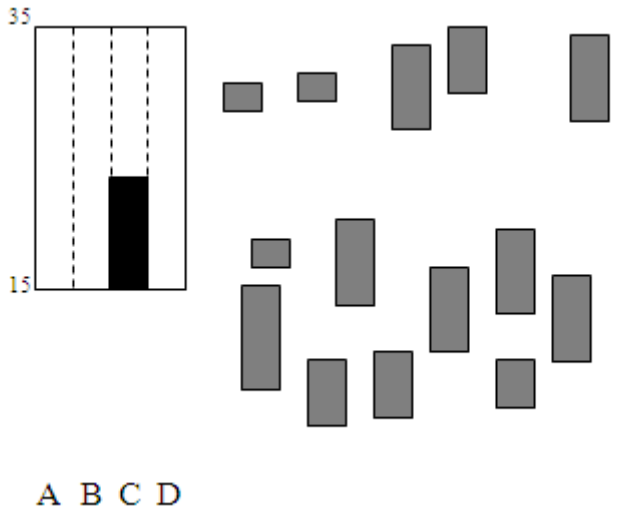


Figure 2.2 Modélisation des charges de travail individuelles comme un problème de mise en boite. Les boites Grises représentent les quarts de travail et la boite noire déjà placée permet de réduire la charge effective de l'employé C

10- L'ajout de la contrainte :

Cumulative ((e_{lu} α , . . . , e_{di} γ, C), (1, . . . , 1), (d_{lu} α, . . . , d_{di} γ, 15), 35) (1.16)

Réintroduit les variables eij et ajoute donc une redondance dans le modèle. Mais parce qu'il exprime d'une façon différente et complémentaire une certaine sous- Structure du problème, ce superflu apparent peut mener à une résolution plus efficace.

11- Propagation

Le moteur de résolution à la base de la PPC, appelé propagation de contraintes, diffuse à travers l'ensemble des contraintes les résultats de raisonnements locaux à chacune d'entre elles, obtenus par des algorithmes de filtrage. Chaque type de contrainte possède son propre algorithme de filtrage pour éliminer du domaine des variables de sa portée les valeurs localement incohérentes ne pouvant pas mener à une solution. La communication entre les différents algorithmes de filtrage se fait à travers les domaines des variables. L'algorithme de filtrage d'une contrainte pourra s'activer lorsque le domaine d'une des variables de sa portée est modifié. Ce mécanisme facilite grandement la coopération de contraintes hétéroclites et explique la flexibilité dans la formulation des modèles.

Il est préférable pour certains types de contraintes de ne pas activer leur algorithme de filtrage chaque fois qu'un domaine pertinent est modifié. Parfois, il est suffisant de ne l'activer que lorsque ce domaine ne contient plus qu'une seule valeur ou encore lorsque la plus petite ou la plus grande valeur du domaine est retirée. Bien qu'un algorithme de filtrage, en réduisant un domaine, puisse en activer un autre et ainsi de suite, la propagation de contraintes se termine nécessairement puisque chaque activation est le résultat du retrait d'une valeur dans un domaine et que nous ne pouvons qu'en retirer un nombre fini de fois : en effet, le nombre de variables dans un modèle est fini et chaque domaine est également fini. La réduction d'un seul domaine provoque parfois l'activation de plusieurs algorithmes de filtrage. Une autre propriété importante est que le résultat final de la propagation ne dépend pas de l'ordre dans lequel ceux-ci sont activés.

Caractérisation du niveau de cohérence :

Bien qu'il soit certainement profitable qu'un algorithme de filtrage élimine des valeurs incohérentes du domaine des variables, il est également souhaitable de caractériser les valeurs à éliminer. On utilise la notion de niveau de cohérence[3] comme mesure de qualité des algorithmes de filtrage. Du point de vue théorique, plusieurs niveaux de cohérence ont été définis et ceux-ci servent à décrire le travail effectué par un algorithme de filtrage. Historiquement, on a d'abord appliqué la notion de cohérence aux contraintes binaires.

Définition :

Etant donné une contrainte binaire c de portée {x, y} et les domaines D_x et D_y pour les variables x et y respectivement, la cohérence d'arc est atteinte pour c si : et seulement si

- pour chaque valeur $v \in D_x$, il existe $(v, v') \in c$ ou $v' \in D_y$ et
- pour chaque valeur $v \in Dy$, il existe $(v', v) \in c$ ou $v' \in D_x$.

La cohérence d'arc garantit que les domaines des deux variables ne contiennent que des valeurs pouvant mener à une solution pour la contrainte: chaque valeur pour une variable est supportée par au moins une autre pour l'autre variable (on dit qu'elle constitue son support). On peut étendre cette notion de cohérence, comme toutes les autres d'ailleurs, à l'ensemble des contraintes d'un problème de satisfaction de contraintes P. Si chacune des contraintes atteint la cohérence d'arc, on dit que la cohérence d'arc est atteinte pour P. Notons que cela ne veut pas dire que toute valeur restante participe nécessairement à une solution de P : la propriété de cohérence demeure locale à chacune des contraintes.

Cette notion de cohérence se généralise facilement aux contraintes mettant en jeu plus de deux variables. Selon les auteurs, on parlera tantôt de cohérence d'hyper-arc, tantôt de cohérence d'arc généralisée. Nous préférerons l'expression cohérence de domaine.

Définition :

étant donné une contrainte c de portée $\{x_1, x_2, \ldots, x_k\}$ et les domaines D_{xi} $(1 \leq i \leq k)$, la cohérence de domaine est atteinte pour c si et seulement si pour chaque x_i $(1 \leq i \leq k)$ et chaque valeur $v \in D_{xi}$, il existe $(v_1, \ldots, v_{i-1}, v, v_{i+1}, \ldots, v_k) \in c$ ou $v_j \in D_{xj}$.

En général, atteindre la cohérence de domaine pour une contrainte demande un effort qui croit de façon exponentielle avec le nombre de variables dans sa portée 4. La cohérence de bornes est une option plus abordable, quoique strictement moins forte. Elle s'appuie sur une relaxation des domaines discrets des variables à des intervalles continus. On note x^{min} et x^{max} les plus petite et plus grande valeurs dans D_x, respectivement.

Le terme « consistance » est également utilisé pour sa proximité lexicale (mais malheureusement pas sémantique) de l'anglais consistance. Nous verrons néanmoins que la structure sous-jacente peut parfois être exploitée pour que l'effort demeure polynomial.

Définition Etant donné une contrainte c de portée $\{x_1, x_2, \ldots, x_k\}$ et les domaines D_{xi} $(1 \leq i \leq k)$, la cohérence de bornes est atteinte pour c si et seulement si pour chaque

x_i $(1 \leq i \leq k)$:

- il existe ($r_1, \ldots, r_{i-1}, x_i^{min}, r_{i+1}, \ldots, r_k$) ∈ c ou r_j est un nombre réel appartenant à l'intervalle [x_j^{min}, x_j^{max}] et
- il existe ($r_1, \ldots, r_{i-1}, x_i^{max}, r_{i+1}, \ldots, r_k$) ∈ c ou r_j est un nombre réel appartenant à l'intervalle [x_j^{min}, x_j^{max}].

11-1-Algorithmes de filtrage

Les algorithmes de filtrage peuvent être classés en deux catégories, selon le type de contrainte pour lequel ils sont définis. Les contraintes simples ont généralement des algorithmes de filtrage relativement faciles à définir alors que les contraintes globales font appel à des techniques plus complexes, parfois issues de la recherche opérationnelle.

11-2- Contraintes binaires

Le développement d'algorithmes de filtrage des contraintes binaires a constitué un domaine de recherche extrêmement actif au cours des dernières années. Les algorithmes, qui permettent d'atteindre la cohérence de domaine, sont notés par les lettres AC (Arc Consistance) suivies d'un numéro les identifiant. Ainsi, une dizaine d'algorithmes (AC-3, AC-4, AC-5, AC-6, AC-7, AC-2000, AC-2001, AC-8, AC-3d, AC-3.2 et AC-3.3) se distinguent par leur complexité en temps et en espace mémoire utilisé. Nous donnons ici les grandes lignes de quelques-unes de ces techniques puisque plusieurs sont des variations d'une même idée. Les mesures de complexité concernent une contrainte binaire liant, par exemple,

x et y.

AC-3 est l'algorithme de base qui applique directement la définition de la cohérence de domaine, c'est-a-dire qu'il vérifie que pour chaque valeur a de D_x, il existe bien une valeur b dans D_y agissant comme support aux yeux de la contrainte pour laquelle on filtre. La complexité en espace est donc nulle ; par contre, la complexité en temps est dans $O(d^3)$, ou` d est la taille du plus grand domaine.

AC-4, pour accélérer le temps de filtrage, propose de conserver dans un tableau la liste de toutes les solutions partielles (en regard de x et de y) associées à chaque valeur a de D_x. En implantant correctement cette structure de données, on peut réduire la complexité en temps à $O(d^2)$, mais la complexité en espace atteint elle aussi $O(d^2)$

AC-6 et AC-7 poussent un peu plus loin cette idée en montrant qu'il est possible de ne conserver qu'un seul support pour chaque valeur. Pour chaque paire variable-valeur (x, a), ces algorithmes maintiennent la liste de toutes les autres paires variable-valeur supportées par

(x, a). Ainsi, on conserve une complexité en temps de $O(d^2)$ mais la complexité en espace est réduite à O(d).

Les autres techniques proposent des raffinements ainsi que la combinaison et l'amélioration des techniques vues précédemment.

11-3-Contraintes arithmétiques

Dans le cas des contraintes arithmétiques, leur sémantique permet de spécialiser et d'accélérer les algorithmes de filtrage. Comme auparavant, soit x^{max}, la valeur maximum du domaine d'une variable x et x^{min}, sa valeur minimum. Le filtrage des contraintes arithmétiques telles que =, /=, <, ≤, > et ≥ liant deux variables x et y s'effectue par l'application des traitements suivants :

- x = y : lorsqu'une des deux variables prend une valeur (x = v), la même valeur est attribuée à la seconde (y = v). Réciproquement, lorsqu'une valeur est retirée du domaine d'une des variables ($v \notin D_x$), elle est aussi retirée du domaine de la seconde ($v \notin D_y$).

- x ≠ y : lorsqu'une des deux variables prend une valeur (x = v), cette valeur est alors retirée du domaine de l'autre variable (v ∉ Dy).

- x ≤ y (x < y) : le filtrage de cette contrainte consiste à éliminer toute

Valeur v de D_x qui soit supérieure (ou égale) à y^{max}. De la même maniéré, on doit retirer toute valeur v de D_y qui soit inférieure (ou égale) à x^{min}. Le traitement pour x ≥ y (x > y) est très semblable.

