Contributions à la résolution des WCSP et approches déclaratives pour la fouille de données

Samir Loudni

HABILITATION

À DIRIGER DES RECHERCHES

présentée à

L'Université Caen Normandie

École doctorale SIMEM

Spécialité : Informatique et applications

par

Samir LOUDNI

Soutenue publiquement le *5 Octobre 2016*

Contributions à la résolution des WCSP et approches déclaratives pour la fouille de données

Simon de Givry	Chargé de recherches, INRA Toulouse	(Rapporteur)
Christine Solnon	Professeur, INSA Lyon	(Rapportrice)
Christel Vrain	Professeur, Université d'Orléans	(Rapportrice)
Patrice Boizumault	Professeur, Université Caen Normandie	(Examinateur)
Bruno Crémilleux	Professeur, Université Caen Normandie	(Examinateur)
Narendra Jussien	Professeur, TELECOM LILLE	(Examinateur)
Lakhdar Sais	Professeur, Université d'Artois Lens	(Examinateur)
Alexandre Termier	Professeur, Université Rennes 1	(Examinateur)
Gérard Verfaillie	Directeur de Recherche, ONERA Toulouse	(Invité)

Table des matières

Chapitre 1

Introduction

1.1 Parcours de recherche

Mes activités de recherche concernent la programmation par contraintes (PPC), la résolution de problèmes d'optimisation sous-contraintes et la fouille de données. Elles s'étendent sur une douzaine d'années depuis ma thèse préparée à l'École des Mines de Nantes et soutenue en 2002 à l'université de Nantes, et se sont poursuivies au sein du GREYC. Ces activités s'articulent autour de trois axes principaux :

1. Conception et mise en œuvre de méthodes hybrides pour la résolution de problèmes d'optimisation sous contraintes, faisant coopérer méthodes complètes procédant par "séparation et propagation" avec des approches incomplètes procédant par recherche locale.
2. Relaxation de contraintes globales et fonctions de coût décomposables, permettant de prendre en compte les préférences formulées par l'utilisateur et de quantifier la violation de manière fine selon le niveau d'insatisfaction et la nature de la violation.
3. Étude des apports de la programmation par contraintes pour la fouille de données, avec un aspect original consistant à introduire des techniques utilisées en fouille de données pour améliorer les encodages PPC pour la fouille de grands ensembles de données.

Après avoir suivi le DEA Combinatoire, Parallélisme et Modélisation aléatoire de l'Université de Picardie Jules Vernes d'Amiens, j'ai eu le plaisir d'être accueilli à l'EMN dans le cadre du projet EOLE [1] pour une thèse encadrée par Patrice Boizumault. Le sujet portait sur la conception et la mise en œuvre de méthodes hybrides pour la résolution de problèmes d'optimisation sous contraintes dans un contexte *anytime*, en utilisant la programmation par contraintes. Dans un tel contexte, les meilleures méthodes sont celles capables de produire très vite de (très) bonnes solutions et de les améliorer au fur et à mesure du temps. Dans ce cadre, j'ai apporté deux contributions importantes : (i) une méthode de recherche hybride `VNS`/`LDS`+`CP` [Loudni et Boizumault, 2003] ayant de très bonnes propriétés d'un point de vue anytime, et (ii) sa mise en œuvre avec succès pour résoudre un problème de réservation en-ligne de connexions dans les réseaux ATM [2] pour France Telecom R&D Lannion [Loudni *et al.*, 2006].

1. Le projet EOLE (Environnement d'Optimisation en LignE dédié télécom), soutenu par le Réseau National de Recherche en Télécommunications (RNRT), a réuni les partenaires THALES Research & Technology, ONERA Centre de Toulouse, Cosytec, Bouygues/DTN, Ecole des Mines de Nantes et France Telecom R&D Lannion.
2. Asynchronous Transfer Model.

Lors de mon arrivé au GREYC comme maître de conférences en 2003, j'ai naturellement intégré l'ancienne équipe Algorithmique, autour des travaux menés dans le thème intitulé "Contraintes et Graphes". Ce thème s'intéresse aux graphes, hypergraphes et aux problèmes de satisfaction de contraintes (CSP). J'y ai poursuivi des activités de recherche sur les méthodes de résolution et d'optimisation dans le cadre des *Weighted Constraint Satisfaction Problem* (WCSP) (appelé aussi réseaux de fonctions de coût), un formalisme puissant et très souple permettant la modélisation de nombreux problèmes d'optimisation combinatoire sous contraintes. Dans ce contexte, Nicolas Levasseur a proposé dans sa thèse de nouvelles heuristiques génériques, couvrant à la fois la résolution par des méthodes arborescentes et par des méthodes à voisinages variables. Concernant la recherche arborescente, en s'appuyant sur l'historique des solutions rencontrées durant la recherche, il a proposé de nouvelles heuristiques de sélection de valeur et de variable pertinentes. Concernant la méthode à voisinages variables `VNS/LDS+CP`, il a proposé d'intégrer la topologie du réseau de contraintes et les coûts dans le choix de variables à réaffecter, conduisant à des améliorations de performance très visibles sur les jeux de données testés.

Dans le cas des réseaux de contraintes, la notion de *contrainte globale* a été fondamentale dans le succès de la PPC, tant d'un point de vue modélisation (en synthétisant élégamment des ensembles de contraintes) que résolution (en mettant à profit des algorithmes de filtrage hérités d'autres domaines – généralement la Recherche Opérationnelle). Lorsqu'on s'attaque à des problèmes réels, une difficulté récurrente apparaît dans la nature sur-contrainte des problèmes traités. Il est alors nécessaire d'être capable de relaxer certaines contraintes pour trouver une solution. Dans la lignée des travaux sur la relaxation de contraintes globales initiés par Willem-Jan van Hoeve, Jean-Philippe Métivier a défini dans sa thèse un cadre unifié de relaxations pour différentes contraintes globales existantes permettant de prendre en compte de manière fine des préférences ou des coûts de violation de ces contraintes. Il a ensuite appliqué ce cadre à la résolution de problèmes d'emplois du temps de personnel hospitalier (NRPs : *Nurse Rostering Problems*, benchmarks de la communauté timetabling RO), archétype des problèmes sur-contraints complexes à modéliser.

Enfin, j'ai étudié l'intérêt d'exploiter la structure du graphe sous-jacent à un réseau de contraintes dans la résolution des problèmes d'optimisation sous contraintes. Ma motivation vient du fait que plusieurs problèmes et benchmarks rencontrés lors de mes précédents travaux avaient un graphe fortement structuré (présence de groupes de variables formant des *clusters*, ces groupes n'étant eux-mêmes pas ou peu connectés entre eux). Les travaux menés jusqu'alors exploitaient la structure du graphe de contraintes uniquement dans le cadre des méthodes de recherche complète. L'idée repose sur l'exploitation d'une décomposition du problème en arbre de clusters (aussi appelée *décomposition arborescente*). C'est, à ma connaissance, une approche qui n'a pas été explorée dans les méthodes de recherche locale. C'est dans ce contexte de s'inscrivent les travaux de Mathieu Fontaine. Il a proposé dans sa thèse un schéma générique de recherche locale qui exploite la décomposition arborescente pour guider efficacement l'exploration de l'espace de recherche. Ce schéma, noté `DGVNS` [3], utilise la décomposition arborescente comme une carte du problème afin de construire des voisinages pertinents. Il a également étudié l'intérêt d'exploiter une bonne décomposition arborescente afin de renforcer la qualité des voisinages explorés par `DGVNS`. Ce travail s'est ensuite poursuivi dans le cadre de la thèse en cotutelle de Abdelkader Ouali, en proposant des stratégies parallèles pour `DGVNS`. En effet, les calculateurs actuels qui sont naturellement multi-processeurs et multi-cœurs, fournissent une énorme puissance de calcul.

3. Decomposition Guided Variable Neighborhood Search.

L'exploitation de ces capacités de calcul sur ces architectures permet d'attaquer des problèmes combinatoires de grande taille. Dans ce cadre, Abdelkader Ouali a proposé différents schémas de parallélisation de la méthode `DGVNS`, permettant d'explorer tous les clusters de la décomposition arborescente en parallèle. Ces schémas reposent sur une architecture maître-esclave et diffèrent essentiellement par la façon dont la communication s'effectue entre le maître et les esclaves (synchrone ou asynchrone) et par la fréquence des solutions échangées (communiquer les meilleures solutions intermédiaires au cours de la recherche ou seulement à la fin pour identifier la meilleure solution globale).

La création de l'équipe CoDaG [4] dans le cadre du quadriennal 2012-2015, suite à la fusion entre le thème "Contraintes et Graphes" de l'ancienne équipe Algorithmique et de la partie "Fouille de données" de l'ancienne équipe "Données, Document, Langue", m'a amené à m'intéresser aux liens entre la fouille de données et la programmation par contraintes. Ce contexte a été particulièrement enrichissant et stimulant pour développer des activités de recherche sur les apports de la programmation par contraintes pour la fouille de données. Un premier travail, réalisé dans le cadre du post-doc de Jean-Philippe Métivier et financé par le projet CPER Basse-Normandie, a porté sur la conception d'un langage à base de contraintes permettant d'exprimer de manière déclarative l'extraction d'ensembles de motifs (i.e. des motifs dont l'intérêt dépend des autres motifs). Ce premier travail m'a permis de me rendre compte du manque de flexibilité et de souplesse du cadre actuel où les contraintes sont usuellement rigides et avec de nombreux paramètres difficiles à régler. Pour lever ce verrou, Willy Ugarte a étudié dans sa thèse l'introduction de la souplesse dans le processus de modélisation et d'extraction de motifs. Les principaux résultats portent sur (1) la définition d'un cadre général mettant en œuvre les contraintes souples de seuil dans un extracteur de motifs en utilisant la relaxation de contraintes en PPC, (2) l'extraction des motifs Pareto-optimaux (appelés skypatterns) ainsi que leurs versions relâchées, et (3) la construction du cube des skypatterns qui permet d'étudier les différents ensembles de skypatterns selon les combinaisons possibles d'un ensemble de mesures.

1.2 Projets collaboratifs

Dans cette section, je présente les travaux de recherche menés dans le cadre de projets labellisés auxquels j'ai participé en tant que chercheur et/ou coordinateur pour le GREYC ainsi que dans le cadre de la collaboration entre l'équipe CoDaG et le LITIO de l'université d'Oran 1.

ANR Blanc FICOLOFO [5] (Jan. 2011 - Oct. 2014). *Filtrage par cohérences locales fortes pour les réseaux de fonctions de coûts.* Participants : INRA Toulouse, LIRMM, IRIT, GREYC. **J'ai été le coordinateur pour le GREYC**.

Le projet a eu pour objectif de construire de nouvelles cohérences locales plus fortes et des fonctions de coût globales pour les réseaux de contraintes pondérés, afin d'accroître davantage l'expressivité du formalisme WCSP pour la modélisation et la résolution de nouveaux problèmes réels d'optimisation de grande taille.

Contributions : La première approche du projet était de généraliser les contraintes globales aux fonctions de coût. En poursuivant sur cette base, nous avons introduit la notion de décomposition de fonctions de coût globales et nous avons montré comment les contraintes globales

4. COntraintes, DAta mining et Graphes.
5. `http://costfunction.org/ficolofo`

décomposables pouvaient être relâchées sous la forme de fonctions de coût globales décomposables, permettant une intégration simplifiée des contraintes globales (relaxées) dans les réseaux de fonctions de coût. Ainsi, une large famille de fonctions de coût globales, décomposables, a été analysée et implémentée dans le solveur état de l'art `toulbar2`[6].

ANR Blanc Hybride[7] **(Déc. 2011 – Nov. 2016)**. *Hybridation de la fouille de données et du traitement automatique des langues.* Participants : INRIA-NGE, MoDyCo, Orphanet INSERM, GREYC.

L'utilisation de ressources textuelles est un élément incontournable pour la découverte d'information, tout particulièrement dans le domaine bio-médical. Pour répondre à cette problématique, le projet vise à combiner des méthodes issues de la fouille de données et du traitement du langage naturel afin d'extraire de l'information et de la connaissance dans des domaines spécialisés. La cible applicative est l'analyse de documents scientifiques sur les maladies rares.

Contributions : Grâce à l'impulsion donnée par le projet ANR Hybride, j'ai pu explorer une autre direction de recherche portant sur l'extension du cadre déclaratif de la PPC pour la modélisation et l'extraction de motifs séquentiels. En nous appuyons sur la notion de motif séquentiel avec *wildcards*[8], nous avons proposé un premier modèle CSP pour l'extraction de ces motifs sous contraintes multiples dans une base de séquences.

Lors de ces travaux, nous avons pu constater la relative faiblesse des approches déclaratives existantes qui ne passent pas à l'échelle comparées aux méthodes spécialisées, en raison de l'encodage booléen des motifs séquentiels qui utilise des contraintes réifiées. Cette réflexion nous a conduit à explorer une autre voie consistant à introduire les techniques utilisées en fouille de données pour améliorer les performances de la PPC pour la fouille de séquences. Le premier résultat dans cette direction est la définition d'une contrainte globale pour l'extraction de motifs séquentiels qui exploite le principe des bases projetées. Les résultats obtenus sont importants car ils montrent, pour la première fois, qu'une approche PPC peut concurrencer (et même surpasser) les algorithmes spécialisés de l'état de l'art.

Travaux en collaboration avec le LITIO. J'ai été moteur dans la collaboration, développée depuis 3 ans, entre l'équipe CoDaG et le LITIO de l'université d'Oran 1. Cette collaboration s'est entre autres concrétisée par la thèse en cotutelle de Abdelkader Ouali que je codirige. Le premier volet de cette thèse a porté sur la conception et la mise en œuvre de méthodes hybrides parallèles exploitant la structure d'un problème en vue d'accélérer le processus de résolution. Le second volet consiste à s'attaquer au problème de passage à l'échelle des approches déclaratives existantes pour le clustering conceptuel sous contraintes afin de résoudre des problèmes de clustering de (très) grande taille. Dans ce cadre, Abdelkader Ouali a proposé une approche originale qui exploite l'efficacité des extracteurs de motifs utilisés par la communauté fouille de données et le cadre déclaratif de la programmation linéaire en 0/1. Il a également montré l'intérêt applicatif de son approche sur une grande variété de problèmes de clustering de motifs sous contraintes, non seulement en termes de qualité des motifs extraits, mais aussi en termes de temps d'exécution.

Je participe également à l'encadrement de la thèse de Mehdi Maamar en collaboration avec le LIRMM. L'objectif de cette thèse est d'étudier les apports des méthodes de fouille pour l'aide

6. `https://mulcyber.toulouse.inra.fr/projects/toulbar2/`
7. `http://hybride.loria.fr/`
8. Un wildcard est un symbole spécial qui peut remplacer n'importe quel item (attribut) dans une séquence.