Par exemple, si on considère les deux variables x et y telles que D_x = {1, 2, 3, 4} et

D_y = {3, 4, 5} ainsi que la contrainte x > y, le filtrage associé permet de conclure que x = 4 et y = 3. Il est à noter que toutes ces méthode de filtrage pour les contraintes arithmétiques binaires atteignent la cohérence de domaine.

Les contraintes arithmétiques non binaires ont généralement la forme $a_1x_1 + a_2x_2 + \cdots + a_nx_n = d$ ou $a_1, \ldots,$ an sont des constants, $x_1, \ldots, x_n$ des variables, d est une constante et ou l'opérateur de comparaison peut-être substitué par , ou ≥. Pour effectuer le filtrage de ces contraintes, on se

contente généralement d'atteindre la cohérence de borne en raisonnant sur les valeurs extrêmes de chaque variable. Par exemple, si on veut filtrer la contrainte $2x - y - 3z \leq 5$ pour la variable y, on déduit que $y \geq 2x - 3z - 5$ et donc qu'il est possible d´éliminer du domaine (D_y) de y toutes les valeurs inférieures à $2x^{min} - 3z^{max} - 5$. On procède ensuite de la même maniéré pour x et z en éliminant de D_x toutes les valeurs supérieures à $(y\ max + 3z\ max + 5)/2$ et en retirant de D_z toutes les valeurs inférieures à $(2x^{min} - y^{max} - 5)/3$

11-4-Contraintes globales

Nous avons quelques contraintes globales fréquemment utilisées en programmation par contraintes. Nous donnons ici quelques détails sur les algorithmes de filtrage associés à la contrainte de distribution de valeurs, à la contrainte d'ordonnancement avec ressource cumulative et à la contrainte d'appartenance à un langage régulier.

Contraintes de distribution de valeurs. La contrainte distribue (Y, V, X) peut

être représentée par un ensemble de contraintes plus simples de la manière suivante :

$$y_i = \sum_{i \in \{1,2,\ldots,n\}} (xi = vi) \qquad j \in \{1, 2, \ldots, m\}$$

Si on considère de manière simultanée l'ensemble des variables Y, il est possible d'obtenir un filtrage plus efficace. L'algorithme de filtrage proposé par Régin [4] est le plus couramment utilisé pour atteindre et maintenir la cohérence de domaine pour cette contrainte. Cette technique est basée sur un algorithme de flot appliqué à un graphe particulier nommé le graphe de valeurs G = (N, A). Ce graphe (fig. 1.3) est constitué de deux couches de nœuds, l'une représentant les variables et l'autre, l'ensemble des valeurs. Un arc (i, j) est défini entre une paire de nœuds de types différents (une variable x_i et une valeur v_j) lorsque la valeur v_j appartient au domaine de la variable x_i.

On ajoute ensuite à ce graphe deux nœuds additionnels : s lié à tous les nœuds de type variable et t relié à tous les nœuds de type valeur. à chaque arc (i, j) du graphe, on associe une paire de valeurs (d_{ij}, c_{ij}) ou d_{ij} représente la demande minimum devant le traverser et c_{ij} exprime sa capacité maximale. Tous les arcs reliant s aux nœuds de variables se voient attribuer une demande et une capacité de 1, (d_{si}, c_{si}) = (1,1), alors que les arcs reliant les variables aux valeurs de leur domaine sont marqués d'une demande de 0 et d'une capacité de 1, (d_{ij}, c_{ij}) = (0,1). On attribue ensuite aux arcs reliant une valeur v_j au nœud une demande

égale à la valeur minimum du domaine de y_j et une capacité la valeur maximale du domaine de y_j, $(d_{jt}, c_{jt}) = (y_i^{min}, y_i^{max})$.

Enfin, on associe à l'arc reliant t à s la paire (n,n), ou` n correspond au nombre de variables de X.

Pour vérifier si la contrainte est cohérente par rapport aux domaines des variables X, il suffit de définir un flot réalisable dans le graphe des valeurs (fig. 2.3), soit de faire circuler n unités de flot de s vers t. Définir un flot consiste à attribuer à chaque arc (i, j) une valeur f_{ij} de manière à ce que :

$d_{ij} \leq f_{ij} \leq c_{ij} \quad \forall (i, j) \in A$

$\sum_{i|(i,k)\in A} fik = \sum_{i|(i,k)\in A} fik$

$\forall k \in N$

C'est-`a-dire que le flot de chaque arc soit compris entre sa demande et sa capacité, et ce, tout en respectant la conservation du flot à chaque nœud.

Il est ensuite possible de maintenir la cohérence de domaine de manière très efficace en travaillant avec les composantes fortement connexes du graphe résiduel associé au flot. Le lecteur peut consulter Régin [4] pour connaitre les détails de cette technique.

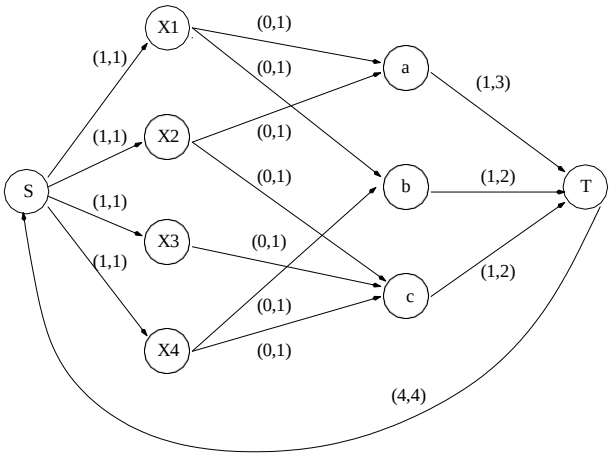


Figure 2.3 Graphe des valeurs pour le filtrage de distribue (Y, V, X)

Ou

$Y = \{y_a, y_b, y_c\}$, $D_Y = \{\{1, 2, 3\}, \{1, 2\}, \{1, 2\}\}$, $V = \{a, b, c\}$ et $X =$

$\{x_1, x_2, x_3, x_4\}$, $D_X = \{\{a, b\}, \{a, c\}, \{c\}, \{b, c\}\}$.

Contraintes d'ordonnancement avec ressource cumulative. Pour la contrainte cumulative(S, P, R, k), on a proposé plusieurs algorithmes de filtrage (atteignant la cohérence de borne). Les algorithmes principaux sont basés sur le calcul de bilans énergétiques. L'énergie utilisée par une tache i durant son exécution est donnée par la valeur $e_i = r_i p_i$, produit de la durée de la tache par la ressource consommée. Nous présentons ici une première règle permettant de déterminer par raisonnement énergétique si un problème est irréalisable.

On peut estimer durée minimale pi(t1, t2) d'exécution d'une tache i sur un intervalle de temps $[t_1, t_2]$ à partir des dates possibles d'exécution de i, à savoir les intervalles de temps $[s_i, s_i + p_i]$ tels que $s_i \in \{s_i^{min}, \ldots, s_i^{max}\}$ (ou s_i^{min} et s_i^{max} représentent les dates au plus tôt et au plus tard de début de la tache i).

Une estimation provient de $p_i(t_1, t_2) = \max(0, \min(p_i, t_2 - t_1, s_i^{min} + p_i - t_1, t_2 - s_i^{max}))$ et l'énergie consommée par i dans l'intervalle $[t_1, t_2]$ est supérieure ou égale à

$e_i(t_1, t_2) = r_i \bullet p_i(t_1,t_2)$. L'énergie totale minimale consommée par les tâches est donnée par $\sum_i = 1,2,..,n\ e_i(t_1, t_2)$. Elle doit être au plus égale à l'énergie totale disponible $k(t_2 - t_1)$, sinon le problème est irréalisable. Les valeurs de t1 et t2 sont choisies parmi les temps de début au plus tôt (s^{min}) et au plus tard (s^{max}) de toutes les taches.

Baptiste et al. [5] montrent comment le raisonnement énergétique permet aussi d'effectuer des ajustements des bornes s^{min} et s^{max} en $O(n^3)$, ainsi que d'autres règles de filtrage associées à la contrainte cumulative. Par une étude expérimentale, les auteurs ont aussi mis à jour des dominances entre différentes règles d'ajustement applicables au RCPSP. Ils montrent notamment que le comportement du raisonnement énergétique dépend clairement du type de problème testé. En effet, son cout en temps de calcul se ressent fortement sur les problèmes hautement disjonctifs ou un grand nombre de taches sont incompatibles deux à deux. Au contraire, il prouve son efficacité sur les problème Hautement cumulatifs.

Toutefois, dans le cas particulier ou` l'on ne peut exécuter qu'une seule tâche à la fois (k = 1), il est généralement plus efficace d'utiliser l'algorithme du pour effectuer le filtrage de la contrainte cumulative.

Contraintes d'appartenance à un langage régulier. L'algorithme de filtrage pour la contrainte réguler(X, A) parcourt la suite de variables $X = (x_1, x_2, \ldots, x_n)$ en appliquant l'automate A, bâtissant ainsi un graphe multiparti $(N^1, N^2, \ldots N^{n+1}, A)$ ou chaque partie N^i contient un sommet par état de l'automate et Chaque arc de A correspond à une paire variable-valeur (x_i, v_j) [6]. Soit τ , la fonction de transition de l'automate. Dans un premier temps, on ajoute un arc (i, j) allant du nœud correspondant à l'état k dans la partie Ni au nœud correspondant à l'état ℓ dans la partie suivante (N^{i+1}) s'il existe une valeur $v_j \in D_i$ telle que τ (k, vj) = ℓ. Dans un second temps, on retire tout arc n'appartenant pas à un chemin de l'état initial dans la partie N 1 à un état final dans la partie Nn+1. Un tel chemin agit comme support des arcs (autrement dit, des paires variable-valeur) le composant. Toute valeur

$v_j \in D_{xi}$ pour laquelle aucun arc (i, j) n'est présent dans le graphe se voit retirée du domaine puisqu'elle n'a pas de support. La figure 10.4 fournit un exemple. Puisque aucun arc (2, b) ou (4, b) ne subsiste, le filtrage initial retire la valeur b du domaine des deuxième et quatrième variables. Le graphe est mis à jour au fur et à mesure des modifications des domaines des variables. L'algorithme résultant assure la cohérence de domaine. Son temps de calcul est linéaire dans la taille du graphe pour l'initialisation et linéaire dans la taille des mises à jour du graphe pour les appels subséquents

12- Recherche

Une fois la propagation des contraintes terminée, il n'est pas garanti que toutes les variables auront été instanciées : certains domaines peuvent encore contenir plusieurs valeurs. Puisque la propagation n'est en pratique jamais suffisante pour déterminer des solutions réalisables, il est généralement nécessaire d'entreprendre une recherche de ces solutions. Ce processus s'effectue par l'ajout dynamique de contraintes et la génération d'un arbre de recherche ou chaque feuille représente une solution réalisable.