à localisation de fautes dans les programmes à partir de traces d'exécution. En effet, la trace d'exécution fournie par un model-checker est souvent longue et difficile à comprendre, et de ce fait d'un intérêt très limité pour le programmeur qui doit débugger son programme. Ainsi, Mehdi Maamar a proposé une première formalisation du problème d'aide à la localisation de fautes dans un programme comme un problème de classification supervisée (exécutions correctes versus exécutions erronées) en fouille de données et a utilisé nos récents travaux sur l'extraction d'ensembles de motifs pour identifier les top-k suites d'instructions les plus suspectes. Une étape de classement des instructions issues des top-k motifs permet de localiser la faute.

1.3 Plan du mémoire

Ce mémoire est organisé en deux grandes parties. La première, présente mes travaux menés autour de la résolution des réseaux de fonctions de coût. La seconde, décrit plus spécifiquement mes travaux récents sur les apports de la programmation par contraintes pour la fouille de données. Ce mémoire utilise le matériel des différentes thèses que j'ai co-encadrées.

Le chapitre 2 regroupe les notions de PPC qui seront utilisées tout au long de ce mémoire. Il présente les notions de base relatives aux CSP et aux WCSP, et les principales méthodes de résolution associées.

Le chapitre 3 met l'accent sur nos travaux concernant les approches hybrides pour la résolution des WCSP. Tout d'abord, nous commençons par nos travaux sur les heuristiques génériques. Ensuite, nous présentons nos contributions concernant l'exploitation de la décomposition arborescente au sein des méthodes de recherche locale à voisinages variables (`VNS`), suivi par nos travaux portant sur la relaxation de contraintes globales. Nous montrons l'apport des contraintes globales relaxées pour la modélisation et la résolution des problèmes d'allocation d'emplois du temps des infirmières dans les hôpitaux (NRPs). Nous terminons ce chapitre par nos travaux sur les fonctions de coût décomposables et montrons comment les contraintes globales décomposables peuvent être relâchées sous la forme de fonctions de coût globales décomposables.

Le chapitre 4 introduit les principales notions utilisées en extraction de motifs ensemblistes, la modélisation, sous forme de CSP, des problèmes d'extraction de motifs sous contraintes, l'encodage booléen du ou des motif(s) recherché(s) et la problématique de l'extraction d'ensembles de motifs fondée sur des contraintes. Enfin, nous terminons par un rappel des principales notions liées à l'extraction de motifs séquentiels.

Le chapitre 5 présente nos activités plus récentes concernant l'utilisation de programmation par contraintes pour la fouille de données. Premièrement, nous présentons la notion de contrainte souple de seuil et montrons comment de telles contraintes peuvent êtres mises en œuvre pour l'extraction des top-k motifs souples. Deuxièmement, nous introduisons la notion de souplesse dans la relation de dominance et montrons l'apport de la PPC pour offrir un cadre unifié de manipulation de la souplesse dans le problème d'extraction des skypatterns. Troisièmement, nous introduisons une nouvelle structure originale appelée cube de skypatterns et présentons deux approches efficaces pour construire un tel cube. Quatrièmement, nous présentons une synthèse de nos contributions pour l'extraction de motifs séquentiels sous contraintes – à partir d'une base de séquences – en utilisant une approche PPC. Nous terminons ce chapitre par nos contributions dans les autres axes de recherche auxquels je me suis intéressé : aide à la localisation de fautes dans

un programme et le clustering conceptuel sous contraintes en PL-0/1. Une conclusion présente les voies de recherche que je souhaite poursuivre ces prochaines années.

1.4 Travaux reliés aux thèses et post-docs co-encadrés

La plupart des travaux présentés dans ce mémoire ont été effectués en collaboration. Plusieurs résultats sont reliés à des thèses ou travaux de post-doctorats que j'ai co-encadrés. Je décris brièvement ci-dessous les thèses soutenues, celles en cours et les encadrements post-doctoral, ainsi que l'endroit où les travaux associés sont décrits dans ce mémoire.

Thèses soutenues :

— Thèse Nicolas Levasseur (Oct. 2004 – Dec. 2008), *Heuristiques de recherche pour la résolution des WCSP* [Levasseur, 2008], thèse de l'UCBN. Co-encadrement avec Patrice Boizumault. Les résultats marquants de cette thèse sont présentés au chapitre 3, section 3.1.

— Thèse Jean-Philippe Métivier (Oct. 2006 – Avril 2010), *Relaxation de contraintes globales : Mise en œuvre et Application* [Métivier, 2010], thèse de l'UCBN. Co-encadrement avec Patrice Boizumault. Les travaux réalisés dans cette thèse sont présentés au chapitre 3, section 3.3.

— Thèse Mathieu Fontaine (Oct. 2009 – Juil. 2013), *Apport des méthodes de décomposition arborescente dans les méthodes de recherche locale à voisinages étendus* [Fontaine, 2013], thèse de l'UCBN. Co-encadrement avec Patrice Boizumault. Les contributions marquantes de cette thèse sont détaillées au chapitre 3, section 3.2.

— Thèse Willy Ugarte (Oct. 2011 – Nov. 2014), *Extraction de motifs sous contraintes souples* [Ugarte, 2014], thèse de l'UCBN. Co-encadrement avec Patrice Boizumault et Bruno Crémilleux. Les contributions marquantes de cette thèse sont décrites au chapitre 5, sections 5.1, 5.2 et 5.3.

Thèses en cours :

— Thèse Abdelkader Ouali (Dec. 2014 – Nov. 2017), *Méthodes hybrides parallèles pour la résolution de problèmes d'optimisation combinatoire : application au clustering sous contraintes*, thèse en cotutelle entre l'UCBN et l'université d'Oran 1. Co-encadrement avec Patrice Boizumault, Yahia Lebbah et Lakhdar Loukil. Les résultats importants de cette thèse sont présentés aux chapitres 3 et 5.

— Thèse Mehdi Maamar (Janv. 2014 – Dec. 2016), *Localisation d'erreurs dans les programmes en utilisant la fouille de données et la programmation par contraintes*, thèse de l'université d'Oran 1. Co-encadrement avec Yahia Lebbah (LITIO) et Nadjib Lazaar (LIRMM). Les premiers résultats de cette thèse sont décrits de manière synthétique au chapitre 5, section 5.5.

— Thèse Valentin Lemiere (Oct. 2015 – Nov. 2018), *Représentations PPC pour la fouille de grands ensembles de données*, thèse de l'UCBN. Co-encadrement avec Patrice Boizumault et Arnaud Lallouet.

Encadrement post-doctoral :

— Jean-Philippe Métivier (Oct. 2011 – Mars 2013), *Fonctions de coûts globales, décomposables*, Post-doc financé par le projet ANR FICOLOFO. Co-encadrement avec Patrice Boizumault. Les résultats de ce post-doc sont décrits au chapitre 3, section 3.4.

— Jean-Philippe Métivier (Avril 2013 – Juillet 2013), *Fouille de motifs séquentiels et PPC*, Post-doc financé par le projet ANR Hybride. Co-encadrement avec Thierry Charnois. Les résultats de ce post-doc sont décrits au chapitre 5, section 5.4.1.

— Amina Kemmar (Mars 2014 – Juin 2014), *Fouille de motifs séquentiels et PPC*, Post-doc financé par le projet ANR FICOLOFO. Co-encadrement avec Yahia Lebbah. Les résultats de ce post-doc sont décrits au chapitre 5, sections 5.4.2 et 5.4.3.

Première partie

Approches hybrides pour la résolution des WCSP

Chapitre 2

Contexte : Réseaux de fonctions de coût

Le formalisme des CSP (*Constraint Satisfaction Problem*) offre un cadre unificateur pour modéliser, étudier et résoudre de nombreux problèmes combinatoires. Il a l'avantage de pouvoir représenter les propriétés que doit satisfaire une solution, sous la forme de **variables** et de **contraintes**. L'intérêt d'une telle représentation, est de pouvoir développer des méthodes génériques de résolution indépendamment du problème traité. Toutefois, malgré sa généricité, le formalisme des CSP ne permet pas l'expression de préférences entre solutions, ni le calcul d'une solution dans le cas d'un problème sur-contraint. En effet, dans de nombreuses situations, une solution est requise même si pour cela, certaines contraintes (qui sont alors considérées comme peu importantes) ne sont pas satisfaites. Ainsi, plusieurs extensions ont été proposées pour répondre à cette limitation, parmi lesquelles nous pouvons citer [Fargier et Lang, 1993] permettant de modéliser les problèmes ayant des données incomplètes, [Rosenfeld *et al.*, 1976, Schiex, 1992, Ruttkay, 1994] permettant de définir des niveaux de préférences sur les tuples des contraintes, ou encore [Shapiro et Haralick, 1981, Freuder et Wallace, 1992] permettant de traiter les problèmes sur-contraints.

Un cadre unificateur appelé *Valued Constraint Satisfaction Problem* (VCSP) réunissant les différentes extensions citées précédemment a été proposé dans [Schiex *et al.*, 1995]. Dans ce document, nous nous intéressons à une instance de ce modèle, les réseaux de fonctions de coût (*Cost Function Network* ou CFN) ou Weighted CSP (WCSP) [Larrosa, 2002].

Plan du chapitre. Je commence par introduire les notions de CSP, de cohérence et les mécanismes de filtrage associés, ainsi que le schéma général des méthodes de recherche arborescentes. Je présente ensuite le formalisme des WCSP, les méthodes de recherche arborescente pour la résolution des WCSP et les techniques de filtrage associées. Enfin, je termine par les méthodes de recherche locale.

2.1 Le formalisme CSP

Un CSP est de défini formellement de la façon suivante :

Définition 2.1 (CSP)
Un réseau de contraintes est un triplet (X, D, C), avec :
— X, un ensemble fini de n variables ;

— D, un ensemble de domaines finis de valeurs, chaque variable $x_i \in X$ pouvant prendre une valeur de son domaine $D_i \in D$;
— C, un ensemble de contraintes, chacune $c_S \in C$ portant sur un sous-ensemble des variables $S \subseteq X$, appelé portée de la contrainte, et autorisant seulement une partie des combinaisons de valeurs possibles pour les variables de S.

Chaque réseau de contraintes est caractérisé par le triplet (n, d, e) où
— $n = |X|$ est le nombre de variables ;
— $d = \texttt{Max}_{i \in 1..n}(|D_i|)$ est la taille du plus grand domaine ;
— $e = |C|$ est le nombre de contraintes.

Définition 2.2 (Affectation)

— On appelle affectation d'une variable x_i, le fait d'attribuer à x_i une valeur de son domaine. L'affectation de la variable x_i à la valeur a_j est notée (x_i, a_j).
— On appelle affectation complète $\mathcal{A}$=$\{(x_1, a_1),\ldots,(x_n, a_n)\}$, l'affectation de toutes les variables de X. Si au moins une variable n'est pas affectée, on parlera d'affectation partielle.

Définition 2.3 (Contrainte satisfaite)
Une contrainte c_S est dite *satisfaite ssi* les variables $x_i \in S$ sont complètement instanciées et forment un tuple vérifiant c_S.

Un tuple autorisé est une affectation de toutes les variables de la portée d'une contrainte c_S satisfaisant celle-ci. Par opposition, un tuple interdit pour une contrainte c_S correspond à une affectation des variables de S ne satisfaisant pas c_S.

Définition 2.4 (Dureté d'une contrainte)
La *dureté* d'une contrainte est le rapport entre le nombre de ses tuples interdits et la taille du produit cartésien des domaines de ses variables.

L'arité d'une contrainte c_S est égale à la taille de S. Lorsque toutes les arités sont inférieures ou égales à 2, alors on dit que le réseau de contraintes est *binaire*.

Le problème de satisfaction de contraintes (Constraint Satisfaction Problem : CSP) consiste à donner une valeur à chaque variable de telle sorte que toutes les contraintes soient satisfaites.

2.1.1 Méthodes de recherche arborescente

Dans le cas général, l'existence d'une solution d'un CSP est un problème NP-complet. Cependant, de nombreuses méthodes ont été mises au point ces 20 dernières années pour essayer de résoudre efficacement des problèmes de taille industrielle. Parmi ces méthodes, on peut citer les méthodes complètes qui effectuent un parcours systématique de l'espace de recherche. Ces méthodes organisent l'espace de recherche sous forme d'un arbre : à chaque *nœud* de l'arbre correspond une variable (ou point de choix) et à chaque *branche* une des différentes affectations possibles des variables. Le long d'une branche, les domaines des variables se réduisent jusqu'à ce que toutes les variables soient affectées ou que l'on ait détecté un échec. Chaque feuille représente une affectation complète résultant des affectations faites le long de la branche menant à cette feuille. Si cette affectation complète satisfait toutes les contraintes du problème alors il s'agit d'une solution.

Une des principales caractéristiques des méthodes de recherche arborescente pour la résolution des CSP est leur complétude : chaque point de choix est tel qu'il réalise une partition de l'espace de recherche correspondant au nœud courant. De part leur parcours systématique, elles assurent un parcours complet de l'espace des solutions. Ces méthodes sont au cœur des systèmes de programmation par contraintes.

2.1.1a Algorithme BackTrack (BT)

L'algorithme de base pour résoudre les CSP est l'algorithme du BackTrack chronologique (BT) [Golomb et Baumert, 1965] : c'est une recherche en profondeur d'abord qui consiste à énumérer les valeurs possibles des variables récursivement, en remontant à la variable précédemment affectée dès qu'une contrainte est violée par l'affectation courante. La complexité temporelle dans le pire cas de cet algorithme est en $O(d^n)$. En effet, dans le pire des cas l'arbre de recherche est exploré en entier. Cet algorithme peut être amélioré en pratique par l'ajout d'un filtrage des domaines (en prétraitement ou au cours de la recherche après chaque affectation de variable) [Sabin et Freuder, 1994, Bessière et Régin, 1996, Debruyne et Bessière, 2001]. Ce filtrage consiste à supprimer les valeurs qui ne peuvent pas participer à une solution du CSP (cf. section 2.1.2).

2.1.1b Heuristiques sur l'ordre des variables et des valeurs

Les heuristiques sur l'ordre des variables et des valeurs sont une autre voie importante d'amélioration pour l'algorithme de BT. On distingue généralement deux types d'ordre : *statique*, i.e. les choix sont effectués avant la recherche et sont figés ; *dynamique*, i.e. les choix dépendent du sous-arbre à explorer (par exemple, nombre de variables affectées, taille des domaines, ...) et sont effectués à chaque nœud au cours de la recherche.

Le but d'une heuristique de choix de variable est de réduire la taille de l'espace de recherche, généralement en provoquant le plus rapidement possible un échec. De nombreuses heuristiques de choix de variable ont été proposées. Une description détaillée de ces heuristiques peut être trouvée dans la thèse de Nicolas Levasseur [Levasseur, 2008].