Les étapes de branchement à chaque nœud N de l'arbre de recherche peuvent se résumer comme suit (le nœud racine n'étant en fait que le problème lui-même) :

1. Effectuer la propagation des contraintes de N . Si le domaine d'une variable se vide, alors on rejette N et on sélectionne le prochain nœud ;

2. choisir une variable x non instanciée ($|D_x| > 1$) ; si toutes les variables sont instanciées, alors on a une solution réalisable ;

3. Choisir une valeur $v \in D_x$ possible pour x ;

4. Créer deux nouveaux nœuds, N ′ et N ″, auxquels on ajoute de nouvelles

contraintes à l'ensemble des contraintes du nœud courant C(N) :

(a) $C(N') = C(N) \cup (x = v)$

(b) $C(N'') = C(N) \cup (x \neq v)$

5. Ajouter N ′ et N ″ à la liste des nœuds non traités.

Dans cette procédure, on observe trois degrés de liberté (en italique) qui permettent d'adapter la recherche de solutions à la structure particulière du problème. Ces points d'intervention sont le choix de variable, le choix de valeur et la sélection du prochain nœud à traiter. Avant d'aborder ces choix plus en détail, mentionnons qu'il est possible d'effectuer la recherche de solutions en fragmentant le domaine des variables plutôt qu'en leur assignant une valeur. Pour ce faire, on remplace les contraintes de branchement ((x = v) et (x /= v)) par des contraintes permettant de fractionner le domaine d'une variable en deux Ensembles complémentaires.

12-1 -Heuristiques de choix de variable

Lorsque plusieurs variables restent non instanciées, il est nécessaire de choisir la prochaine variable qui fera l'objet d'un branchement. Cette décision a une incidence majeure sur les performances d'une stratégie de recherche et il est important d'effectuer un choix judicieux.

Ordre statique opposé à ordre dynamique.

Lorsque l'ordre de sélection des variables est déterminé avant le début de la recherche, on parle d'un ordre statique alors qu'un ordre dynamique sous-entend que les variables sont sélectionnées chaque nœud en fonction du contexte. Courant et d'un critère donné.

Un ordonnancement statique permet d'accélérer le processus de sélection de variables puisque les calculs ne sont effectués qu'une seule fois, avant le début de l'exploration. Toutefois, cette stratégie ne tire pas profit de l'information disponible et mise à jour durant la recherche. L'ordonnancement dynamique, bien qu'un peu plus couteux en temps de calcul, permet de prendre de meilleures décisions en exploitant l'information introduite par les nœuds

explorés précédemment. Parce qu'il est beaucoup plus performant, on utilise presque toujours l'ordre dynamique.

Dans le choix de la prochaine variable à instancier, on tente de sélectionner des candidats qui mènent à une impasse ; c'est ce qu'on appelle la stratégie de l'échec en premier. L'idée derrière ce principe est de tenter de résoudre le plus rapidement possible les aspects les plus difficiles du problème à traiter. En effet, si une variable est difficile à instancier, cette difficulté ne fera qu'augmenter au fur et à mesure que d'autres variables seront fixées. Il est donc primordial de traiter les variables problématiques le plus tôt possible.

Il existe plusieurs mesures de la difficulté d'instanciation de chaque variable, mais la plus utilisée est sans contredit la taille de son domaine. En effet, le domaine d'une variable qui contient peu de valeurs risque de se vider plus rapidement qu'un domaine qui en contient encore plusieurs. Ainsi, en fixant d'abord les variables qui ont de petits domaines, on réduit les chances qu'elles créent des impasses plus tard.

Ce critère est toutefois général et, lorsqu'on connait bien la structure d'un problème particulier, il est possible de concevoir une stratégie adaptée qui soit plus performante. Ainsi, dans l'exemple d'emploi du temps examiné dans ce chapitre, on pourrait choisir comme stratégie de sélectionner en premier les variables e_{ij} ou p_{jk} ayant le plus petit domaine. Si la définition du problème était plus riche, on pourrait aussi utiliser d'autres données telles la compétence des employés ou la difficulté à combler certains quarts de travail comme les week-ends par exemple.

-Problème d'optimisation.

Lorsqu'on résout un problème d'optimisation et qu'on dispose d'un cout c_{xv} associé à chaque paire variable-valeur (x, v), il est aussi possible de sélectionner les variables pour tenter d'obtenir des solutions de bonne qualité. Les critères de cout minimum et de regret maximum sont les plus fréquemment utilisés.

Le critère de cout minimum consiste à choisir la variable x associée au c_{xv}* le cout minimum pour toutes les paires variable-valeur encore possibles. Dans l'exemple d'emploi du temps, on choisirait donc la variable e_{ij} dont le domaine contient encore l'employé minimisant le cout de couverture de la tache j. Cette méthode s'avère parfois efficace, mais elle est généralement trop myope (elle considère trop peu d'informations) pour donner de bons résultats.

Le critère de regret maximum, un peu comme celui de l'échec en premier, tente de fixer les variables pour lesquelles l'obtention d'un bon cout est difficile. Cela se mesure avec le regret, soit la différence de cout entre les deux meilleures valeurs pour x. Ce critère permet de traiter rapidement les variables pour lesquelles.

La détérioration de l'objectif sera importante s'il devient impossible de leur attribuer la meilleure valeur de leur domaine.

12-2- Heuristiques de choix de valeur

Une fois la variable à traiter définie, la prochaine étape consiste à choisir une valeur dans son domaine. Ce choix est moins crucial que le choix de variable et l'approche choisie est contraire à celle présentée. En effet, ici on essaie plutôt de maximiser les chances de réussite et la qualité de la solution produite. On choisit donc la meilleure valeur présente dans le domaine de la variable choisie.

La meilleure valeur possible peut être définie comme celle qui maximise les chances de satisfaction dans le cas d'un problème de satisfaction de contraintes ou celle qui optimise une fonction objectif dans le cas d'un problème d'optimisation.

Si, dans notre exemple, la contrainte la plus contraignante est l'attribution d'un minimum d'heures de travail hebdomadaire à chaque employé, on pourrait sélectionner l'employé à qui on a attribué le moins d'heures parmi ceux du domaine de la variable e_{ij} choisie.

12-3- Exploration de l'arbre

En dernier lieu, il faut déterminer la maniéré dont sera parcouru l'arbre de recherche, c'est-à-dire dans quel ordre seront développés les nœuds créés par la procédure de branchement.

-Recherche en profondeur

On utilise le plus souvent la stratégie de recherche en profondeur (Depth First Search: DFS) en programmation par contraintes, et ce, pour plusieurs raisons. D'abord, cette stratégie de recherche est particulièrement bien adaptée à la détermination d'une solution réalisable. En effet, puisque les solutions du problème à résoudre se retrouvent dans les feuilles de l'arbre de recherche, la meilleure stratégie consiste à plonger le plus rapidement possible. Comme la

PPC s'est surtout développée autour des problèmes de satisfaction de contraintes, la recherche en profondeur à longtemps été la méthode de référence pour l'exploration arborescente. illustre l'ordre dans lequel les feuilles d'un arbre de recherche sont évaluées par une recherche en profondeur. De plus, l'absence d'une méthode de borne universelle (comme la relaxation linéaire en programmation mathématique) rend l'utilisation d'une recherche par meilleur candidat (Best First Search: BFS) difficile à implanter dans les librairies de PPC. Enfin, chaque nœud conservé d'un arbre de recherche en PPC nécessite la sauvegarde de toutes les variables, de tous les domaines et de toutes les contraintes. Le fait que la recherche en profondeur génère peu de nœuds (consommant donc peu de mémoire) a constitué un atout important lors de l'implémentation des premiers solveurs de contraintes.

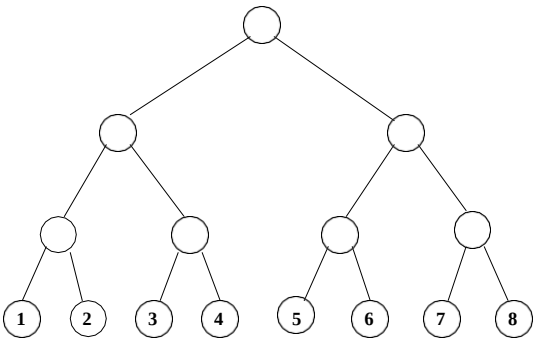


Figure 2.4 Ordre de parcours des feuilles lors d'une recherche en profondeur.

Recherche à divergence limitée.

Le principal inconvénient de la recherche en profondeur est qu'elle accorde une très grande importance aux décisions qui sont prises dans les premiers niveaux de l'arbre de recherche. En effet, avant de reconsidérer une mauvaise décision prise dans les premières branches, il est nécessaire d'explorer une sous-arborescence souvent très importante. La recherche à divergence limitée (Limited Discrepancy Search: LDS) fut donc proposée pour pallier ce problème et diriger la recherche plus tôt vers de meilleures solutions.

Le principe de base derrière la recherche à divergence limitée est la confiance portée à l'heuristique de choix de valeur. Lorsqu'une branche de l'arbre de recherche suit une affectation variable-valeur qui n'est pas la première suggérée par l'heuristique, on parle alors d'une divergence. Le processus de recherche commence donc par traverser l'arbre de recherche en ne permettant aucune divergence (ce qui correspond à toujours suivre la branche de gauche) ; ensuite il tolère une seule divergence, puis deux, trois, etc. De cette maniéré, les premières solutions établies sont celles qui respectent au mieux les suggestions de

l'heuristique de choix de valeur. De plus, une mauvaise décision prise en haut de l'arbre de recherche sera vite annulée lorsque le niveau de divergence permis est bas. On essaie généralement d'effectuer les choix divergents en haut de l'arbre de recherche, puisqu'à ce niveau on dispose de moins d'information et l'heuristique de choix de valeur est plus susceptible de se tromper. La figure 2.6 illustre l'ordre de visite des feuilles lors d'une recherche à divergence limitée.

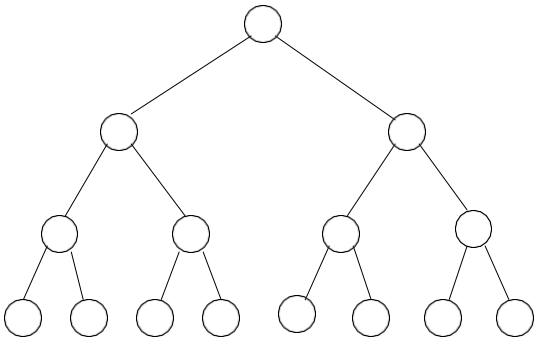


Figure 2.5 Ordre de parcours des feuilles lors d'une recherche à divergence limitée. L'ordre apparait à l'intérieur du cercle alors que le nombre de divergences est indiqué en dessous.

12-4- APPROCHES HYBRIDES

On nomme généralement approches hybrides les méthodes qui utilisent et combinent la programmation par contraintes et d'autres techniques issues de la recherche opérationnelle. Le développement des méthodes hybrides va bon train depuis quelques années et un mémoire dédié à ce sujet a récemment été publié [7]. Bien que des algorithmes complexes issus de la recherche opérationnelle soient utilisés depuis plusieurs années dans le filtrage des contraintes globales, l'attention des chercheurs a récemment porté sur l'aspect de l'optimisation d'une mesure de qualité. Des méthodes telles que le filtrage basé sur les couts réduits, la génération de colonnes basée sur la programmation par contraintes, la décomposition de Bender logique et la recherche à grand voisinage illustrent bien cet effort de mise en commun des connaissances et des techniques. Nous survolons ici quelques- unes de ces méthodes qui sont parfois utilisées pour résoudre des problèmes d'ordonnancement ou d'emploi du temps.

L'idée derrière le filtrage basé sur les couts réduits est d'éliminer du domaine d'une variable non seulement les valeurs menant à des incohérences, mais aussi toutes celles ne menant pas à la solution optimale. Cette méthode est applicable lorsqu'on dispose d'une borne inférieure sur la fonction objectif omin (pour un problème de minimisation), d'une solution réalisable de valeur z et des couts réduits c' associés au fait que la variable x prenne

la valeur v. Pour effectuer un filtrage basé sur les couts réduits, il suffit d'éliminer de D_x toutes les valeurs v telles que $o^{min} + c'_{xv} > z$

Cette méthode est particulièrement efficace lorsqu'on dispose d'un algorithme spécialisé pour calculer les couts réduits.

La génération de colonnes basée sur la programmation par contraintes essaie de tirer profit de la flexibilité de la programmation par contraintes pour résoudre les sous-problèmes associés à la génération de colonnes Ces problèmes. qui consistent par exemple à construire une journée de travail, la route d'un Véhicule ou le plan de production d'une machine, sont généralement sujets à un grand nombre de contraintes toutes aussi variées que complexes. Le riche langage de la programmation par contraintes permet donc d'exprimer ces contraintes naturellement, et ce, peu importe leur structure. Les problèmes d'emploi du temps se prêtant bien à cette hybridation, le modèle utilisant les variables p_{jk}, légèrement modifié, pourrait servir de générateur de colonnes dans une approche par génération de colonnes.

La décomposition de Bender logique, tout comme la génération de colonnes, s'articule autour d'un problème maitre et de sous-problèmes résolus tour à tour. Le rôle du problème maitre est de proposer une solution optimale au problème à résoudre tout en ignorant un certain nombre de contraintes difficiles à traiter. La réalisabilité de la solution proposée est ensuite évaluée par rapport à l'ensemble complet des contraintes spécifiées dans les sous-problèmes. Dans le cas du non- respect de certaines contraintes, des coupes linéaires s'ajoutent à la description du problème maitre. Dans la version hybride, les sous-problèmes sont résolus en programmation par contraintes avec un modèle qui détermine des instanciations variable-valeur toujours incohérentes (ce qu'on appelle des no good) qui, une fois transmises au problème maitre, sont converties en coupes linéaires.

La recherche locale basée sur la PPC est issue du désir de résoudre des problèmes complexes à l'aide de méthodes heuristiques tout en bénéficiant de la flexibilité de la programmation par contraintes et de ses puissants mécanismes de propagation. Le principe consiste à modéliser les opérateurs et leur voisinage comme un problème combinatoire que l'on résout à l'aide de la programmation par contraintes. Ainsi, puisque cette technique permet d'explorer un voisinage implicitement plutôt qu'explicitement, il est possible de définir des voisinages plus grands et il y a donc moins de risques de se faire piéger dans un minimum local.

13-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 3 :

Etude de cas

1-Introduction

Dans ce chapitre nous allons présenter notre étude opérationnelle à travers laquelle nous avons réalisé notre planificateur. Nous commençons au premier lieu par la présentation de notre environnement de développement et par la suite on terminera par la présentation de quelques fonctionnalités assurées par ce système.

2-Outils de développement

2-1-Optaplanner

Qu'est-ce qu' OptaPlanner ?

Chaque organisation est confrontée à des problèmes de planification : fournir des produits ou des services avec un ensemble limité de ressources limitées (employés, actifs, temps et argent). OptaPlanner optimise une telle planification pour faire plus d'affaires avec moins de ressources. C'est ce qu'on appelle la programmation de la satisfaction des contraintes (qui fait partie de la discipline de la recherche opérationnelle).

OptaPlanner est un moteur de satisfaction de contraintes léger et intégrable qui optimise les problèmes de planification. Il résout des cas d'utilisation tels que :

- Gestion des horaires de travail: horaires infirmiers, réparateurs, …
- Planification de l' agenda: planification des réunions, des rendez-vous, des travaux de maintenance, des annonces, …
- Emploi du temps pédagogique: planification des cours, stages, examens, présentations conférences, …
- Routage de véhicules : planification d'itinéraires de véhicules (camions, trains, bateaux, avions, …) pour déplacer des marchandises et/ou des passagers vers plusieurs destinations à l'aide d'outils de cartographie connus …
- Conditionnement de bacs : remplir des conteneurs, des camions, des navires et des entrepôts de stockage avec des articles, mais également emballer des informations sur des ressources informatiques, comme dans le Cloud computing …
- Ordonnancement de l'atelier : planification des lignes d'assemblage des voitures, planification de la file d'attente des machines, planification des tâches de la main-d'œuvre, …

• Découpe de stock : minimiser les déchets lors de la découpe de papier, d'acier, de moquette, …

• Programmation sportive : planification des matchs et des horaires d'entraînement pour les ligues de football, les ligues de baseball, …

• Optimisation financière : optimisation du portefeuille d'investissement, répartition des risques, …

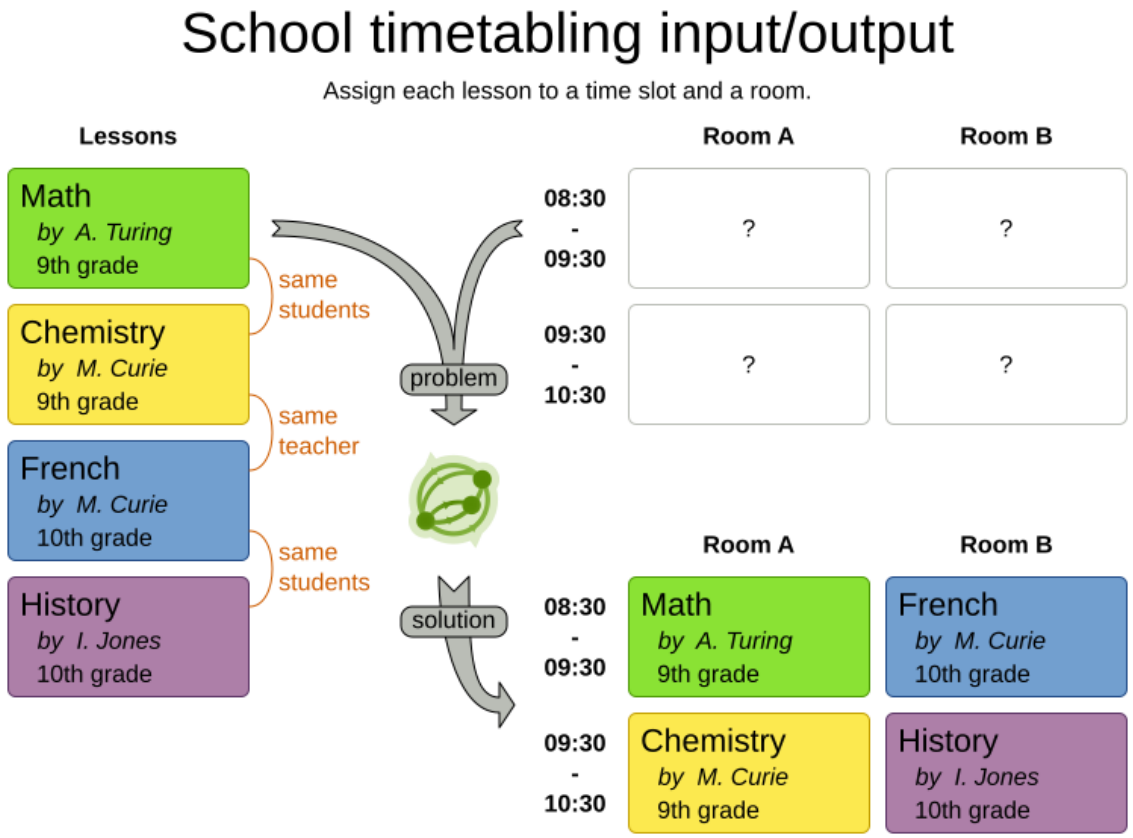


figure (3.1) : exemple du solver de contrainte optaplanner.

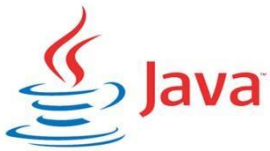
2-2- Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de

production de Logiciels libre qui soit extensible, universel et Polyvalent, en s'appuyant principalement sur Java.

Son objectif est de produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation (notamment environnement de développement

intégré et Framework) mais aussi d'AGL recouvrant modélisation, conception, test, gestion de Java est un langage interprété, ce qui signifie qu'un programme compilé n'est pas directement exécutable par le système d'exploitation mais il doit être interprété par un autre programme, qu'on appelle interpréteur.

Un programmeur Java écrit son code source, sous la forme de classes, dans des fichiers dont l'extension est .java .



Ce code source est alors compilé par le compilateur java en un langage appelé bytecode et enregistre le résultat dans un fichier dont l'extension est .class. Le byecode ainsi obtenu n'est pas directement utilisable. Il doit être interprété par la machine virtuelle de Java qui transforme alors le code compilé en code machine compréhensible par le système d'exploitation. C'est la raison pour laquelle Java est un langage portable : le bytecode reste le même quel que soit l'environnement d'exécution.

Dans ce chapitre‘, nous décrivons un problème général de confection d'horaires non cycliques en nous aidant des travaux de Jaumard et al. [13] et de Feuilleton [26] sur les unités de soins infirmiers de l'Hôpital Le problème décrit est assez général pour pouvoir représenter chacun des contextes particuliers cités plus haut.

3-Présentation du problème de planning

3-1-Formulation du problème

3-1-1-Le personnel

Il s'agit ici uniquement des infirmiers ou infirmières. Les problèmes étudiés ne concernent qu'un seul type de personnel à la fois bien qu'il soit tout à fait envisageable d'utiliser la présente approche pour traiter des problèmes mixtes. On aura donc des problèmes pour le personnel infirmier Voici la liste des attributs du personnel que HIBISC (Horaire d'Infirmières Basé Sur les Contraintes) prend en compte (ils suffisent pour exprimer l'ensemble des règles rencontrées jusqu'à présent.