Les heuristiques de choix de valeur cherchent à aiguiller la recherche vers les branches les plus prometteuses de l'arbre de recherche. Parmi ces heuristiques, on retrouve :

- *lexicographique* (lex) : les valeurs sont simplement ordonnées selon l'ordre de leur domaine.
- *basée « look-ahead »* [Frost et Dechter, 1995] : cette heuristique classe les valeurs selon l'ordre croissant du nombre de retraits provoqués par l'affectation de celle-ci.
- *basée sur les explications* [Cambazard et Jussien, 2006] : cette heuristique utilise les informations liées au retrait d'une valeur (c'est-à-dire aux affectations justifiant ce retrait).
- *basée sur les impacts* [Refalo, 2004] : cette heuristique ordonne les valeurs par ordre croissant d'impact. L'impact associé à une valeur mesure l'influence de son affectation sur la taille de l'espace de recherche.

2.1.1c Recherche arborescente rendue incomplète

L'idée des affaiblissements d'algorithmes complets consiste à diriger la recherche dans des espaces de l'arbre jugés prometteurs, afin de trouver une solution très rapidement. C'est le principe des recherches guidées par une heuristique de choix de valeur qui produisent un parcours

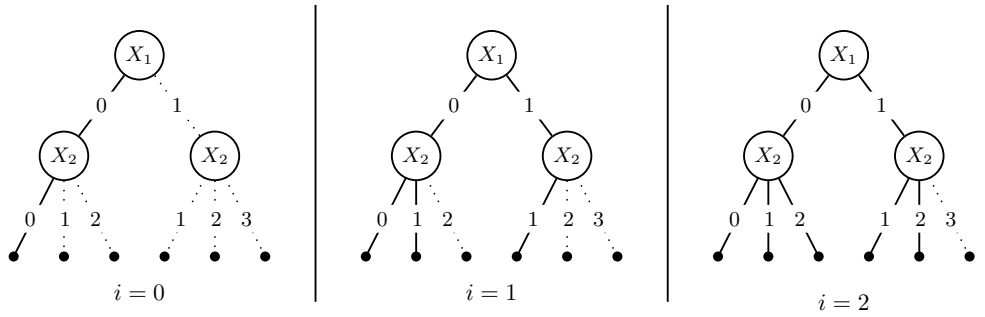


FIGURE 2.1 – Arbres de recherche explorés par LDS avec $\delta_{max} = 2$.

limitant les divergences par rapport aux choix proposés par l'heuristique. Citons la recherche LDS [Harvey et Ginsberg, 1995] qui limite le nombre de divergences. C'est une recherche arborescente dans laquelle l'ordre de visite des nœuds est modifié.

Soit h une heuristique de choix de valeur dans laquelle on a une grande confiance. Le principe de LDS est de suivre l'heuristique h lors du parcours de l'arbre de recherche, mais en permettant seulement un petit écart (δ) par rapport aux choix de h. On s'autorise donc δ écarts (ou *discrepancies*) à l'heuristique h. Pour un nombre maximal δ_{max} d'écarts autorisés, LDS explore l'arbre de recherche de manière itérative selon un nombre croissant d'écarts allant de i=0 à i=δ_{max}. À chaque itération, i est incrémenté de 1.

Dans la suite, nous ne considérons que des arbres d-aires. Dans un tel cas, l'heuristique h va ordonner toutes les valeurs du domaine d'une variable. La première valeur, selon l'ordre donné par h, n'engendre aucune augmentation de δ. La deuxième valeur augmente la valeur de δ de 1 et la troisième de 2, etc.

L'intérêt de LDS est d'explorer seulement les parties les plus intéressantes de l'arbre de recherche (selon les heuristiques) et non son intégralité comme le fait l'algorithme de BT. La figure 2.1 représente les branches de l'arbre de recherche exploré (en trait plein) lors des différentes itérations de LDS (avec δ_{max}=2). Nous verrons dans la section 2.3 comment exploiter ce type de méthodes dans une recherche locale.

2.1.2 Méthodes de filtrage

Le principal défi des méthodes arborescentes est de réduire la taille de l'arbre de recherche exploré en filtrant les domaines des variables au fur et à mesure, c-à-d en retirant les valeurs (ou combinaisons de valeurs) qui ne peuvent mener à aucune solution (du réseau de contraintes) et en propageant les conséquences de ces retraits. Plusieurs niveaux de cohérence et mécanismes de filtrage ont été proposés permettant de maintenir un certain niveau de cohérence souhaité à chaque nœud de l'arbre de recherche.

Pour un réseau de contraintes binaires, le filtrage le plus souvent utilisé est la *cohérence d'arc*. Il consiste à considérer chaque contrainte isolément et à supprimer les valeurs qui ne peuvent pas satisfaire la contrainte (i.e. qui n'appartiennent à aucune combinaison autorisée par la contrainte).

Définition 2.5 (COHÉRENCE D'ARC D'UN RÉSEAU DE CONTRAINTES BINAIRES)
Une contrainte binaire est arc-cohérente s'il existe pour chaque affectation d'une de ses variables

une affectation de l'autre variable (appelé *support*) qui satisfait la contrainte. Un CSP binaire est arc-cohérent (Arc Consistency (AC)) si toutes ses contraintes sont arc-cohérentes.

De nombreux algorithmes de renforcement d'arc-cohérence ont été proposés. AC-3 fut une des premières méthodes proposées [Mackworth, 1977]. Sa complexité en temps est en $O(e \cdot d^3)$. Plusieurs améliorations de cet algorithme ont été proposées.

On peut trouver une description détaillée de ces algorithmes dans le chapitre 3 du livre "Handbook of Constraint Programming" [Bessière, 2006]. Une extension de l'arc-cohérence pour les réseaux de contraintes n-aires, appelée *arc-cohérence généralisée* (GAC) ou encore *cohérence globale*, est également proposée dans [Bessière et Régin, 1997].

2.2 Le formalisme WCSP

Le formalisme des VCSP [Schiex *et al.*, 1995] étend le formalisme des CSP en associant une valuation à chaque tuple de chaque contrainte. Pour un tuple, sa valuation exprime le degré d'insatisfaction engendré lorsque les variables de la contrainte sont affectées aux valeurs du tuple. De telles valuations seront prises dans un ensemble E munie d'une *structure de valuations*.

Une structure de valuations permet de représenter les coûts (ou valuations) des tuples des contraintes ainsi que la manière de les agréger. Le formalisme WCSP est une variante des VCSP avec une structure de valuation spécifique.

Définition 2.6 (Structure de valuations d'un WCSP)
La structure de valuations associée à un WCSP est un quintuplet $\mathcal{S} = (E^+, \oplus, \leq, 0, \top)$ où
- $E^+ = [0 \dots \top]$ est un intervalle d'entiers, avec $\top$ un entier positif quelconque ou $+\infty$,
- $\leq$ est l'opérateur d'ordre standard sur E^+,
- $\oplus$ est l'addition bornée dans $\mathbb{N}$ définie par : $\forall a, b \in E^+, a \oplus b = min(\top, a + b)$. Un opérateur de soustraction de coût, noté $\ominus$, est également défini pour « retirer » un coût d'une fonction comme suit :

$$\forall a \geq b \in \mathcal{E}, a \ominus b = \begin{cases} \top \; si \; a = \top \\ a - b \; sinon \end{cases}$$

Définition 2.7 (WCSP)
Un réseau de fonctions de coût (CNF) est défini par un quadruplet $(X, D, W, \mathcal{S})$ où :
- $X = \{x_1, \dots, x_n\}$ est l'ensemble des variables du problème,
- $D = \{D_1, \dots, D_n\}$ est l'ensemble des domaines finis associés aux variables,
- $\mathcal{S} = (E^+, \oplus, \leq, 0, \top)$ est une structure de valuations,
- $W = \{w_{S_1}, \dots, w_{S_e}\}$ est l'ensemble des fonctions de coût, chacune $w_{S_i} \in W$ portant sur un ensemble des variables $S_i \subset X$ et associe un coût entier dans E^+ pour chaque combinaison de valeurs possibles des variables de S_i (noté D^{S_i}) : $w_{S_i} : D^{S_i} \to E^+$

$\top$ est un coût entier maximum utilisé pour représenter les affectations interdites. Une contrainte **dure** est donc représentée par une fonction de coût prenant des valeurs dans l'ensemble $\{0, \top\}$ seulement ; dans le cas contraire la contrainte est dite **molle**.

Définition 2.8 (L'hypergraphe d'un WCSP)
Soit $\mathcal{P} = (X, D, W, \mathcal{S})$ un réseau de fonctions de coût, l'hypergraphe (X, A) associé à $\mathcal{P}$ est défini par :

— un sommet u_i par variable $x_i \in X$;
— une hyper-arête par portée S telle que $\exists\ w_S \in W$.

Définition 2.9 (Voisinage d'une variable)
Le voisinage $Vois(x_i)$ d'une variable x_i est l'ensemble des variables comprises dans la portée d'une fonction de coût contenant x.

$$Vois(x_i) = \{x_j \in X \mid \exists\ w_S \in W, (x_i\ \in S) \wedge (x_j \in S)\}$$

On associe au WCSP une fonction $w_\emptyset$ d'arité nulle. C'est une constante ajoutée à toute affectation et sert de minorant utilisé par DFBB (voir section 2.2.1). Dans ce qui suit, on notera par $\mathcal{A}[S]$ la projection des valeurs de l'affectation complète $\mathcal{A}$ sur les variables de S.

Définition 2.10 (Coût d'une affectation dans un WCSP)
Le coût d'une affectation complète $\mathcal{A} = \{(x_1, a_1), \ldots, (x_n, a_n)\}$ est donnée par :

$$f(\mathcal{A}) = w_\emptyset \bigoplus_{i=1}^{e} w_{S_i}(\mathcal{A}[S_i])$$

Le problème de satisfaction de contraintes pondérées (Weighted Constraint Satisfaction Problem (WCSP)) consiste à trouver une affectation complète $\mathcal{A}$ de l'ensemble des variables de coût $f(\mathcal{A})$ minimal. Il s'agit d'un problème NP-difficile.

2.2.1 Depth First Branch and Bound (DFBB)

Le schéma classique de résolution d'un WCSP repose sur une recherche arborescente en profondeur d'abord basée sur le calcul d'un minorant, DFBB (Depth First Branch and Bound). À chaque nœud de l'arbre de recherche, un minorant ($\mathcal{LB}$) du coût de toutes les extensions (affectations complètes) issues de l'affectation partielle courante est calculé. Ce minorant est ensuite comparé au majorant du coût d'une solution optimale du problème. Celui ci correspond au coût de la meilleure solution trouvée à ce point de la recherche. Ainsi, si le minorant est supérieur au majorant, aucune extension de la solution partielle courante ne pourra être de meilleure qualité, et le sous-arbre peut être élagué.

La complexité de DFBB est la même que BT. L'efficacité de DFBB dépend de la qualité du calcul du minorant $\mathcal{LB}$. Un minorant trivial consiste à sommer le coût des fonctions dont les variables sont affectées. Nous allons voir dans la section 2.2.2 comment calculer de bien meilleurs minorants.

2.2.2 Filtrage par cohérences locales souples

Plusieurs travaux ont été menés pour adapter au cadre WCSP des propriétés (et algorithmes efficaces) définies pour le cadre CSP, dans le but de permettre la déduction de valeurs impossibles et de fournir à la recherche un minorant le plus précis possible. On peut citer par exemple la cohérence de nœud (NC) [Larrosa, 2002], la cohérence d'arc souple (AC) [Schiex, 2000]), la cohérence d'arc directionnelle (DAC) [Cooper, 2003, Larrosa et Schiex, 2003], la cohérence

Algorithme 1 : Algorithme de projection.

```
1 procédure Project (w_S, x_i, a_j, α) ;
2 début
4 |   pour tout A ∈ D^S tel que A[x_i] = a_j faire
5 |     w_s(A) ← w_s(A) ⊖ α;
7 |   w_i(a_j) ← w_i(a_j) ⊕ α ;
8 fin
```

Algorithme 2 : Algorithme d'extension.

```
1 procédure Extend (w_S, x_i, a_j, α) ;
2 début
4 |   pour tout A ∈ D^S tel que A[x_i] = a_j faire
5 |     w_s(A) ← w_s(A) ⊕ α;
7 |   w_i(a_j) ← w_i(a_j) ⊖ α ;
8 fin
```

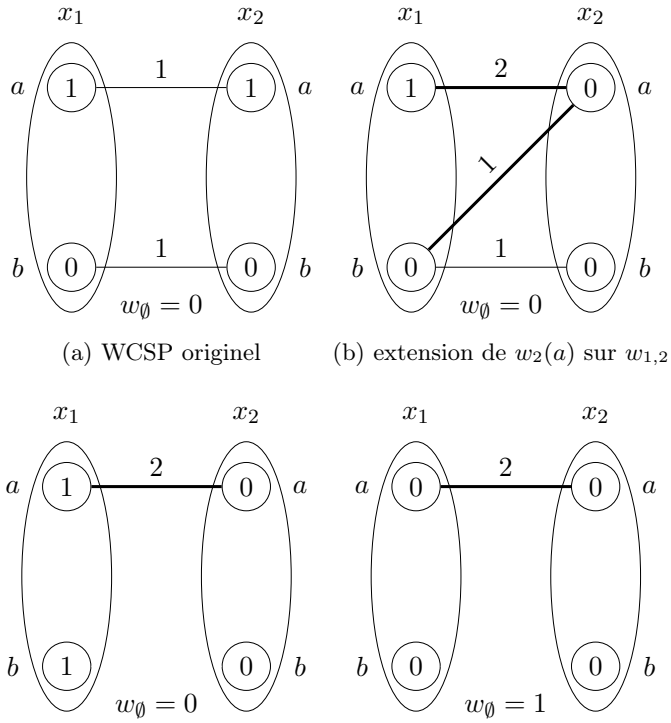
d'arc directionnelle complète (FDAC) [Cooper, 2003, Larrosa et Schiex, 2003], la cohérence d'arc existentielle et directionnelle (EDAC) [de Givry *et al.*, 2005] et l'arc cohérence virtuelle (VAC) [Cooper *et al.*, 2008].

Les algorithmes évoqués ci-dessus utilisent des opérations de transfert de coûts, appelées transformations préservant l'équivalence (EPTs). L'idée principale est de rechercher des *reformulations* en temps polynomial d'un WCSP de manière à faire apparaître explicitement un bon minorant de l'optimum du problème traité. Ces reformulations transforment le réseau en un réseau équivalent, c'est à dire qu'elles ne changent pas le coût des affectations complètes. Elles agissent en déplaçant des coûts entre les fonctions, avec pour objectif de transférer les coûts vers la fonction d'arité nulle $w_\emptyset$ [de Givry, 2011].

Nous supposons par la suite l'existence d'une fonction de coût unaire, noté w_i, portant sur chaque variable x_i du problème. Les EPTs classiques utilisées dans les algorithmes de filtrage par cohérences locales souples sont les suivantes :

- opération de *projection* (cf. Algorithme 1) qui déplace un coût α d'une fonction w_S vers une fonction unaire $w_i(a_j)$, avec $x_i \in S$ et $a_j \in D_i$.
- opération *d'extension* (cf. Algorithme 2), qui est l'opération inverse de la projection, vise à déplacer un coût d'une fonction unaire vers une autre fonction de coût.
- opération de *projection unaire* qui déplace un coût α d'une fonction unaire w_i vers $w_\emptyset$.