- l'ancienneté

- le niveau de compétence,

- le statut temps complet partiel (exprimé par la charge de travail).

3-1-2- Les tâches

Une tâche est une période de travail indivisible au cours d'une journée pouvant être accomplie par une seule personne. Elle est caractérisée par un symbole , une heure de début , une heure de fin et éventuellement un nombre d'heure payée (s'il s'agit d'une activité payée) .

L'ensemble des tâches comprend, en Plus des quarts de travail, plusieurs activités reliées au travail qui nécessitent également d'être planifiées. Il S'agit concrètement de cours et de réunions. Ce sont des tâches plus rares que les quarts de travail et qui interviennent a peu près une fois par semaine.

Remarque 1 : dans la suite du mémoire on utilisera le terme quart pour désigner une tache . on parlera également de quart off pour un jour de congé et de quart Vac pour un jour de vacance

Les types de tâches. Un type de tâche est un sous ensemble de tâches ayant une caractéristique commune . HIBISC US (Horaire d'Infirmières Basé Sur les Contraintes) permet à l'utilisateur de définir n'importe quel type de tâche. La définition de ces types est inclus dans les données. On parlera par exemple du type nuit pour designer l'ensemble des quarts de nuit. Dans les salles d'urgence [20, 7] , on a besoin un type suite de soin

3-1-3- L'horizon de planification

L'horizon de planification est la période du calendrier pour laquelle un horaire de travail doit être conçu. Il est donc caractérisé par une date de début et une date de fin. Chaque jour de l'horizon, plusieurs tâches peuvent être planifiées (remarque : en général, les quarts de nuit sont considérés comme des tâches rattachées au jour précédant la nuit) . On appelle période de l'horizon tout ensemble de jours non forcément contigus. Une semaine peut désigner selon le contexte une suite de sept jours allant du lundi au lundi ou du dimanche au dimanche.

Les week-ends. Les week-ends sont des périodes particulières débutant à une certaine heure du jeudi ou du vendredi et se terminant à un certain hème du mardi matin. Ces heures peuvent varier selon que le vendredi ou le dimanche est férié ou non.

On ne peut donc pas définir un week-end comme une période. mais plutôt comme un ensemble de tâches (toutes celles qui couvrent la période du week-end). On dit qu'un week-end est travaillé par un individu I si et seulement si I effectue l'une des tâches reliées au week-end.

3-1-4- Les règle

Cette section contient l'ensemble des règles prises en compte complètement ou partiellement par H I B I S C U S. Ces règles ne sont jamais utilisées toutes au même temps dans un contexte donne. Certaines d'entre elles peuvent donc être contradictoires ou Légèrement redondantes.

3-1-4-1 Définitions et notations

Séquences de quarts de même type. Une séquence de quels de type T est une suite de quarts consécutifs de type T telle que le quart précèdent et le quart suivant, s'ils existent, ne sont pas de type T.

Les patrons. Un patron est une suite de type de quarts. On différencie les patrons de jours sur une période où chaque type du patron correspond a une journée de la période et les patrons de séquences qui décrivent des suites d'enchaînements de quarts autorisés. Pour représenter un patron, on écrira les nom des types dont il est composé séparés par le signe

3-1-4-2- Satisfaction de la demande [DEM]

Ces règles assurent que suffisamment de quarts sont pourvus tout au long de la période de planification. Elles visent à obtenir une couverture minimale ou désirée. Afin de pouvoir exprimer la demande dans le plus grand nombre de contextes, celle-ci peut être définie chaque jour pour un type de quarts T quelconque et pour tout sous-ensemble du personnel. Elle définit une valeur cible ou des bornes sur l'effectif du personnel concerné effectuant un quart de type T.

Exemple 1 : les jours de week-ends l'effectif du personnel infirmier travaillant les quarts de nuit doit être compris entre trois et cinq.

3-1-4-3 Les disponibilités

Chaque membre du personnel, en fonction de ses qualifications, de son statut régulier ou termes partiel, de la législation en vigueur et de ses choix de vacances, n'est pas disponible en tout temps. Les règles de disponibilité permettent au personnel de définir des horaires flexibles.

Pré-affectations et affectations interdites

Certaines portions d'horaires peuvent être prédéterminées (pré-affectations) ou partiellement déterminées (affectations interdites) . C'est souvent le cas du personnel à temps partiel qui travaille souvent dans plusieurs institutions. Les jours de vacances sont aussi traités comme des pré-affectations dans la plupart des contextes.

Exemple 2 :l'infirmier X n'est disponible que pendant les deux premières semaines .

Exemple 3 : l'infirmier Y et disponible que pour les quarts du jour.

-Propositions pour les vacances

Dans le cas où les vacances ne sont pas des pré-affectations. le personnel peut spécifier un ensemble de jours candidats ainsi qu'un nombre minimum et un nombre maximum de jours de vacances à choisir parmi ces jours candidats.

Exemple 4 : L'infirmière doit prendre entre cinq et dix jours de vacances au cours des deux dernières semaines de l'horizon.

3-1-4-4- Respect de la charge de travail

Suivant les contextes, la charge spécifiée dans les contrats de travail peut s'exprime mer en heures de travail ou en nombre de quarts. Dans tous les cas, cette règle impose un minimum et un maximum à la charge réelle. Il est aussi possible de répartir la charge quasi-uniformément ou selon une distribution voulue en définissant des charges sur des sous-périodes de l'horizon.

Exemple 5: l'infirmier X doit faire cinq quarts D, cinq quarts E et trois quarts N au cour de l'horizon.

Exemple 6 : l'infirmier Y doit travailler exactement 80 heures tous les deux semaines.

3-1-4-5- Règles ergonomiques

C'est la catégorie de règles la plus fournie et la plus hétérogène. Elle contient toute règle visant à maintenir un niveau de qualité pour les horaires individuels (voir les critères de Warner [9] , Certaines d'entre elles peuvent être obligatoires, d'autres, laissées au choix de chacun des membres du personnel.

- Patrons de jours sur une période donnée

Certaines périodes particulières de l'horizon (les week-ends, les jours fériés) doivent observer certains patrons. On dit qu'une période observe un patron (les deux ayant obligatoirement la même taille) si et seulement si chaque jour de la période a un quart du type indiqué à la même position dans le patron la figure 2.1 présent un horaire dans le quelle les périodes grisée satisfont le patron J/M/N

J : jour.

M : matinée.

N : nuit.

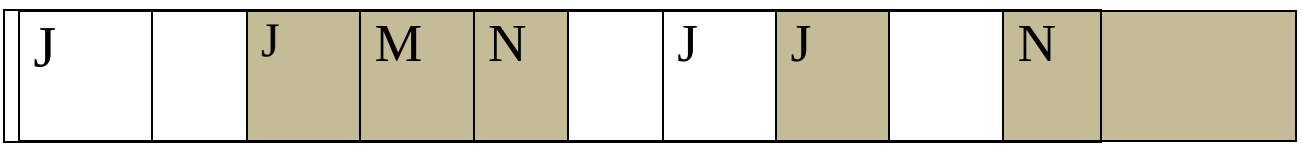


Figure 3.1 - Exemple l'horaire satisfaisant le patron J/M/N sur les périodes grisées.

Exemple 7: le personnel préfère ne pas avoir a travailler un seul des deux jours du week-end

- **Patrons de séquences**

L'enchaînement de certaines activités est interdit ou incommode pour le personnel. Les patrons de séquences décrivent les successions possibles des activités sur l'horizon.

En pratique, on utilise surtout des patrons de séquences binaires et ternaires. Soit T = {T_1 , ... , T_n} un ensemble de n types de quart formant une partition de l'ensemble des quarts. Un horaire quelconque peut alors être divise en une succession {S_1 , ... ,S_m) de séquences de types de T. Cet horaire est valide par rapport à un ensemble Pl de patrons de longueur l autorisés si et seulement si pour toute sous-séquence de (S_1, ... , S_m) de taille l , il existe un patron de Pl correspondant à cette sous-séquence (c'est-à-dire pour lequel chaque type correspond dans l'ordre au type de celle-ci).

exemple, étant donné un ensemble de patrons ternaires autorisés incluant $T_2/T_5/T_3$ et $T_5/T_3/T_1$, l'horaire de la figure 2.2 est valide.

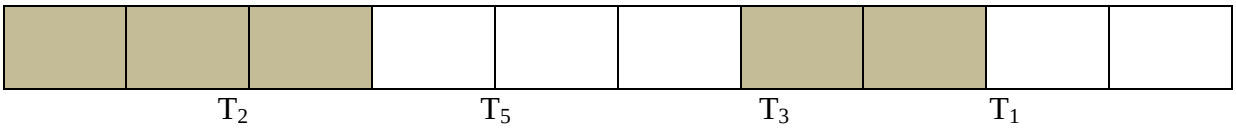


Figure 3.2 Horaire découpé en séquences de types.

- Nombre d'affectations consécutives

La longueur des séquences des activités d'un certain type peut être bornée.

Exemple 8 : une infermière ne doit pas travailler plus de sept jours consécutifs.

-Week-ends consécutifs

Les week-ends font souvent l'objet de règles de distribution spéciales. Ces règles spécifient un minimum et un maximum pour la longueur des séquences de week-ends travaillés et non travaillés de chaque membre du personnel.

Exemple 9 : une infermière X ne doit. pas travailler au cours de deux weekends consécutif.

- **Séquences consécutives :** Une séquence vérifiant certaines propriétés (taille, patrons, etc...) doit être immédiatement suivie par une autre séquence vérifiant d'autres propriétés.

3-1-4-7 - Distribution des quarts

Cette règle assure que pour chacun des membres du personnel, le nombre de quarts importons n'est pas trop éloigné du quota en le confinant entre deux bornes.

Exemple 10 : une infermière ne doit pas effectuer plus de cinq quarts de week-end par

Horizon.

Distribution entre plusieurs types de quarts

Le personnel peut spécifier une distribution souhaitable entre plusieurs types de quarts (T q) 1< q< n disjoints. Dans le cas ou elle est souple, cette règle vise â équilibrer les écarts en valeur absolue par rapport aux proportions de chaque type de quart.

Exemple 11 : Le rapport entre le nombre de quarts cubes et le nombre de quarts chocs des horaires individuels doit se rapprocher le plus passible du rapport entre le nombre total de quarts cubes et le nombre Total de quarts chocs disponibles au cours de la période de planification.

Exemple 12 : L'infirmière Y doit effectuer an cours de l'horizon deux fois plus de quarts de soir que de quais de nuit.

- **Préférences et aversions**

Le personnel peut spécifier individuellement ses préférences et ses aversions en termes d'affectations désirables, neutres et non désirables. Cette règle est fondamentalement souple. On cherchera à maximiser les préférences et à minimiser les aversions.