Exemple 2.1. Soit le réseau à deux variables x_1 et x_2 à domaine $\{a, b\}$ de la figure 2.2a ayant 3 fonctions de coût : deux fonctions de coût unaires w_1 et w_2 portant respectivement sur x_1 et x_2 et une fonction de coût binaire $w_{1,2}$. Les coûts unaires non nuls sont représentés dans des cercles et les coûts binaires non-nuls sur des arêtes. L'extension de la fonction de coût unaire de x_2 vers $w_{1,2}$ permet de déplacer le coût lié à (x_2, a) sur $w_{1,2}$ (figure 2.2b). Puis, la projection sur $w_1(b)$ de $w_{1,2}$ sur la figure 2.2c, permet d'incrémenter $w_1(b)$ de 1. Enfin, le minorant peut être augmenté de 1 en déplaçant le coût minimal de w_1 vers $w_\emptyset$ (figure 2.2d).



(a) WCSP originel (b) extension de $w_2(a)$ sur $w_{1,2}$

(c) projection de $w_{1,2}$ sur $w_1(b)$ (d) projection de w_1 sur $w_\emptyset$

FIGURE 2.2 – Opérations d'extension et de projection.

2.2.2a Cohérences pour les WCSP

À partir de la notion d'équivalence, et des opérations de projection et d'extension, plusieurs notions de cohérence ont été proposées.

Définition 2.11 (NŒUD COHÉRENCE [LARROSA, 2002])
Une variable x_i est NC si chaque valeur $a \in D_i$ vérifie $w_i(a) + w_\emptyset < \top$ et $\exists\ a \in D_i$ telle que $w_i(a) = 0$. Un $WCSP$ est NC si, et seulement si, toutes ses variables le sont.

Tout WCSP $\mathcal{P}$ peut être rendu nœud cohérent en projetant itérativement les valuations minimales des fonctions de coût unaires sur $w_\emptyset$. Puis, toute valeur qui n'est pas nœud cohérente est filtrée. Enfin, si aucun domaine n'est vide, $\mathcal{P}$ est alors NC. L'établissement de la nœud cohérence a une complexité temporelle de $O(nd)$.

L'arc cohérence. Elle permet de filtrer les valeurs des domaines en tenant compte des fonctions de coût binaires.

Définition 2.12 (ARC COHÉRENCE [SCHIEX, 2000])
Une fonction de coût binaire w_{ij} est arc-cohérente s'il existe pour chaque affectation d'une de ses variables une affectation de l'autre variable telle que le coût soit nul. Un WCSP binaire est arc-cohérent (AC*) si toutes ses fonctions de coût binaires sont arc-cohérentes et si toutes ses variables sont NC.

Cohérence d'arc directionnelle. Une amélioration de AC* consiste à combiner les coûts binaires et les coûts unaires dans le calcul du minimum à projeter. DAC a été originellement

introduit sur les fonctions de coût binaires en s'appuyant sur la notion de support complet et en limitant la recherche du support complet dans une seule direction pour chaque fonction de coût selon un ordre défini sur les variables [Larrosa et Schiex, 2003].

Définition 2.13 (Cohérence d'arc directionnelle complète)
Soit un CFN $\mathcal{P} = (X, D, W, \mathcal{S})$ constitué de fonctions de coût $w_S \in W$ avec $|S| = 2$ (notées w_{ij}) et un ordre total $\prec$ sur les variables :

- pour toute fonction de coût $w_S \in W$, un tuple $\mathcal{A} \in D^S$ est un support complet pour une valeur $a \in D_i$ de $x_i \in S$ *ssi* $\mathcal{A}[x_i] = a$ et $w_S(\mathcal{A}) \bigoplus_{x_j \in S, j \neq i} w_j(\mathcal{A}[x_j]) = 0$.
- un CFN P est DAC par rapport à l'ordre $\prec$ *ssi* pour toute variable x_i, pour toute valeur $a \in D_i$ et pour toute fonction de coût $w_{ij} \in W$, telle que $i < j$, il existe un support complet pour (x_i,a) sur w_{ij}. P DAC* s'il est DAC et NC.

Terminal DAC. DAC a été initialement étendu aux fonctions de coût non binaires dans [Lee et Leung, 2009] et [Sanchez *et al.*, 2008]. Une autre extension, appelée T-DAC (pour Terminal DAC), a été également proposée :

Définition 2.14 (T-DAC [Allouche *et al.*, 2012a])
Étant donné un ordre total $\prec$ sur les variables, un CFN est dit T-DAC par rapport à l'ordre $\prec$ *ssi*, pour toute fonction de coût w_S, toute valeur (x_i, a) de la variable minimum $x_i \in S$ selon $\prec$ est DAC* pour w_S.

2.3 Recherches locales

Quand on a du mal à guider une recherche arborescente et que l'espace de recherche est trop grand pour être exploré entièrement, il devient intéressant d'utiliser des méthodes de recherche locale. Contrairement aux méthodes arborescentes qui construisent une solution, les méthodes de recherche locale effectuent des mouvements dans l'espace de recherche. Bien que ces méthodes ne soient pas complètes, et ne puissent par conséquent pas effectuer de preuve d'optimalité, elles produisent très souvent des solutions de bonne qualité dans des temps de calcul raisonnables.

2.3.1 Principe d'une recherche locale

Une méthode de recherche locale (RL) est fondée sur deux éléments essentiels : la définition d'un *voisinage* (permettant d'obtenir les solutions proches d'une solution donnée) et la donnée d'une *stratégie d'exploration* du voisinage.

Définition 2.15 (Voisinage d'une solution)
Soit $\mathcal{S}$ un ensemble de solutions. On appelle **voisinage** toute application $N : \mathcal{S} \to 2^{\mathcal{S}}$ associant à toute solution $S \in \mathcal{S}$ un ensemble de solutions noté $N(S)$ dites voisines.

Définition 2.16 (Mouvement)
On appelle **mouvement** l'opération qui consiste à passer d'une solution $S \in \mathcal{S}$ à une solution voisine $S' \in N(S)$.

Définition 2.17 (Minimum local pour un voisinage N)
Une solution S est dite minimum localement au voisinage N si, et seulement si, pour toute solution S' dans $N(S)$: $f(S) \leq f(S')$.

L'algorithme 3 décrit un schéma générique d'une recherche locale, avec $iter_{max}$ le nombre maximal de mouvements autorisés sans amélioration de la qualité de la solution. L'algorithme

Algorithme 3 : Shéma générique d'une recherche locale.

```
1  Fonction rechercheLocale(iter_max, N) ;
2  début
3      S ← genSolInit(P) ;
4      iter ← 1 ;
5      while (iter < iter_max) ∧ (not TimeOut) do
6          S' ← selectVoisin(S) ;
7          si (f(S') < f(S)) alors
8              S ← S' ;
9              iter ← 1 ;
10         sinon
11             iter ← iter + 1 ;
12     retourner S ;
13 fin
```

part d'une solution initiale S (ligne 3), puis tente de l'améliorer en enchaînant une succession de mouvements locaux. À chaque mouvement, elle sélectionne une solution S' dans son *voisinage* $N(S)$ si possible de meilleure qualité (ligne 6). Lorsque la solution courante n'a pas pu être améliorée pendant $iter_{max}$ mouvements, ou que le *TimeOut* est atteint (ligne 5), l'algorithme retourne la meilleure solution trouvée (ligne 12).

La stratégie d'exploration d'un voisinage est l'autre composante de base dans une RL. Elle permet de préciser comment passer d'une solution S à une autre solution voisine $S' \in N(S)$. Plusieurs stratégies peuvent être envisagées pour effectuer ce choix. Une première stratégie, dite **première amélioration**, consiste à énumérer les voisins de S jusqu'à en rencontrer un qui améliore la solution courante. Une autre stratégie, dite **meilleure amélioration**, consiste à explorer le voisinage $N(S)$ de manière complète afin de rechercher le meilleur voisin.

Aspects stochastiques. Pour éviter des biais dus à des choix déterministes qui feraient explorer toujours les mêmes parties de l'espace de recherche, une part d'aléatoire est introduite dans ces algorithmes les rendant stochastiques. Par exemple, le choix de la solution initiale peut être complètement aléatoire. Le choix d'un voisin fait souvent intervenir une part d'aléatoire.

Minima locaux. L'algorithme de base de la recherche locale est l'algorithme de *descente* (ou *Hill-Climbing*), qui n'accepte que des voisins améliorant la solution courante. Cet algorithme est souvent piégé dans des minimums locaux. De nombreux mécanismes ont alors été définis pour essayer de contrôler les méthodes de recherche locale et les guider afin d'éviter qu'elles restent bloquées dans un minimum local. On les appelle généralement des *méta-heuristiques*[9]. Les plus connues sont la *recherche taboue* [Glover, 1986] qui garde une mémoire à court terme pour éviter de cycler et le *recuit simulé* [Kirkpatrick *et al.*, 1983] qui autorise des mouvements détériorants avec une probabilité qui décroît au cours de la recherche. L'idée de base de ces méthodes est de *diversifier* l'exploration de l'espace de recherche.

Diversification. Le but de la diversification est de parcourir un grand nombre de régions différentes de l'espace de recherche, afin d'assurer que l'espace de solutions est correctement exploré et ainsi localiser la région contenant l'optimum global.

9. Le lecteur intéressé par une synthèse des méta-heuristiques pourra se référer à [Hao *et al.*, 1999], pour de plus amples détails.

Algorithme 4 : Changement de voisinage.

```
1 Procédure NeighbourhoodChange(S, S', k);
2 début
3     si (f(S') < f(S)) alors
4         S ← S' ; k ← 1 /*Make a move*/ ;
5     sinon  k ← k + 1 /*Next neighbourhood*/ ;
6 fin
```

Algorithme 5 : Pseudo-code de la recherche `VNS`.

```
1  Fonction VNS(S, k_max, t_max);
2 début
3     répéter
4         k ← 1;
5         répéter
6             S' ← Shake(S,N_k) /* Shaking */ ;
7             S'' ← selectPremierVoisin(S') /* Local search */ ;
8             NeighbourhoodChange(S, S'', k) /*Change neighbourhood*/ ;
9         jusqu'à k = k_max ;
10        t ← CpuTime();
11    jusqu'à t > t_max ;
12    retourner S
13 fin
```

2.3.2 Variable Neighbourhood Search

Contrairement aux méthodes de recherche locale présentées ci-dessus, la recherche à voisinage variable (`VNS`), [Mladenovic et Hansen, 1997], est une recherche à grand voisinage qui utilise méthodiquement plusieurs types de voisinages, dans le but de s'échapper des minima locaux.

Soit $L = \{N_1, \ldots, N_{k_{max}}\}$ une liste finie de *structures de voisinage*, ordonnée par ordre croissant de taille, où $N_k(S)$ ($k \in [1 \ldots k_{max}]$) désigne l'ensemble des solutions voisines de S dans N_k. Dans la plupart des méthodes de recherche locale, nous avons $k_{max} = 1$.

L'idée centrale de la recherche `VNS` se trouve dans le changement systématique de la structure de voisinage, selon la procédure `NeighborhoodChange`. L'algorithme 4 détaille son schéma général. Soit une solution courante S et S' la nouvelle solution obtenue dans N_k ; Si S' est de meilleure qualité que S (ligne 3), alors S' devient la nouvelle solution courante et on retourne à N_1 (ligne 4). Sinon, on passe à N_{k+1} (ligne 5). Le fait d'utiliser plusieurs structures de voisinage permet de *diversifier* l'exploration de l'espace de solutions afin d'accéder à un plus grand nombre de régions intéressantes.

L'algorithme 5 détaille le pseudo-code de `VNS`, avec N_k ($k = 1, \ldots, k_{max}$) la k-ième structure de voisinage. Soit t_{max} le temps maximal alloué à la recherche ($TimeOut$). L'algorithme part d'une solution initiale S souvent générée aléatoirement. À chaque itération de la boucle des lignes (5-8), une solution S' est sélectionnée aléatoirement (phase de *shaking*) dans la k-ième structure de voisinage de S ($S' \in N_k(S)$) (ligne 6). Une méthode de recherche locale est ensuite appliquée en utilisant S' comme solution initiale et une nouvelle solution S'' est retournée (ligne 7). Ensuite, la procédure de changement de voisinage est appelée avec pour argument S, S'' et k (ligne 8).

Algorithme 6 : Pseudo-code de l'algorithme VNS/LDS+CP.

```
1  8
2  Fonction VNS/LDS+CP (X, W, k_init, k_max, δ_max) ;
3  début
4      S ← genSolInit(X, W) ;
5      k ← k_init ;
6      tant que (k < k_max) ∧ (not TimeOut) faire
7          X_un ← Hneighborhood(N_k, X, W, S) ;
8          A ← S\{(x_i, a) | x_i ∈ X_un} ;
9          S' ← LDS + CP(A, X_un, δ_max, f(S), S) ;
10         NeighbourhoodChange(S, S'', k) ;
11     retourner S ;
12 fin
```

L'algorithme s'arrête dès que l'on a atteint le $TimeOut$ (ligne 3).

Les grands voisinages constituent donc une alternative très intéressante pour diversifier la recherche et assurer la convergence vers un optimum local de haute qualité. Toutefois, sur des problèmes de grande taille, l'exploration de ces voisinages peut s'avérer très coûteuse en temps et peut ralentir considérablement l'amélioration de la solution courante.

Afin de privilégier l'exploration de petits voisinages, Variable Neighborhood Decomposition Search (VNDS) [Hansen *et al.*, 2001], qui est une extension de VNS, vise à réduire l'espace de solutions parcouru par la recherche locale. Pour cela, à chaque itération, la recherche est effectuée uniquement sur un sous-problème déterminé heuristiquement. Le principe est de fixer une sous partie du problème et de rechercher de nouvelles solutions en ne modifiant que les parties « libres » du problème. Cette recherche est effectuée par la recherche VNS appliquée au sous-problème déterminé précédemment.

2.3.3 VNS/LDS+CP

Dans le cadre de ma thèse, j'ai proposé la méthode VNS/LDS+CP [Loudni et Boizumault, 2003, Loudni et Boizumault, 2008], qui tire parti des mécanismes de filtrage liés aux WCSP afin d'accélérer le parcours de l'espace de recherche. VNS/LDS+CP peut être vue comme une extension de VNDS où la phase de shaking est effectuée en désaffectant k variables du problème, tandis que la phase de reconstruction est réalisée grâce à une méthode arborescente partielle (LDS), combinée avec des mécanismes de propagation de contraintes sur les fonctions de coût (CP).

L'algorithme 6 décrit son pseudo-code, avec W l'ensemble des fonctions de coût, k_{init} (resp. k_{max}) le nombre minimal (resp. maximal) de variables à désaffecter et δ_{max} la valeur maximale de la discrepancy pour LDS. Il part d'une solution initiale S générée aléatoirement (ligne 4). Un sous-ensemble de k variables (avec k la dimension du voisinage) est sélectionné dans le voisinage N_k (i.e. l'ensemble des combinaisons de k variables parmi X) par l'heuristique de choix de voisinage Hneighborhood (ligne 7). Une affectation partielle $\mathcal{A}$ est alors générée à partir de la solution courante S, en désaffectant les k variables sélectionnées ; les autres variables (i.e. non sélectionnées) gardent leur affectation dans S (ligne 8). $\mathcal{A}$ est alors reconstruite en utilisant une recherche arborescente partielle de type LDS, aidée par une propagation de contraintes (CP) sur les fonctions de coût. Si LDS trouve une solution de meilleure qualité S' dans le voisinage de

S, S' devient la solution courante et k est réinitialisé à k_{init}. Sinon, on cherche de nouvelles améliorations dans N_{k+1} (structure de voisinage où $(k+1)$ variables de X seront désaffectées). Ce traitement est assuré par la procédure `NeighborhoodChange` (ligne 10). L'algorithme s'arrête dès que l'on a atteint la dimension maximale des voisinages à considérer k_{max} ou le $TimeOut$ (ligne 6).

Chapitre 3

Contributions à la résolution des WCSP

Ce chapitre introduit nos travaux portant sur la résolution des WCSP donnant lieu à quatre contributions :

- La section 3.1 présente les contributions de la thèse de Nicolas Levasseur sur différentes heuristiques génériques (i) pour le choix de variable et de valeur basées sur l'historique des solutions et (ii) pour le choix de voisinage « dense » pour la méthode `VNS`/`LDS`+`CP`.
- La section 3.2 introduit un cadre générique, proposé par Mathieu Fontaine dans sa thèse, pour l'exploitation de la décomposition en arbre d'un graphe, afin de guider efficacement le choix des voisinages dans les méthodes de recherche locale de type `VNS` (cf. section 3.2.2). Plusieurs propositions d'amélioration de ce schéma (noté `DGVNS`) sont également détaillées (cf. sections 3.2.3 et 3.2.4). Enfin, nous décrivons en section 3.2.5 différentes stratégies parallèles, proposées par Abdelkader Ouali dans sa thèse, permettant d'explorer les clusters de la décomposition arborescente en parallèle dans `DGVNS`.
- La section 3.3 introduit un cadre unifié de relaxations (orienté variables et orienté décomposition), proposé par Jean-Philippe Métivier dans sa thèse, pour différentes contraintes globales existantes, permettant de prendre en compte de manière fine des préférences où des coûts de violation des contraintes. Une jolie justification de ce cadre est ensuite donnée, montrant son apport sur un problème éminemment sur-contraint et complexe.
- La section 3.4 présente nos récents travaux, menés dans le cadre du projet ANR FICO-LOFO, sur les fonctions de coût globales décomposables.
- La section 3.5 conclut le chapitre par une discussion sur nos contributions les plus marquantes.

3.1 Heuristiques de recherche pour la résolution des WCSP

3.1.1 Heuristiques génériques pour les WCSP

(Travail en collaboration avec Nicolas Levasseur et Patrice Boizumault ; publications associées : ICTAI'07 [Levasseur *et al.*, 2007] et JFPC'08 [Levasseur *et al.*, 2008b]).

Contexte. Définissant l'ordre d'exploration de l'espace de recherche des méthodes arborescentes, les heuristiques de choix de valeur et de variable ont une influence importante sur l'efficacité de ces méthodes. Il est bien sûr toujours intéressant d'utiliser la connaissance liée au problème pour

faire en premier les choix les plus pertinents et se diriger vers les branches les plus prometteuses. En l'absence de connaissance spécifique, il est alors utile d'avoir des heuristiques génériques et efficaces. Bien que de nombreuses heuristiques génériques aient été proposées pour les méthodes de recherche arborescente dans le cadre des CSP, peu de travaux ont été effectués afin de définir des heuristiques adaptées au cadre des WCSP. Parmi ceux-ci nous pouvons citer les heuristiques de choix de valeur MinAC [Larrosa et Schiex, 2003] et de variable *2-sided Jeroslow-like* [de Givry *et al.*, 2003] basée sur les coûts des contraintes. MinAC exploite les coûts unaires associées aux valeurs des domaines des variables, obtenus par filtrage par les cohérences locales décrites en section 2.2.2. En sélectionnant la valeur ayant le coût unaire le plus petit, la recherche est guidée vers les sous-espaces les moins incohérents. Toutefois, cette heuristique présente deux inconvénients majeurs :

— elle ne permet pas de départager les nombreux candidats ex-æquo ;
— elle est dépourvue de mécanisme de mémorisation lui permettant d'adapter ses choix en fonction de ses erreurs passées.

Contributions. Afin de pallier les inconvénients évoqués précédemment, nous avons défini un critère global, H-Qualité (HQ), plus indépendant des mécanismes de filtrage, qui exploite l'historique des solutions trouvées au cours de la recherche. En se basant sur ce critère, nous avons proposé de nouvelles heuristiques de choix de valeur et de variable.

3.1.1a Heuristiques de choix de valeur basées sur la H-Qualité

La H-Qualité s'appuie sur une idée simple : associer à chaque valeur de chaque variable le coût d'une meilleure solution connue passant par cette valeur (solution déjà explorée ou obtenue par échantillonnage glouton).

Définition 3.1 (H-qualité d'une affectation [Levasseur *et al.*, 2008b])
Pour $x_i \in X, a \in D_i$ et un historique H.

$$HQ_{val}(x_i, a, H) = \min\{f(\mathcal{A}) \mid \mathcal{A} \in H \wedge \mathcal{A}[x_i] = a\}$$

$HQ_{val}(x_i, a, H)$ représente le coût de la meilleure solution de l'historique H contenant l'affectation $(x_i = a)$. Plus les solutions contenues dans H sont de bonne qualité, plus la notion de H-Qualité devient pertinente car approchant, pour chaque affectation $(x_i = a)$, le coût minimal d'une affectation complète contenant $(x_i = a)$.

En se basant sur ce critère, nous avons étudié deux variantes qui ordonnent les valeurs du domaine d'une variable selon l'ordre croissant [10] des valeurs d'H-Qualité :

— `HQOnce`, qui effectue un échantillonnage glouton à chaque nœud mais uniquement pour les valeurs n'apparaissant dans aucune solution connue,
— `HQAll` qui effectue un échantillonnage à chaque nœud sur toutes les valeurs. L'idée est d'augmenter la diversité et le nombre de solutions présentes dans l'historique.

Échantillonnage glouton. Certaines valeurs d'H-Qualité peuvent être inconnues : pour une valeur $a \in D_i$, H peut très bien ne contenir aucune solution contenant (x_i, a). Dans ce cas, afin de renseigner la H-Qualité de l'affectation (x_i, a), nous effectuons un échantillonnage glouton à

10. En effet, plus la valeur d'H-Qualité d'une affectation est petite, plus cette affectation a des chances de générer peu d'incohérences. A égalité, les valeurs ex-æquo sont départagées par MinAC.

partir de l'affectation partielle $\mathcal{A}_p \cup \{(x_i = a)\}$. Chaque échantillonnage glouton est guidé par une heuristique de choix de valeur basée sur MinAC. En cas d'égalité nous départageons en choisissant la valeur la moins souvent affectée depuis le début de la recherche [11]. Par ailleurs, afin de trouver rapidement une solution, la phase de filtrage est désactivée et les coûts unaires pour chacune des valeurs des variables non encore affectées sont mis à jour par *forward checking* : pour toute fonction de coût $w_S \in W$ contenant l'affectation (x_i, a) et ayant une seule variable x_j non affectée dans S, pour toute valeur $b \in D_j$, si la valeur (x_j, b) génère une nouvelle violation, celle-ci est projetée sur $w_j(b)$.

3.1.1b Heuristiques de choix de variable basées sur les H-Qualités

À partir des H-qualités des valeurs du domaine d'une variable, nous avons défini un critère de qualité au niveau variable HQ_{var}, en agrégeant les H-qualités des valeurs d'une variable par un opérateur de type *max* ou *moyenne*.

Combiner HQ_{var} avec d'autres critères. Comme pour les heuristiques de choix de variable pour les CSP, nous avons proposé de combiner la H-Qualité d'une variable avec la taille de son domaine et son degré comme suit :

1. L'heuristique dom/$HQ_{var}(op)$ sélectionne la variable ayant le plus petit ratio suivant $|D_i|/HQ_{var}(x_i, H, op)$, avec op désignant l'opérateur d'agrégation utilisé.
2. Initialement, les variables sont ordonnées selon trois heuristiques : taille de leur domaine (dom), leur degré (fdeg) et leur H-Qualité. Soit p_i un compteur représentant la somme des positions de la variable x_i dans ces trois ordres. L'heuristique `RANK`(dom,fdeg,$HQ_{var}(op)$). sélectionne la variable ayant le plus petit compteur p_i.

Initialisation des HQ_{var}. Au début de la recherche, aucune solution n'étant présente dans l'historique, les H-Qualités de toutes les variables sont inconnues. Dans le but de les calculer, les H-Qualités des valeurs doivent être initialisées auparavant (i.e. avant le début de la recherche).

Complexité. L'initialisation des H-Qualités de toutes les valeurs des variables nécessite dans le pire cas $n \times d$ échantillonnages, chacun étant effectué en $O(e \times d)$. La complexité temporelle de cette phase d'initialisation est $O(e \times n \times d^2)$.

Bilan des expérimentations. Nous avons réalisé plusieurs séries d'expérimentations sur des instances réelles du problème d'allocation de fréquences à des liens radio `RLFAP` mais aussi sur des instances binaires de WCSP générées aléatoirement à partir d'un programme fourni par Simon de Givry. Les résultats de nos expérimentations ont montré que les **heuristiques de choix de valeur** `HQOnce` et `HQAll` guidaient plus efficacement la recherche que MinAC. Par ailleurs, nous avons observé que l'écart de performances entre ces deux heuristiques augmentait avec l'augmentation de la dureté et/ou de la connectivité du graphe de contraintes. Nous avons également remarqué que l'heuristique `HQOnce` obtenait de meilleurs résultats lorsque la dureté des contraintes était faible. Cependant, lorsque celle-ci augmentait, les mises à jour régulières des H-Qualités effectuées par `HQAll`, permettaient de diversifier l'historique et ainsi de guider plus efficacement la recherche. Pour les **heuristiques de choix de variable**, les résultats étaient décevants car aucune tendance définitive ne s'était dégagée. Toutefois, nous avons constaté que, sur les instances à forte connectivité et dureté, l'heuristique `RANK`(`moyenne`) était plus pertinente.

11. Pour chaque valeur, nous maintenons un compteur mémorisant le nombre de fois que celle-ci a été affectée.

3.1.2 Heuristiques génériques de choix de voisinage

(Travail en collaboration avec Nicolas Levasseur et Patrice Boizumault ; publications associées : HM'08 [Levasseur *et al.*, 2008a] et EAAI'10 [Loudni *et al.*, 2010]).

Contexte. Une des difficultés des méthodes de recherche locale à grands voisinages est le choix pertinent du voisinage à explorer à chaque étape de la recherche. C'est le cas notamment pour la recherche `VNS`/`LDS`+`CP`, qui exploite une famille de voisinages formés de toutes les affectations obtenues en réaffectant k variables de l'affectation courante. Une approche traditionnelle consiste à sélectionner ces variables parmi celles qui participent à des violations. Cette heuristiques est simple et facile à mettre en œuvre. Toutefois, elle présente deux inconvénients majeurs : (*i*) elle ne tient pas compte de la topologie du graphe de contraintes et (*ii*) elle ne prend pas en compte les coûts de violation des fonctions de coût. En effet, elle peut sélectionner des variables indépendantes. Dans un tel cas, il est peu probable de les reconstruire sans violer plusieurs fonctions de coût.

Dans `PGLNS` [Perron *et al.*, 2004], les auteurs ont proposé une heuristique de choix de voisinage générique qui utilise le volume de propagation des contraintes pour définir le voisinage par *réduction*. L'intérêt de cette heuristique est d'éviter de choisir un ensemble de variables dont les valeurs vont être déduites à partir des variables restant affectées. Sa principale faiblesse est le grand nombre de propagations qu'elle effectue. De plus, sa dépendance vis à vis de l'efficacité de l'algorithme de filtrage basé sur les CSP, rend difficile son adaptation au cadre WCSP.

Contributions. En nous appuyons sur la notion de *degré de liberté* d'une variable, nous avons proposé d'exploiter l'hypergraphe d'un WCSP (cf. définition 2.8) pour sélectionner un voisinage « dense » offrant une plus grande opportunité d'optimisation des coûts. Puis, nous avons défini un cadre général permettant d'étendre ces heuristiques aux WCSP afin de tenir compte du coût des contraintes violées.

3.1.2a Heuristiques de choix de voisinage basées sur le degré de liberté

Définition 3.2 (Degré de liberté d'une variable [Loudni *et al.*, 2010])
Soient $\mathcal{A}$ une affectation complète, $\mathcal{X}_{un}$ un ensemble de variables à désaffecter et x une variable incluse dans $\mathcal{X}_{un}$. Le degré de liberté de x est égal au nombre de variables dans $Vois(x)$ incluses dans $\mathcal{X}_{un}$. Si $Vois(x) \subseteq X_{un}$, le degré de liberté de x est dit *maximal*.

La figure 3.1 illustre plusieurs choix effectués par les heuristiques que nous proposons. Les fonctions de coût satisfaites (resp. violées) par l'affectation complète courante sont toujours représentées par des arcs en pointillés (resp. pleins).

1. ConflictVar-Connected sélectionne aléatoirement la prochaine variable parmi celles en conflit et voisines des variables déjà sélectionnées (cf. figure 3.1a). La première variable est choisie aléatoirement parmi celles en conflit.

2. ConflictVar-Star sélectionne une première variable x_c (en noir) dite "variable centre" (cf. figure 3.1b), puis choisit aléatoirement les prochaines variables parmi celles en conflit dans $Vois(x_c)$ (i.e. en gris). Si toutes celles-ci sont sélectionnées, un centre est de nouveau déterminé parmi les variables en conflit voisines de celles déjà choisies (i.e. en pointillés).

3. ConflictVarAndSat-Star est une extension de ConflictVar-Star. Elle élargit, lorsque toutes les variables en conflit et voisines du centre ont été sélectionnées, le choix aux variables voisines non conflictuelles. Sur la figure 3.1c, la prochaine variable à être sélectionnée est celle en pointillés.