3-1-4-6- Règles d'équité

Les règles d'équité servent à garantir à ce qu'aucun membre du personnel ne se trouve avantagé par rapport aux autres relativement à la satisfaction d'un critère souple. Il peut s'agir des préférences et aversions ou de n'importe quelle autre règle souple.

3-2- Particularité des unités de soins infirmiers

3-2-1 Définition de la demande

Dans ce contexte, une journée est divisée en périodes de quota (intervalles de temps). Étant donné un ensemble de quarts, les périodes sont choisies maximales de telle sorte que

l'ensemble des quarts simultanés ne change pas au cours d'une période. De ce fait, une période de quota est caractérisée par l'ensemble des quarts qui la couvrent .

Exemple 1.8 : l'effectif du personnel de niveau de compétence 1 travaillant pendant

La période p3 doit être compris entre trois et cinq.

3-2-2- Vacances et week-ends

Dans ce contexte, les vacances sont des congés payés (un jour de vacances compte pour une charge de 8 heures)

- Exception pour la règle des week-ends non brisés

Les week-ends doivent, à quelques exceptions près, être non brisés. Cette règle est appliquée a tous les week-ends de l'horizon excepté ceux qui délimitent des périodes de vacances. Le tableau 3.1 répertorie les patrons congé / travail possibles en fonction de la fin de la semaine précédente et le début de la semaine suivante. On parle d'une fin (resp. d'un début) de semaine de travail si le dernier (resp. premier) jour de la semaine différent d'un congé est un jour travaillé. On parle d'une fin ou d'un début de semaine de vacances si ce jour est un jour de vacances.

Tableau 3.1 Week-ends brisés et non brisés autorisés

Fin de la semaine précédente	week-end		Début de la semaine suivante
	Samedi	Dimanche	
vacances	congé	travail	travail
vacances	congé	conge	travail
travail	congé	conge	vacances
travail	travail	congé	vacances
travail	congé	congé	travail
travail	travail	travail	travail
vacances	congé	congé	vacances

3-2-3- Exception pour la règle sur les séquences de week-ends

On retrouve la règle sur les séquences de week-ends travaillés et non travaillés mais avec Mme exception pour les week-ends appartenant à une période de vacances. Une période de vacances caractérise un ensemble de jouis consécutifs de conges et de vacances. Un week-end compris entre un début Ile semaine de vacances et une fin de semaine de vacances ne doit pas être travaillé. Après la période de vacances cependant, le rythme des séquences doit reprendre comme s'il n'avait pas été interrompu. La figure 2.4 est un exemple d'alternance de week-ends travaillés et non travaillés valide en période de vacances.

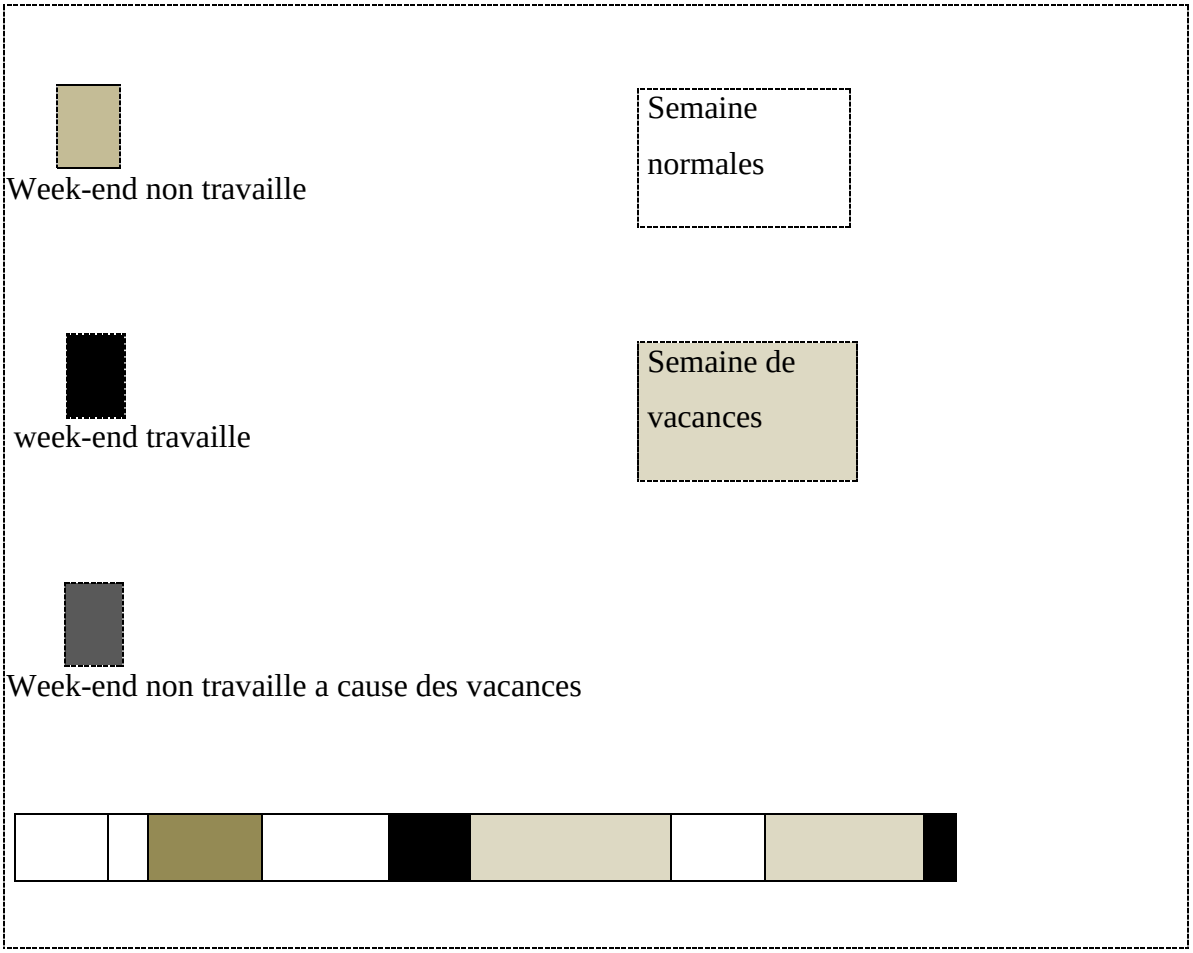


Figure 3.4 Conséquences des vacances sur l'enchaînement des week-ends

Tableaux 3.2 de description des taches de l'unité de d'analyse des infermières

Symbol	Durée	Debut	Fin	Period
matin	8H	8:30	15:30	jour
Le soir	8H	15:30	23:30	soir
Chef des infermier	SH	11:30	19:3 tJ	jour
La journèe	12H	7 :30	19 :30	**jour**
nuit	12H	11 :30	6 :30	**soir**

4- Modélisation du problème

4-1- Variables de décision, paramètres et domaines

Comme on a vu précédemment, le CSP est généralement décrit par la vue infermière-jour, la vue infirmier-tâche ou la vue infirmier-intervalle de temps et une vue infirmier-modèle de poste.

Une vue infirmier-jour est une représentation directe d'une planification bidimensionnelle. Par conséquent, les variables de décision peuvent être dénies pour chaque infermière chaque jour comme v_{ij} où $1 \leq i \leq N$ indique l'ensemble des infermières et $1 \leq j \leq P$ indique les jours dans une période de planification.

Les domaines de ces variables comprennent les postes en service et les postes libres. Les postes en service peuvent inclure n'importe quel nombre de postes par jour, mais il est fréquent d'utiliser seulement les postes nuit/jour/matinée/et après midi. Ainsi, les variables de décision peuvent généralement prendre 10 valeurs ou plus, ce qui augmentent les heurs de calcul, une réduction de domaines de variables L'idée est de mettre toutes les valeurs des postes libres Dans la situation générale, quand il ya Z poste par jour, v_{ij} peut prendre Z + 1 valeurs possibles :

$$\mathbf{V_{iJ}} : \begin{cases} 1 : \text{l'infermier i et hor service le jour j} \\ \mathbf{2} : \text{l'nfermier i travaille la quart 1 le jour j} \\ . \\ . \\ . \\ . \\ . \\ \boldsymbol{z} : \text{l'nfermier i et hor service le jour j} \end{cases}$$

Le document donne un exemple avec quatre postes par jour. Les valeurs des postes libres sont réduites à une valeur

Le poste du matin, le poste la journée et le poste de nuit et l'après-midi, de sorte que la variable de décision aura quatre valeurs possibles :

$$\mathbf{V_{ij}} \begin{cases} 0 : \text{l'infermier i est hors service le jour j} \\ 1 : \text{l'infermier i travaille le poste de matin le jour j} \\ 2 : \text{l'infermier i travaille le poste de l'aprè midi le jour j} \\ 3 : \text{l'infermier i travaille le poste de nuit le jour j} \end{cases}$$

Pour les modèles binaires , les variables de décision peuvent être personnalisées pour être v_{ijk} où ij sont les mêmes indices que pour v_{ij} , et $1 \leq k \leq Z$ indique les Z postes possibles en un jour , pour Z = 3, v_{ijk} est binaire :

$$V_{ijk}\begin{cases}1 : \text{l'infermier i travaille le poste k le jour j.}\\ 0 : \text{autrement.}\end{cases}$$

4-2-Contraintes :

On a vu dans les dimensions du problème les différents types de contraintes et les sources de décision qui les produisent.

Par la suite on représente un planning de contraintes qui surviennent couramment avec le CSP :

1. Charge de travail des infermières (minimum / maximum).
2. Poste de travail identique consécutif (minimum / maximum / nombre exact).
3. Poste / jour de travail consécutif (minimum / maximum / nombre exact).
4. Niveaux de compétences et catégories des infermières.
5. Préférences ou exigences des infermières.
6. Jours libres des infermières (minimum / maximum / jours libre consécutifs).
7. Temps libre entre les postes de travail (minimum).
8. Affectations de type (s) de poste (maximum type de poste, les exigences pour chaque type de poste).
9. Congés et vacances (prévisible), par exemple, jours fériés, les congés annuels.
10. Week-end de travail (actif), par exemple, week-end complet.
11. Modèles de poste.
12. Contraintes entre les postes, par exemple les postes ne pouvant pas être attribués à une personne en même temps.
13. Exigences (de différents types) d'infermières ou la demande du personnel pour chaque poste (minimum / maximum / nombre exact).

4-3-Fonctions Objectif :

Généralement, avec des problèmes d'optimisation, nous trouvons des modèles qui utilisent les fonctions objectives standards, telles que celles des modèles de la programmation mathématique (MP). Dans d'autres modèles, on trouve les fonctions objectifs ou d'évaluation (target or évaluation functions) qui sont utilisées pour guider la génération de résultats ou d'évaluer les résultats.