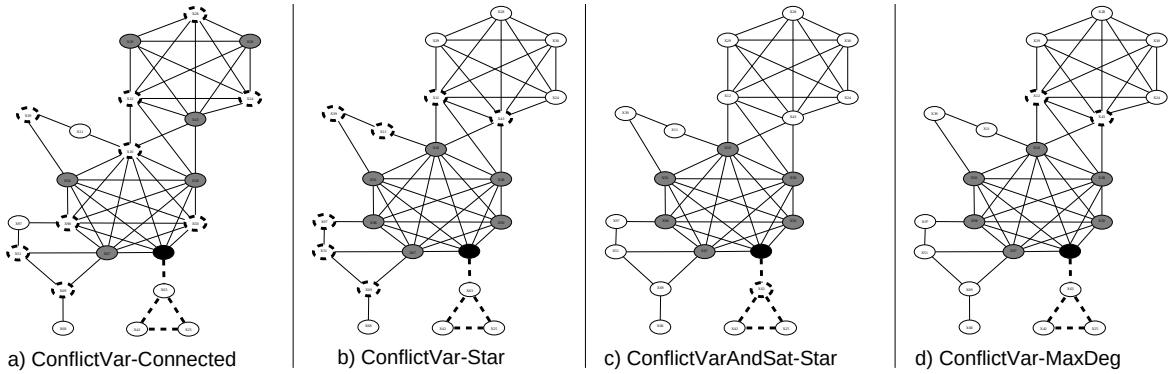


FIGURE 3.1 – Exemple de sélection de variables pour différentes heuristiques de choix de voisinage.

4. ConflictVar-MaxDeg choisit en premier une variable parmi celles en conflit. La prochaine variable est sélectionnée parmi celles (en conflit ou non) ayant le plus de variables voisines déjà sélectionnées (les égalités sont départagées aléatoirement). Sur la figure 3.1d chaque variable en pointillés a deux variables voisines sélectionnées, alors que les autres variables en ont au plus une. Notons que cette heuristique permet de sélectionner un ensemble de variables fortement connectées.

Prise en compte des coûts. Nous avons également proposé un cadre générique permettant d'étendre les heuristiques définies précédemment aux WCSP [Levasseur *et al.*, 2008c]. Pour une affectation complète courante $\mathcal{A}$, les coûts des fonctions violées par $\mathcal{A}$ sont d'abord triés par ordre décroissant de violation. $nbSets$ sous-ensembles, notés $ec_1, ec_2, ...ec_{nbSets}$, sont ensuite crées, avec $nbSets$ un paramètre à fixer au début de la recherche. Chaque sous-ensemble ec_i contient les $\lfloor (i*e)/nbSets \rfloor$ plus fortes violations. Au cours de la recherche, seules les variables qui participent à des fonctions ayant un coût de violation supérieur ou égal à $min(ec_b)$ sont considérées comme étant en conflit, la valeur de b augmentant proportionnellement avec la taille du voisinage k à considérer (pour plus de détail voir l'article [Loudni *et al.*, 2010]).

Bilan des expérimentations. Nous avons mené plusieurs expérimentations sur les instances Scen01 à Scen11 de `RLFAP` [12] [Cabon *et al.*, 1999] ainsi que sur les instances binaires WCSP aléatoires structurées par un arbre binaire de clusters kbtree [13]. Pour chaque heuristique H_i, chaque instance a été résolue avec la méthode `VNS`/`LDS`+`CP` utilisant H_i comme heuristique de choix de voisinage. Ces expérimentations montrent que :

- les heuristiques basées sur un degré de liberté plus grand sont nettement plus pertinentes.
- la sélection des variables non-conflictuelles est un critère important. En effet, ConflictVarAndSat-Star surpasse nettement ConflictVar-Star.
- l'extension [14] des heuristiques aux fonctions de coût permet d'améliorer les performances de celles-ci.

12. Il s'agit d'un benchmark issu d'une application militaire du Centre d'Electronique de l'Armement (CELAR). Le problème consiste à assigner un nombre limité de fréquences à un ensemble de liens radios entre des paires de sites, afin de minimiser les interférences dues à la réutilisation des fréquences.

13. Ces instances sont disponibles à l'adresse : http ://costfunction.org/en/benchmark

14. Lors des expérimentations, seules les extensions de ConflictVar et ConflictVar-Star ont été réalisées.

— les heuristiques les plus performantes sont ConflictVar-MaxDeg et ConflictVar-Star-Cost, avec un léger avantage pour ConflictVar-Star-Cost.

Discussion. L'intégration de la topologie du réseau de contraintes dans le choix des variables à réaffecter constitue une idée naturelle et pertinente, en particulier sur des instances fortement structurées. Les améliorations de performance très visibles sur les jeux de données testés, m'ont conduit à m'intéresser aux méthodes de décomposition de graphes (en arbre de clusters de largeur faible – décomposition en arbre ou Tree Decomposition) pour construire des heuristiques exploitant directement clusters et séparateurs de clusters. Ce travail a fait l'objet d'une thèse que j'ai co-encadrée [Fontaine, 2013] et dont les principaux résultats sont présentés à la section suivante.

3.2 Exploitation de la structure d'un réseau

Contexte. Comme nous l'avons indiqué en introduction, ma motivation vient du fait que plusieurs problèmes et benchmarks rencontrés lors de mes précédents travaux avaient un graphe fortement structuré (présence de groupes de variables formant des *clusters*, ces groupes n'étant eux-mêmes pas ou peu connectés entre eux). C'est le cas pour le problème *d'allocation de fréquence à des liens radio* (`RLFAP`) (cf. la figure 3.2b), le problème de *sélection des prises de vue d'un satellite d'observation terrestre* (`SPOT5`) [15] [Bensana *et al.*, 1999], ou encore le problème de la *sélection de marqueurs représentatifs d'une population d'individus en génétique* (`tagSNP`) [16] [Hirschhorn et Daly, 2005] (cf. la figure 3.2a). Les méthodes de décomposition de graphes en arbre de clusters (décomposition arborescente) semblent être de bons candidats pour exhiber ce type de structures. Les travaux menés jusqu'alors exploitaient les décompositions arborescentes uniquement dans le cadre des méthodes de recherche complètes. Exploiter une telle information dans les méthodes de recherche locale permettrait, d'une part, d'identifier des structures de voisinage pertinentes et d'autre part, de mieux guider l'exploration de l'espace de recherche.

Contributions. Dans ce contexte, nous avons proposé d'exploiter la structure du graphe sous-jacent à un réseau de contraintes pour : i) identifier des structures de voisinage pertinentes dans le contexte d'une recherche locale à voisinages variables (`VNS`), ii) mieux guider l'exploration des voisinages dans `VNS`. Dans cette section, je présente notre démarche à travers trois contributions :

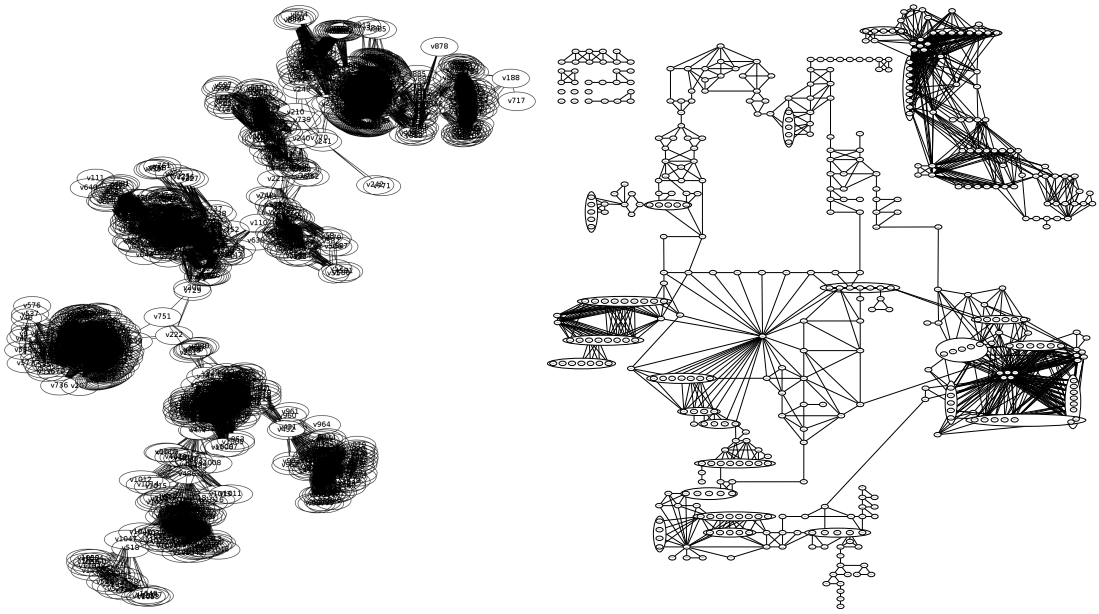
(1) un schéma générique (`DGVNS`), exploitant la décomposition arborescente (section 3.2.2) ;
(2) deux propositions d'amélioration de ce schéma (sections 3.2.3 et 3.2.4) ;
(3) plusieurs stratégies d'exploitation parallèle des clusters dans `DGVNS` (section 3.2.5).

3.2.1 Décomposition arborescente d'un graphe

La décomposition arborescente [Robertson et Seymour, 1986] vise à diviser un graphe en clusters et à organiser ces clusters sous la forme d'un arbre de jonction (ou arbre de clusters) du graphe original.

15. Il s'agit d'un problème fourni par le CNES et l'ONERA pour la planification quotidienne d'un satellite d'observation de la terre, `SPOT5`. Le problème consiste à sélectionner les prises de vue à effectuer dans la journée en prenant en compte les contraintes matérielles du satellite tout en maximisant les gains liés à la vente de ces prises de vue.

16. Le problème consiste à choisir un sous-ensemble de marqueurs biologiques `SNP`, appelé `tagSNP`, permettant de capturer le maximum d'information génétique. Ce problème est considéré comme très difficile du fait de sa proximité avec le problème de *set covering* (NP-difficile) [Sánchez *et al.*, 2009].



(a) Instance #14226 de `tagSNP` (b) Instance Scen08 de `RLFAP`

FIGURE 3.2 – Exemples de problèmes fortement structurés.

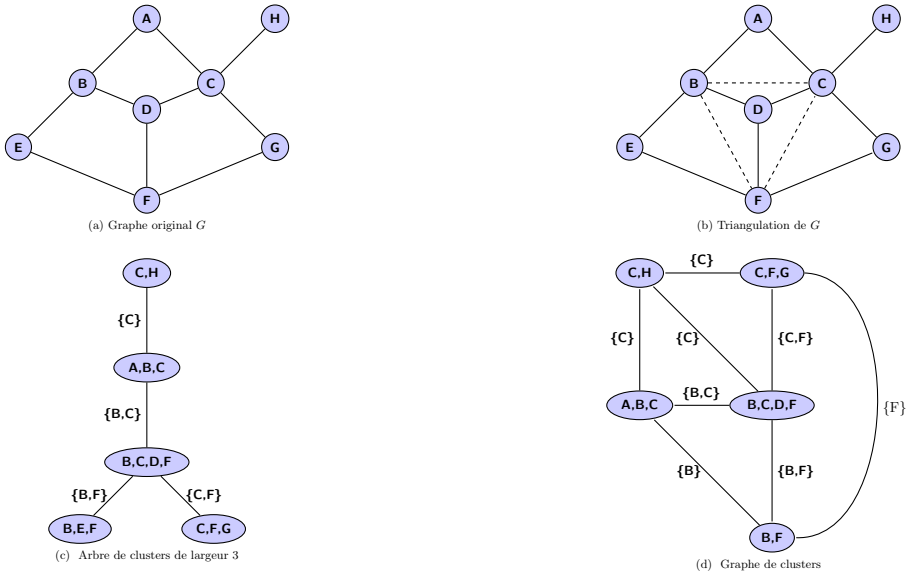
Définition 3.3 (DÉCOMPOSITION ARBORESCENTE D'UN WCSP)
Soit $G = (X, E)$ l'hypergraphe d'un WCSP $(X, D, W, \mathcal{S})$, on appelle décomposition arborescente de G un couple $(\mathcal{C}_T, T)$ où $T = (I, F)$ est un arbre avec I l'ensemble des sommets, F l'ensemble des arêtes et $\mathcal{C}_T = \{C_i \,|\, i \in I\}$ est un ensemble fini de clusters, chaque cluster étant un sous-ensembles de X qui vérifie :

— $\bigcup_{i \in I} C_i = X$;
— $\forall w_S \in W$, $\exists\, i \in I$ t.q. $S \subseteq C_i$;
— $\forall (i, j, k) \in I^3$, si j est sur le chemin de i à k dans T alors $C_i \bigcap C_k \subseteq C_j$.

Définition 3.4 (LARGEUR D'UNE DÉCOMPOSITION ARBORESCENTE, SÉPARATEUR)
Soit $(\mathcal{C}_T, T)$ une décomposition arborescente. Le paramètre $w^-(T) = \texttt{Max}_{i \in I}(|C_i| - 1)$ est appelé largeur de la décomposition. La largeur d'arbre (ou tree-width) d'un graphe est la plus petite largeur sur toutes ses décompositions arborescentes. L'ensemble des variables partagées entre deux clusters C_i et C_j, noté $sep(C_i, C_j)$, est appelé séparateur : $sep(C_i, C_j) = C_i \cap C_j$. On appelle variables propres de C_i l'ensemble des variables qui n'appartiennent qu'au cluster C_i.

Définition 3.5 (GRAPHE DE CLUSTERS)
Un graphe de clusters pour une décomposition arborescente (C_T, T), est un graphe non-orienté $G_T = (\mathcal{C}_T, E_T)$ dont les sommets sont les éléments de $\mathcal{C}_T$ et il existe une arête $(C_i, C_j) \in E_T$ entre les sommets C_i et C_j ssi $sep(C_i, C_j) \neq \emptyset$.

Calculer une décomposition de largeur minimale est un problème NP-difficile. Toutefois, ce problème peut être résolu en un temps polynomial si le graphe à décomposer est *triangulé* (i.e. tous ses cycles de taille supérieure à quatre ont une corde). On peut donc construire une décomposition arborescente d'un graphe non triangulé en produisant une triangulation de ce graphe. Plusieurs heuristiques reposant sur la notion de triangulation de graphe permettent



(a) Graphe original G (b) Triangulation de G (c) Arbre de clusters de largeur 3 (d) Graphe de clusters

FIGURE 3.3 – Exemple de décomposition d'un graphe et triangulation associée.

le calcul de décompositions approchées. Elles fournissent un majorant de la largeur d'arbre. Pour nos expérimentations, nous avons utilisé les heuristiques *maximum cardinality search* (`MCS`) [Tarjan et Yannakakis, 1984] et *minimum fill-in* (`min-fill`) [Yannakakis, 1981] car elles offrent clairement le meilleur compromis entre qualité de la décomposition et temps de calcul.

La figure 3.3 présente, pour un graphe G (figure 3.3a), un exemple de triangulation (figure 3.3b) et un exemple de décomposition arborescente de largeur 3 (figure 3.3c) associé au graphe de clusters de la figure 3.3d. Les arêtes entre les clusters sont étiquetées par l'ensemble des sommets communs aux clusters qu'elles relient.

3.2.2 `VNS` Guidée par la Décomposition (`DGVNS`)

(Travail en collaboration avec Mathieu FONTAINE et Patrice BOIZUMAULT ; publications associées : ICTAI'11 [Fontaine *et al.*, 2011], RAIRO [Fontaine *et al.*, 2013a] qui est une version étendue de l'article de JFPC'12 [Fontaine *et al.*, 2012], HM'13 [Loudni *et al.*, 2013b]).

Notre intuition est de sélectionner les variables à désinstancier dans un même cluster. En effet, toute affectation d'une variable du cluster impacte directement l'affectation des autres variables appartenant au même cluster. Afin de tirer parti du fort lien entre les variables d'un cluster, à la place des structures de voisinage N_k dans `VNS`/`LDS`+`CP` (cf. section 2.3.3), `DGVNS` utilise des structures de voisinage $N_{k,i}$ permettant de focaliser l'exploration de l'espace de recherche autour d'un cluster.

Définition 3.6 (STRUCTURE DE VOISINAGE $N_{k,i}$)
Soit G un graphe et G_T=($\mathcal{C}_T$,E_T) le graphe de clusters associé. Soit C_i un cluster de $\mathcal{C}_T$. La structure de voisinage $N_{k,i}$ désigne l'ensemble des combinaisons de k variables (k étant la dimension du voisinage) parmi C_i.

L'algorithme 7 décrit le pseudo-code de `DGVNS`. Ainsi, pour favoriser les mouvements dans

Algorithme 7 : Pseudo-code de `DGVNS`

```
1  Fonction DGVNS (X, W, k_init, k_max, δ_max);
2  début
3      Soit G le graphe de contraintes de (X, W) ;
4      Soit (C_T, T) une décomposition arborescente de G ;
5      Soit C_T = {C_1, C_2, ..., C_p} ;
6      S ← genSolInit(X, W) ;
7      k ← k_init ;
8      i ← 1 ;
9      tant que (k < k_max) ∧ (not TimeOut) faire
10         C_s ← CompleteCluster(C_i, k) ;
11         X_un ← Hneighborhood(N_{k,i}, C_s, W, S) ;
12         A ← S\{(x_i, a) | x_i ∈ X_un} ;
13         S' ← LDS + CP(A, X_un, δ_max, f(S), S) ;
14         NeighbourhoodChangeDGVNS(S, S', k, i);
15     retourner S ;
16 fin
```

des régions fortement liées, à chaque itération, les k variables à désaffecter sont sélectionnées au sein d'un même cluster C_i (ou C_i et ses clusters voisins dans le cas où C_i a moins de k variables) (ligne 10). En effet, le cluster est une structure qui permet d'identifier ces régions, du fait de sa taille (plus petite que le problème original), et du lien fort entre les variables qu'il contient. De plus, grâce à la forte connectivité entre les variables, l'étape de reconstruction pourra bénéficier d'un filtrage plus efficace et d'un calcul de minorant de bien meilleure qualité. Ensuite, un sous-ensemble de k variables est sélectionné dans le voisinage $N_{k,i}$ (i.e. l'ensemble des combinaisons de k variables parmi C_s) par la fonction `Hneighborhood` (ligne 11). Tout comme `VNS`/`LDS`+`CP`, une affectation partielle $\mathcal{A}$ est alors générée, puis reconstruite en utilisant une recherche arborescente partielle de type LDS, aidée par une propagation de contraintes (CP) sur les fonctions de coût (ligne 13). L'algorithme s'arrête dès que l'on a atteint la dimension maximale du voisinage à considérer k_{max} ou le *TimeOut* (ligne 15).

`DGVNS` est paramétré par la procédure `NeighborhoodChangeDGVNS` qui détermine, à chaque itération, le prochain cluster C_i (i.e. stratégie de changement de voisinage) en fonction de la qualité de la nouvelle solution S' (ligne 14). Nous avons proposé dans [Loudni *et al.*, 2013b] deux stratégies qui exploitent le graphe de clusters. Elles correspondent à deux schémas de déplacement entre clusters en fonction de l'amélioration ou non de la qualité de la solution courante.

i) `DGVNS`-1 : changer systématiquement de cluster. L'idée principale est de considérer successivement tous les clusters C_i. Soit p le nombre total de clusters, *succ* une fonction successeur, qui associe à un cluster i son successeur $C_{succ(i)}$ [17] et $N_{k,i}$ la structure de voisinage courante. Si aucune solution de meilleure qualité que la solution courante n'a été trouvée par `LDS+CP` dans le voisinage de S, `DGVNS`-1 cherche de nouvelles améliorations dans $N_{(k+1),succ(i)}$ (structure de voisinage où $(k+1)$ variables seront désaffectées). En effet, se déplacer du cluster C_i vers le cluster $C_{succ(i)}$ permet de favoriser l'exploration de nouvelles régions de l'espace de recherche et de tenter de trouver des solutions de meilleure qualité qui peuvent se trouver dans ces régions. De plus, quand un minimum local est atteint dans le voisinage courant, passer de k à $k+1$ permet

17. si $(i < p)$ alors $succ(i) = i + 1$ sinon $succ(p) = 1$.

de renforcer la diversification par l'élargissement de la dimension du voisinage. À l'inverse, si LDS+CP trouve une solution de meilleure qualité S' dans $N_{k,i}$, k est réinitialisé à k_{init} et le cluster suivant est considéré : changer de cluster permettra de diversifier l'exploration autour de S'.

ii) DGVNS-2 : changer de cluster en l'absence d'amélioration. Le principe de cette heuristique est de passer au cluster $C_{succ(i)}$ si aucune amélioration de la solution courante n'a été trouvée par LDS+CP. En effet, rester dans le même cluster en cas d'amélioration de la solution courante permettra de propager, durant la phase de reconstruction, les conséquences des nouvelles affectations des variables de S'. De plus réinitialiser k à k_{init} favorisera de petits mouvements dans le voisinage de S'. Ainsi, contrairement à DGVNS-1, DGVNS-2 effectue un compromis entre intensification et diversification. En effet, tant qu'aucune amélioration n'a été trouvée, DGVNS-2 considère successivement tous les clusters C_i (i.e. diversification), mais dès que la solution courante est améliorée, DGVNS-2 passe à l'intensification.

Bilan des expérimentations. Nous avons mené plusieurs expérimentations sur différentes instances (26 instances au total) de RLFAP, SPOT5 et tagSNP. Toutes les méthodes testées ont été implantées en C++ en utilisant la librairie toulbar2.

i) Apports de la décomposition arborescente. Afin de quantifier l'apport de la décomposition arborescente sur notre approche, nous avons comparé les résultats obtenus par DGVNS-1 avec ceux obtenus par VNS/LDS+CP et IDW [18]. Les résultats de nos expérimentations montrent clairement l'efficacité et la pertinence de DGVNS-1 comparé à VNS/LDS+CP et IDW sur des problèmes structurés tels que RLFAP et SPOT5. Sur les instances tagSNP, DGVNS surclasse clairement VNS/LDS+CP, particulièrement sur les instances de grande taille, où VNS/LDS+CP est incapable d'atteindre l'optimum.

ii) Apports des stratégies de changement de voisinages. Les résultats obtenus montrent clairement l'impact de la stratégie de changement de voisinage sur les performances de DGVNS. DGVNS-1 obtient un meilleur score sur le critère temps de calcul [19] (DGVNS-1 est plus rapide sur 65% des instances considérées). En termes de taux de succès [20], DGVNS-2 présente de meilleures performances : DGVNS-2 obtient les meilleurs taux de succès sur 19 instances contre 13 pour DGVNS-1. Clairement, DGVNS-2 réalise un meilleur compromis entre intensification et diversification.

3.2.3 Choix d'une bonne décomposition arborescente

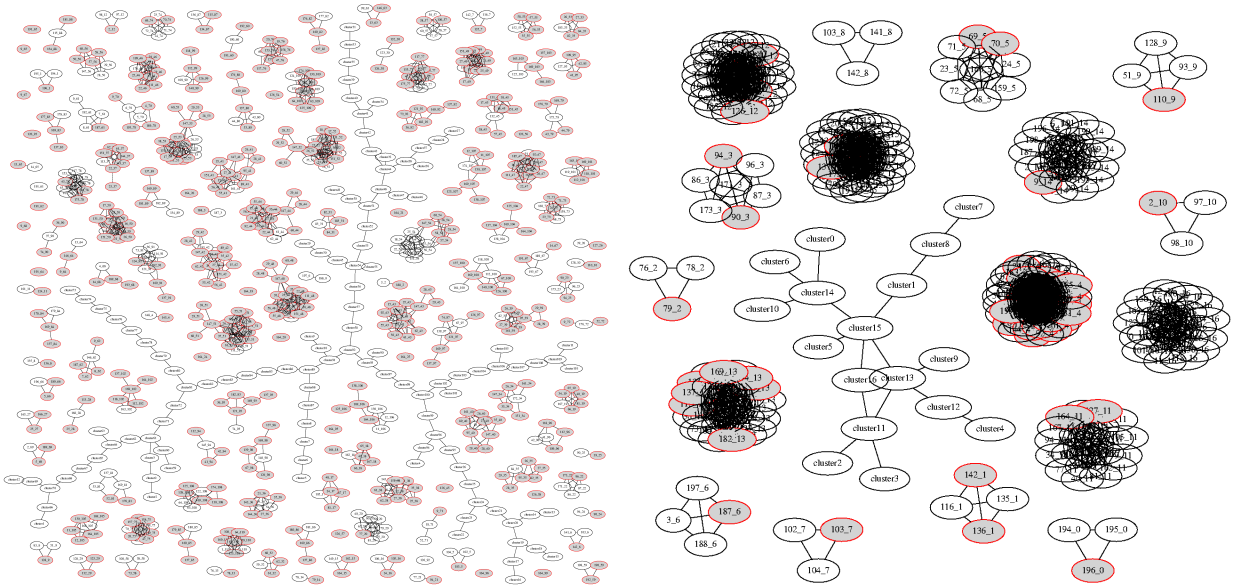
(Travail en collaboration avec Mathieu Fontaine et Patrice Boizumault ; Cf. la thèse de Mathieu Fontaine, ainsi que l'article à JFPC'13 [Fontaine *et al.*, 2013b]).

Une étude plus approfondie sur les décompositions arborescentes utilisées montre que les performances de DGVNS dépendent fortement de la qualité de la décomposition. En effet, les performances sont d'autant meilleures que lorsque le problème se décompose en clusters de tailles raisonnables avec des petits séparateurs. Les heuristiques de décomposition arborescente habituellement utilisées (MCS, min-fill) ont montré leurs limites pour identifier ce type de structures (i.e. présence de clusters faiblement reliés entre eux) en particulier sur les instances tagSNP. En

18. IDW (Intensification Diversification Walk) est une autre méta-heuristique, introduite dans [Neveu *et al.*, 2004], dédiée aux WCSP et qui propose une gestion de voisinage ressemblant aux listes de mouvements candidats (candidats lists) proposées dans [Glover et Laguna, 1997].

19. Temps de calcul moyen nécessaire à la méthode pour atteindre l'optimum sur l'instance considérée.

20. Le nombre d'essais avec succès (pour attendre l'optimum) obtenue par la méthode sur l'instance considérée.



(a) décomposition initiale (b) décomposition après fusion par absorption

FIGURE 3.4 – Instance scen07 de RLFAP.

effet, pour la plupart de ces instances, les décompositions obtenues comportent de nombreux clusters qui se chevauchent très fortement. De plus, ces clusters contiennent peu ou pas de variables propres. Cette forme de redondance entre clusters limite l'effort de diversification de DGVNS, en considérant plusieurs fois des voisinages très proches.

En s'appuyant sur ce constat, nous avons étudié deux schémas complémentaires permettant d'améliorer la décomposition arborescente. Le premier schéma, noté TDTD (*Tightness Dependent Tree Decomposition*), consiste à ne considérer au moment de la décomposition que les fonctions de coûts les plus dures (c'est-à-dire, celles ayant une proportion d'affectations valides inférieure à un seuil de dureté minimal λ). L'idée ici est d'essayer d'identifier les structures de voisinage portant sur les sous-parties du problème contenant des fonctions de coût les plus difficile à satisfaire, c'est-à-dire, toutes celles ayant une dureté supérieure au seuil λ. Notons que l'idée d'exploiter les coûts des fonctions a été aussi exploitée par [Kitching et Bacchus, 2009] dans le cadre des méthodes de recherche arborescente. Le second schéma, noté abs (fusion par absorption), consiste à fusionner les clusters partageant une grande proportion de variables. Dans ce qui suit, je vais détailler le second schéma.

Raffinement basé sur la fusion de clusters. Afin de limiter la redondance entre clusters et donc de renforcer la qualité des structures de voisinage explorées par DGVNS, nous avons proposé de fusionner les clusters qui sont redondants dans la décomposition arborescente. L'idée sous-jacente est d'augmenter la proportion de variables propres dans les clusters. À cet effet, nous avons introduit un nouveau critère appelé *absorption* [Fontaine *et al.*, 2013b] :

Définition 3.7 (ABSORPTION)
Soit $(\mathcal{C}_T, T)$ une décomposition arborescente, C_i et C_j deux clusters de $\mathcal{C}$. L'absorption de C_i par C_j, notée $\texttt{abs}(C_i, C_j)$, est définie par :

$$\texttt{abs}(C_i, C_j) = \frac{|sep(C_i, C_j)|}{|C_i|}$$

Soit $\texttt{abs}_{max}$ le taux d'absorption maximal autorisé, et min_{size} la taille minimale autorisée pour un cluster. La fusion s'effectue sur la décomposition initiale. En partant des feuilles de cette décomposition, tous les clusters $< C_i, parent(C_i) >$ ayant un taux d'absorption supérieur à $\texttt{abs}_{max}$ ou tels que $|C_i| < min_{size}$ sont fusionnés. Pour nos expérimentations, min_{size} a été fixée à la valeur 5 et $\texttt{abs}_{max}$ à 70%. Ainsi, les clusters partageant un peu plus de 2/3 de leurs variables seront fusionnés.

La figure 3.4a montre un exemple de décomposition arborescente de largeur 23 obtenue par MCS sur l'instance CELAR scen07. Cette décomposition est constituée de 110 clusters et la taille de son plus grand séparateur vaut 18. Comme on peut le remarquer, la plupart des clusters ont plus de 90% de leurs variables partagées (cf. ellipses [21] en rouge). La figure 3.4b donne la nouvelle décomposition obtenue après fusion par absorption. Comme on peut le constater, la fusion par absorption permet de supprimer la plupart des clusters redondants et de générer une décomposition contenant au moins 50% de variables propres. Par ailleurs, la taille du plus grand cluster (resp. séparateur) est égale à 49 (resp. 8). Ainsi, la fusion par absorption permet de produire des décompositions de bonne qualité : des clusters de tailles raisonnables avec une forte proportion de variables propres.

Dans [Jégou *et al.*, 2006, Sánchez *et al.*, 2009], les auteurs proposent de fusionner les clusters partageant plus de variables qu'un seuil maximal fixé s_{max} : en partant des feuilles de la décomposition initiale, fusionner les clusters C_i qui partagent plus de s_{max} variables avec leur parent. On obtient ainsi une nouvelle décomposition, dans laquelle aucun séparateur ne contient plus de s_{max} variables. Nous avons pu observer, qu'en pratique, le réglage du paramètre s_{max} dépend du gain en réduction de la taille des séparateurs et donc du nombre de variables partagés par rapport à la perte en augmentation de la taille des clusters résultants. En effet, pour des valeurs de s_{max} élevées, la fusion modifie peu la décomposition initiale. En revanche, pour des valeurs de s_{max} faibles, les décompositions produites ont peu de clusters et possèdent des largeurs de décomposition très élevées.