L'objectif $\sum_{i=1}^{n}\sum_{j=F(x)} \boldsymbol{Pij, Xij} \rightarrow$**Min F(x),** où p_{ij} est l'affectation dès l'infermières i qui travaille sur le modèle de poste j , x_{ij} est la variable de décision pour la vue infermière-modèle de poste et F(i) est l'ensemble des modèles de poste réalisable pour l'infermière i , où le but est de minimiser l'affectation totale pour tous les infermières. Ceci est soumise à des contraintes où chaque infermière travaille exactement un seul modèle de poste réalisable et une demande pour les infermières est remplie pour chaque catégorie chaque jour et nuit.

Les captures de l'application du gestion du planning du personnel infirmier

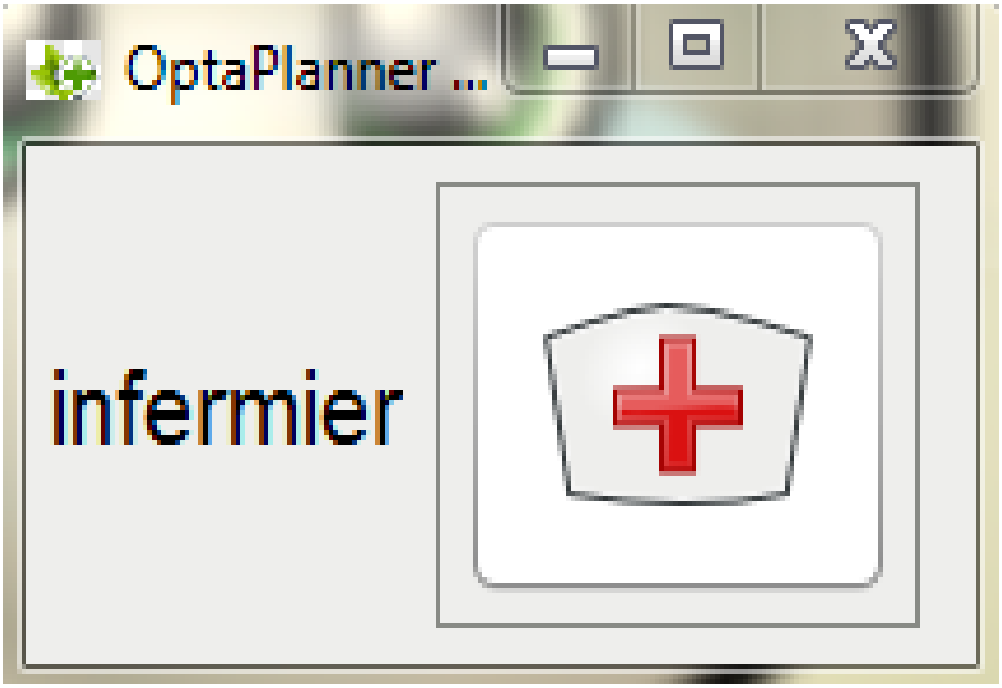


Figure 3.5 Exécution de l'application

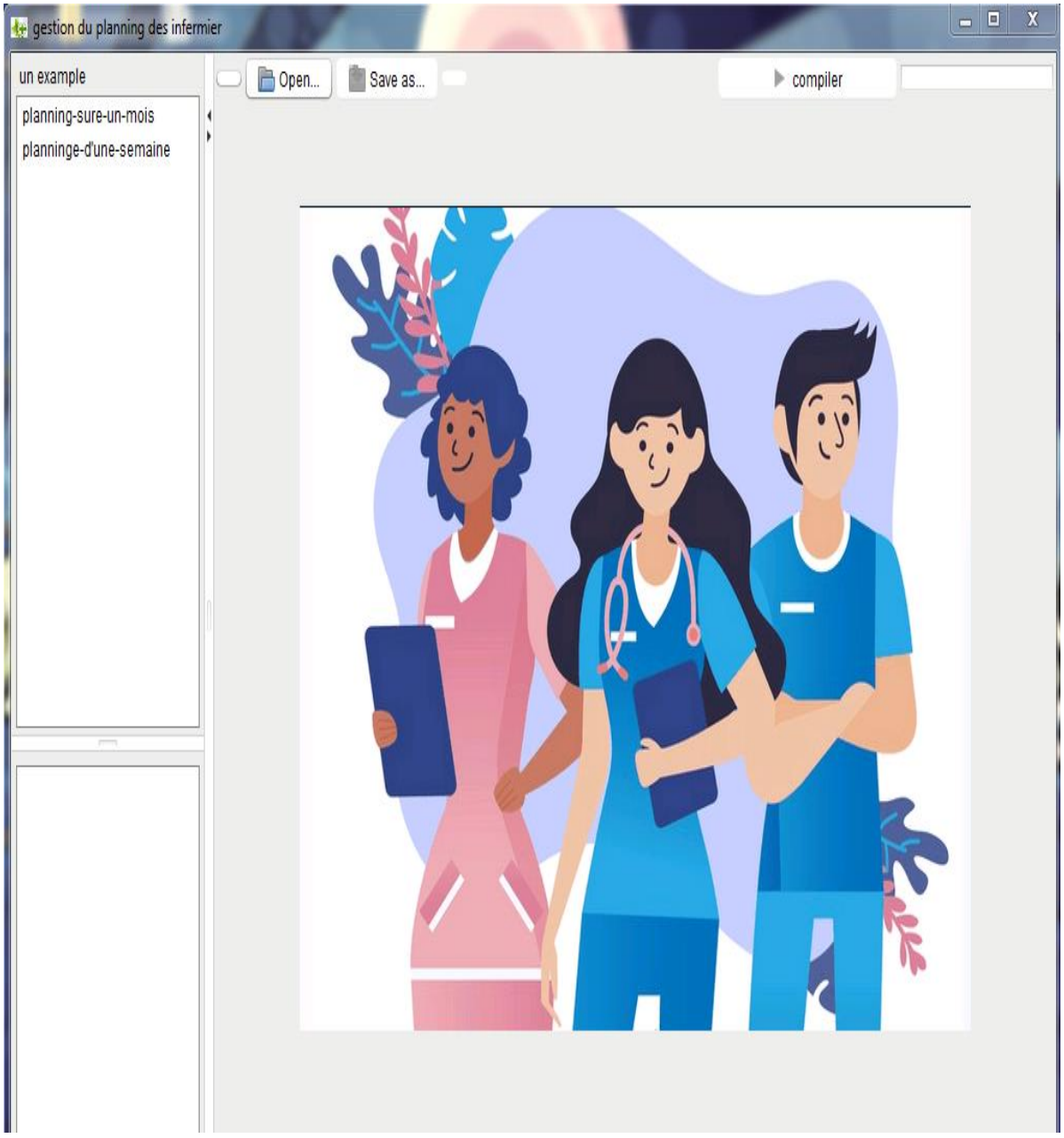


Figure 3.6 Interface de l'application .

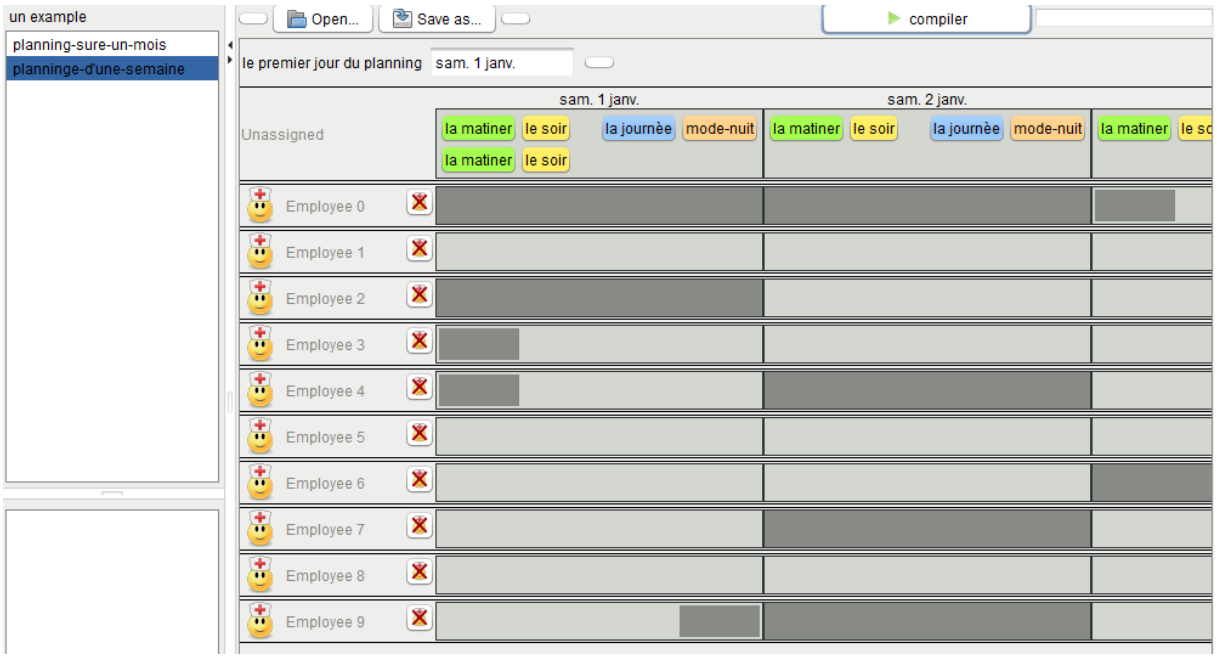


Figure 3.7 Planning Avon l'exécution

Après l'exécution on choisie le type de planning sur un horizon d'une semaine ou sure un horizon de 30 jours et on click sur compiler.

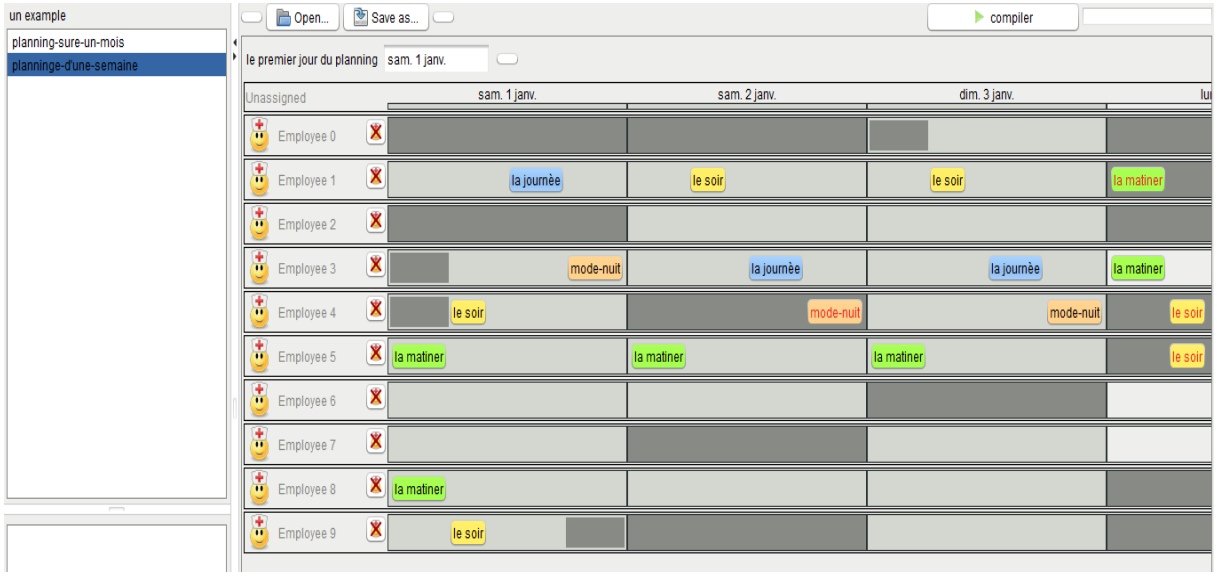


Figure 3.8 Planning après l'exécution

Conclusion générale

Conclusion générale :

Dans ce mémoire, nous avons présenté une méthode de confection d'horaires en milieu hospitalier on utilisant la programmation par contraintes. Les contributions de ce travail sont multiples et interviennent à plusieurs niveaux.

Nous avons également décrit un cadre général permettant de concevoir des stratégies de recherche adaptées à tout type de problème concret que l'on peut extraire du modèle. Nous avons discuté des différentes façons de décomposer le problème et nous avons proposé une structure modulaire originale pour nos heuristiques. Cette dernière contribution dépasse le cadre de la confection l'horaire et constitue un apport non négligeable pour la programmation par contraintes qui souffre d'un manque certain de littérature en ce qui concerne les heuristiques de recherche. Cette structure repose sur le concept d'heuristique d'initiation qui représente la brique de base de toute heuristique. nous avons fourni des heuristiques d'incitation pour deux types de contraintes très largement représentées dans les problèmes de confection d'horaires : les contraintes de cardinalité et les contraintes sommes.

Enfin, nous avons décrit et modélise avec. HIBISCUS (Horaire d'Infirmières Basé Sur les Contraintes) plusieurs contextes réels et hétérogènes que nous avons utilisés comme base de test pour comparer différentes stratégies de l'solution. Les résultats obtenus sont encouragent.

Références Bibliographiques

Référencés Bibliographiques

[1] Apt, K., Principleas of Constraint Programming, Cambridge University Press, 2003.

[2] Dechter, R., Constraint Processing, Morgan Kaufmann Publishers, 2003.

[3]Marriott, K. et P. Stuckey, Programming with Constraints: An Intro- duction, Boston, MIT Press, 1998.

[4] Régin, Generalized Arc Consistency for Global Cardinality Con- straints», dans Proc. of AAAI-96, p. 209–215, AAAI Press/MIT Press, 1996.

[5] Baptiste, P., C. Le Pape et W. Nuijten, Constraint-Based Scheduling, Kluwer Academic Publishers, 2001.

[6] Pesant, G., «A Regular Language Membership Constraint for Finite Se- quences of Variables», dans Proc. of CP'04, p. 482–495, Springer-Verlag LNCS 3258, 2004

[7] Milano, M. Constraint and Integer Programming: Toward a Unified Methodology, t. 27 de Operations Research/Computer Science Interfaces, Klu- wer, 2004

[8]H E Stiller, WPierskalla, et Goth Nurse scheduling using mathema- tical

programming. fipereions fiesearci, 24 :857-870, 1976

[9] D. II. warner. Sclieduling using personnel a Coording to nursing preference : A mathematical apprOaCh. Opcr‹itions Research, 24 :h42-856, 1976.

[10] H.Mayer Hof . solving rostring tasks by generic methodes for constraint optimisation international journal of FONDATION of computer scines 12(5) :671-693,2001

[11] F. Forget. Confection automatisé des horaires d'infermier dans une salle d'ur- gence.

[12] H. Beaulieu. Planification de l'horaire d'infermier dans une salle d'urgence

[13] B. Jaumard, F. Seniet, et T. Vovor. A generalized linear programming model for nurse scheduling. European Journal o Operational Research, 107 :1-18, 1998.

[14] A. Caprara, M. Monaci, et P. Toth. models and algorithrns for a staff scheduling problem (to appear). mathèmatical 2003.

[15] G. Thompson . A Simullated-annealing heuristic for shift using non- continuously aviable emploiyes. muffler Operational fiesenrcn, 23(3) :275— 258, 1990.

[16] G. B. Dantzig. A comment on eddie's traffic delays at toll bootlis. Operations

Aeseerch, 2 :339-341, 1954

[17] K. A. Dowland. Nurse scheduling with tabu search and strategic oscillation.

European Journdl o Gperafton Resedrch, 106(2-3) :3S3-407, 1998.

[18] I. Buzòn. La confection des horaires de travail des médecins rl'urgence résolue à l'aide de la recherche tabou..

[19] K. Heus. Gestion d,es pla.riotings in firiniers, Applicaiion des techniques de pro- gram nattun Our cutrotn/es.Thése de doctoral, Université ,Joseph Fourier Grenoblel, Mai 1996.

[20] S. J. Darmoni, A. Fajner, N. Nlahé, A. Leforestier, M. Vondracek, O. Stelian, ct LI. Ba,ldenweck. Horoplan: computer-assisted nurse scheduliIig usiIig contraint- based programming. Journ.‹if / the Society for Health System,s, 5 :41 -ñ4, 1995.

[21] Y. Caseau, P. Giullo, et E. Levenez. A deductive and object-oriented approach to a complex scheduling problem. In Thi,r‹l Internation,al Con fen.nr'e, DOOD'93, fiñ.oeniz, Arizona USA, December 6-8, 1998, Procer. lings, pages G7-80. Springer, 2020.

[22] S. Abdennadher et H. Schenker. Nurse scheduling using constraint program- ming. In ElP.tenth frrence on Iantiovative Applicn/tons Arti.for.i.ml Intelligence, Orland,o, Florida. Press, 2020.

[23] B. kL W. Cheng, J. H. II. Lee, ct J. C. K. Wu. A constraint-based nurse rostering system using a redundant modeling approach International, gonferenre on Tools with Arti facial intiligence, pages 140-148. Computer Society Press. 2018.

[24] a1. Saadie Planification de l'horaire des infirmière dans une salle d'urgence par a programmation par contraintes. , 2013.

[25] InTinie. Speedshi fl. http ://www.intimesoft.com/.

[26] W. Feuilletin. Génération automatique d'horaires d'infirmière à l'aide de la programmation par contraintes. Rapport Technique C7PQMR PO2000-32-X, Centre de Recherche sur les Transports, Canada, Octobre 2020.

[27] L. hI. Rousseau, G. Pesant, et M. Gendreau. A hybrid algorithm to solve a physi- ciairostering problem. In Second, WorkshoQ OF /HtCyrttifGn AI OTtd OR Tech- niques in Constraint Proqrammiiig for Combinotoyial Oytimizdtion Problems

[28] H. Lapierre Hole. Solving rostering tasks by generic methods for constraint optimisation Internationall Journal o Foundat.ions o J' Computer Science, 12(5) :G71 G93, 2021.

[29] P. Labit. IR.IS : Ainélioration d'une métliode de génération de colonnes pour la confection d'horaires d'infirmiéres. lémoire de maitrise, Ecole Polytechnique de montréal. 2013

[30] Yebbah. L'homme. Accelerating filt hering techniques for numeric CSPs 139 :109 132,

2020.

[31] R. N. Burne ct M. W. Carter. Work force size and single shits schedules with variable demands. *Management* Science, 31(5) :599-607, 198c

[32] H. E. miller, W. P. Pierskalla, et G. J. Roth. Nurse scheduling using mathema- tical programming. fipere/ions fiesearc/i, 24 :857-870, 1976.

[33] Gendreau et. Pesant. The pattern constraint (in prepartation). 2003.

Résume

Ce mémoire de maitrise présente un modèle de programmation par contraintes ainsi que des stratégies de recherche permettant de formuler et de résoudre des problèmes de confection d'horaires en milieu hospitalier (infirmier). Ce problème a déjà été largement étudié et beaucoup d'approches différentes ont été développées a cours des années, mais elles sont souvent trop liées à leur contexte de développement. Concevoir une méthode robuste pouvant s'appliquer à différents contextes reste donc un défi à relever.

Nous décrivons une méthode flexible et plutôt générale de programmation par contraintes qui intègre des heuristiques de recherche de façon originale. Nous comparons quelques stratégies sur un ensemble de données hétérogènes. Dans le cadre de ce mémoire, nous nous sommes concentrés sur la satisfaction plutôt que l'optimisation, mais nous indiquons les directions à suivre pour transformer notre méthode en une méthode d'optimisation Des horaires réalisables

Mots Clés : confection d'horaire, Milieu hospitalier (infirmier), programmation par contraintes, heuristiques de recherche

ملخص

تقدم أطروحة الماجستير هذه نموذجًا لبرمجة القيد بالإضافة إلى استراتيجيات البحث لصياغة وحل مشاكل الجدولة في المستشفيات. تمت بالفعل دراسة هذه المشكلة على نطاق واسع وتم تطوير العديد من الأساليب المختلفة على مر السنين ، لكنها غالبًا ما تكون مرتبطة جدًا بسياقها التنموي. لذلك ، فإن تصميم طريقة قوية يمكن تطبيقها على سياقات مختلفة لا يزال يمثل تحديًا.

نحن نصف طريقة مرنة وعامة إلى حد ما لبرمجة القيد التي تدمج استدلال البحث بطريقة أصلية. نقارن بعض الاستراتيجيات على مجموعة بيانات غير متجانسة. لغرض هذه الأطروحة ، ركزنا على الرضا بدلاً من التحسين ، لكننا نشير إلى التوجيهات التي يجب اتباعها لتحويل طريقتنا إلى طريقة تحسين للجداول الزمنية الممكنة.

الكلمات المفتاحية: الجدولة ، إعداد المستشفى ، برمجة القيد ، استدلال البحث .

Abstract

This master's thesis presents a constraint programming model as well as research strategies for formulating and solving scheduling problems in hospitals. This problem has already been widely studied and many different approaches have been developed over the years, but they are often too tied to their development context. Designing a robust method that can be applied to different contexts therefore remains a challenge.

We describe a flexible and rather general method of constraint programming which integrates search heuristics in an original way. We compare some strategies on a heterogeneous data set. For the purpose of this thesis, we have focused on satisfaction rather than optimization, but we indicate the directions to follow to transform our method into an optimization method of feasible schedules.

Keywords: scheduling , hospital setting , constraint programming , search heuristics.