Bilan des expérimentations. Pour mesurer l'apport de la fusion de clusters, nous avons comparé les performances de DGVNS-1, DGVNS-2 sur les décompositions initiales (sans fusion) ainsi que sur les décompositions raffinées obtenues par fusion de clusters selon les deux critères s_{max} et abs. Nous avons considéré différentes valeurs pour s_{max} : $4, 8, 12, 16, 24$ et 32. Sur l'ensemble des instances, la fusion de clusters par absorption procure des gains substantiels comparés aux résultats obtenus par DGVNS-1 et DGVNS-2 sur la décomposition initiale de MCS. Ces résultats confirment l'intérêt de réduire la redondance entre clusters pour identifier des structures de voisinage plus pertinentes et ainsi renforcer l'effort de diversification de DGVNS. Par ailleurs, la fusion par absorption permet de produire des décompositions ayant une forte proportion de variables propres. De plus, la taille des clusters reste raisonnablement élevée. En effet, sur la plupart des instances, la taille des clusters est en moyenne 2 fois plus grande comparée à la décomposition initiale. De plus, le nombre de clusters reste relativement petit.

Vers une approche exploitant une décomposition dynamique. Une piste prometteuse qui mériterait d'être étudiée est de changer dynamiquement de décomposition durant la recherche. Notre idée est d'adapter la décomposition arborescente en fusionnant dynamiquement certains

21. Chaque ellipse contient un numéro de la forme x_y, avec x le numéro de la variable et y son numéro du cluster.

clusters en fonction de critères qui peuvent dépendre de l'état de la recherche ou des informations apprises lors des précédentes explorations.

3.2.4 Exploitation des séparateurs

(Travail en collaboration avec Mathieu Fontaine et Patrice Boizumault ; publications associées : ENDM'12 [Loudni *et al.*, 2012] et JFPC'13 [Loudni *et al.*, 2013a]).

`DGVNS` est paramétré par la procédure `NeighborhoodChangeDGVNS` qui détermine, à chaque itération, le prochain cluster C_i à considérer. Ce choix est réalisé de façon déterministe selon un ordre fixé arbitrairement au début de la recherche. Nous avons également étudié l'intérêt de guider le choix de ce prochain cluster C_i, en prenant en compte les variables séparateurs. Nous avons proposé deux heuristiques `SGVNS` (pour Separator-Guided `VNS`) et et `ISGVNS` (pour Intensified Separator-Guided `VNS`). L'idée commune aux deux heuristiques est d'utiliser une liste de clusters à visiter en priorité, et de mettre à jour cette liste à chaque fois qu'une solution de meilleure qualité est trouvée, en augmentant la priorité des clusters partageant des variables avec le cluster courant C_i. Les deux heuristiques diffèrent essentiellement dans leur façon de mettre à jour la liste de clusters à chaque fois qu'une solution de meilleure qualité est trouvée [Loudni *et al.*, 2012, Loudni *et al.*, 2013a].

Separator-Guided `VNS`. `SGVNS` exploite l'évolution de la qualité de la solution au cours de la recherche afin de guider l'effort de diversification de `DGVNS` vers des clusters contenant, dans leurs séparateurs, **au moins une variable réaffectée** impliquée dans l'amélioration de la solution courante. La différence principale par rapport au schéma de `DGVNS`, réside principalement dans l'utilisation d'une liste de clusters à visiter en priorité, notée $\mathcal{C}and$. Au départ, cette liste est initialisée par les clusters de la décomposition arborescente. Soit C_i le cluster courant et S la solution courante. À chaque fois qu'une solution de meilleure qualité est trouvée dans le voisinage de la solution courante S, `SGVNS` remet en tête de liste les clusters de la liste partageant des variables avec C_i et ayant conduit à l'amélioration de la solution courante. À l'inverse, si aucune amélioration de la solution courante n'a été trouvée, la dimension du voisinage k est incrémentée et `SGVNS` considère le cluster courant (tête de la liste $\mathcal{C}and$).

Intensified Separator-Guided `VNS`. Comme pour `SGVNS`, `ISGVNS` utilise également une liste de clusters à visiter en priorité, notée P_{List}. Toutefois, contrairement à `SGVNS`, `ISGVNS` insère en queue de P_{List} l'ensemble des clusters partageant des variables avec C_i et ajoute une liste de variables taboues assurant que les dernières variables modifiées ne seront pas reconsidérées dans le voisinage dans $N_{k,i}$ par la fonction `Hneighborhood` lors des L prochaines itérations. La valeur de L est fixée à la taille de P_{List}. Enfin, le prochain cluster à examiner correspond au premier élément de P_{List}, si celle-ci n'est pas vide. Dans le cas contraire, le successeur de C_i dans $\mathcal{C}_T$ est considéré.

Bilan des expérimentations. Les résultats obtenus montrent que ces deux heuristiques permettent d'améliorer les performances de `DGVNS` sur la majorité des instances testées. Les deux heuristiques obtiennent des résultats comparables avec toutefois un léger avantage pour `ISGVNS`.

3.2.5 Exploitation parallèle des clusters dans `DGVNS`

(Travail en collaboration avec Abdelkader Ouali, Lakhdar Loukil, Patrice Boizumault, et Yahia Lebbah ; publications associées : HM'14 [Ouali *et al.*, 2014] et ENDM'15 [Ouali *et al.*, 2015]).

Contexte. Exploiter le parallélisme et les architectures multi-cœurs est un moyen naturel pour accélérer les calculs et résoudre des problèmes combinatoires complexes de grande dimension. Récemment, il y a eu un grand succès en calcul parallèle dans le domaine de l'optimisation combinatoire. Plusieurs versions parallèles de nombreuses métaheuristiques connues ont été proposées avec différents degrés de parallélisation. Dans ce contexte et à la lumière de la prochaine génération d'architectures d'ordinateurs, qui sous peu contiendront des dizaines de cores, la conception d'approches multicore est une perspective de recherche très prometteuse. Une approche naturelle consiste à diviser l'espace de recherche en sous-espaces, puis à explorer les différents sous-espaces en parallèle. Cependant, dans ce type d'approches, définir des mécanismes efficace pour le partage et la communication d'informations entre différents processus n'est pas une tâche aisée. Un autre moyen qui a fait ses preuves en particulier dans la résolutions des problèmes SAT, est l'utilisation de portfolio de solveurs en parallèle afin d'obtenir un meilleur solveur globale, évitant ainsi tout type de communication.

Contributions. Exploiter la décomposition arborescente pour paralléliser l'exploration des clusters dans `DGVNS` est donc une idée naturelle. Dans ce cadre, nous avons proposé trois stratégies de parallélisation pour `DGVNS`. Elles diffèrent essentiellement par la façon dont la communication s'effectue entre le maître et les esclaves (synchrone ou asynchrone) et par la fréquence des solutions échangées (communiquer les meilleures solutions intermédiaires au cours de la recherche ou seulement à la fin pour identifier la meilleure solution globale). Ces contributions se résument aux points suivants :

1. Proposition d'une première stratégie de parallélisation de `DGVNS` nommée *Cooperative parallel* `DGVNS` (`CPDGVNS`) [Ouali *et al.*, 2014] basée sur une architecture maître-esclaves,
2. Proposition de deux autres stratégies de parallélisation, nommées *Replicated Asynchronous* `DGVNS` (`RADGVNS`) et *Replicated Synchronous* `DGVNS` (`RSDGVNS`) [Ouali *et al.*, 2015], destinées à améliorer la diversification de `CPDGVNS`.

Cooperative parallel decomposition guided `VNS`. L'idée de `CPDGVNS` consiste simplement à explorer tous les clusters fournis par une décomposition arborescente en parallèle. L'architecture retenue est de type *maître-esclaves*, où le processus maître mémorise, met à jour et communique la meilleure solution courante, les processus esclaves gèrent l'exploration des clusters individuels. Les processus individuels coopèrent de façon *asynchrone* en échangeant des informations sur la meilleure solution courante. Ceci garantit l'indépendance des processus esclaves individuels et permet de démarrer à partir de plusieurs solutions initiales différentes, favorisant ainsi une meilleure diversification. Pour profiter pleinement de la parallélisation, nous avons fixé le nombre de processus esclaves npr [22] au nombre de clusters issus de la décomposition. Le processus maître procède en trois étapes :

i) Étape d'initialisation. Le maître initie la recherche en lançant l'exécution en parallèle de npr processus esclaves et en envoyant à chaque processus esclave p la même solution initiale, le cluster associé C_c de $\mathcal{C}_T$ et les valeurs des paramètres k_{init}, k_{max} et δ_{max}. La liste $\mathcal{C}_T$ des clusters est gérée en FIFO pour assurer que tout cluster soit traité par un seul processus esclave. Afin de restreindre le choix des variables à désaffecter uniquement aux variables du cluster, k_{max} est initialisée à la taille du cluster affecté au processus esclave.

22. Si le nombre de cœurs est inférieur au nombre de clusters, un même cœur sera utilisé pour traiter différents clusters.

ii) Étape de mise à jour. Durant cette phase, le maître attend la meilleure solution trouvée par chaque processus esclave. Soit S'_p la meilleure solution communiquée par le processus esclave p au maître et S la meilleure solution globale. Si S'_p est de meilleure qualité que S, S'_p devient la meilleure solution globale, le prochain cluster C_c est considéré et k_{max} est réinitialisé à $|C_c|$. Sinon, on cherche de nouvelles améliorations dans le prochain cluster C_c et on incrémente de un le nombre de clusters adjacents adj qui doivent être considérés ainsi que la valeur de k_{max} par le cardinal de l'union des clusters C_j adjacents à C_c (avec $j = 1..adj$). L'augmentation de k_{max} permettra d'élargir l'ensemble de variables candidates à désaffecter par les processus esclaves.

iii) Étape d'intensification. Le but de cette étape est de relancer un processus esclave p à partir de la meilleure solution globale disponible. Cette étape est exécutée si le $TimeOut$ global n'est pas atteint. Sinon, il est arrêté. La résolution se termine lorsque tous les processus esclaves terminent.

Le processus esclave suit le schéma de `DGVNS`. Mais contrairement à ce dernier, le changement de cluster est géré par le maître. Par ailleurs, la stratégie de changement de voisinage est similaire à celle de la méthode `VNS`/`LDS`+`CP` (cf. Algorithme 4). Il nécessite une décomposition arborescente ($\mathcal{C}_T$,T) de G. Il reçoit du maître, l'indice c du cluster qui lui est assigné, les valeurs des paramètres k_{init}, k_{max}, δ_{max}, et la solution initiale S. L'algorithme s'arrête dès qu'il a atteint la dimension maximale du voisinage k_{max} ou un $TimeOut$ local [23].

Stratégies de réplication parallèles pour `DGVNS`. Dans `CPDGVNS`, chaque processus esclave effectue une recherche `DGVNS` dans le cluster qui lui est assigné. La principale limite de cette approche est que les esclaves doivent effectuer un certain nombre d'itérations avant de partager leurs meilleures solutions avec le maître. Cela rend la coopération avec le maître moins fréquente, et limite la diversification de l'exploration de l'espace de recherche par les esclaves. Pour pallier à cet inconvénient, nous avons proposé deux nouvelles stratégies de parallélisation, `RADGVNS` et `RSDGVNS`, permettant de produire rapidement des solutions intermédiaires pour alimenter l'échange d'informations entre les esclaves.

i) Replicated Asynchronous `DGVNS`. Les processus individuels dans `RADGVNS` coopèrent de manière *asynchrone* en échangeant des informations sur la meilleure solution courante. Toutefois, et contrairement à `CPDGVNS`, chaque processus esclave exécute une seule itération d'une recherche locale, en complétant une solution partielle dans le cluster qui lui est associé. Comme pour `CPDGVNS`, le maître attend les nouvelles solutions trouvées par chaque processus esclave p. Si une nouvelle solution S'_p est reçue, elle est comparée à la meilleure solution globale courante S. Mais contrairement à `CPDGVNS`, le changement de cluster dans `RADGVNS` s'effectue plus rapidement (i.e. à chaque fois qu'une solution de moins bonne qualité est renvoyée par l'esclave), tandis que `CPDGVNS` nécessite beaucoup plus de temps avant de changer de cluster.

ii) Replicated synchronous `DGVNS`. Pour étudier l'impact de la synchronisation, nous avons proposé sur la même architecture une version synchrone de la communication entre le processus maître et les processus esclaves. Dans `RSDGVNS`, le processus maître attend la fin de tous les processus esclaves avant d'effectuer le prochain mouvement.

Bilan des expérimentations. Les expérimentations ont été réalisées sur les instances les plus difficiles des problèmes `RLFAP`, `SPOT5`, `tagSNP` et `GRAPH` [24], soit un total de 15 instances. Nous

23. Dans nos expérimentations, le $TimeOut$ local a été fixé à $(global\_TimeOut)/npr$.
24. Instances aléatoires ayant une structure proche des instances `RLFAP`.

avons utilisé un cluster Linux Infiniband de 8 nœuds, chaque nœud est constitué d'un dual-CPU Xeon E5-2650 de 16 cœurs (soit au total 128 processus). Toutes les méthodes ont été implantées en C++ en utilisant la librairie `toulbar2`. La parallélisation a été mise en œuvre dans l'environnement MPI (Message Passing Interface). Quatre méthodes ont été comparées : `DGVNS`, `CPDGVNS`, `RADGVNS` et `RSDGVNS`. Pour chaque méthode, différentes décompositions arborescentes ont été considérées :

(i) décomposition arborescente obtenue par l'heuristique `MCS` et sa version raffinée obtenue par fusion de clusters avec l'absorption,

(ii) décomposition arborescente obtenue par l'heuristique `min-fill` et sa version raffinée obtenue par fusion de clusters avec l'absorption.

Les résultats de nos expérimentations démontrent clairement la supériorité de `RADGVNS` par rapport à `DGVNS`, `CPDGVNS` et `RSDGVNS` sur les différentes décompositions considérées. Cela atteste de l'importance d'échanger souvent la meilleure solution globale entre les esclaves pour améliorer la diversification. Ces résultats montrent également que `RSDGVNS` est la deuxième meilleure stratégie parallèle. Enfin, les décompositions arborescentes raffinées permettent d'améliorer significativement les performances de nos approches parallèles.

3.3 Relaxation de contraintes globales

Je présente dans cette section une synthèse de nos travaux portant sur la définition d'un cadre unifié de relaxations (orienté variables et orienté décomposition) pour différentes contraintes globales existantes, permettant de prendre en compte de manière fine des préférences où des coûts de violation des contraintes.

Tout d'abord, je définis les notions de contrainte globale (cf. section 3.3.1) et de contrainte globale relaxée (cf. section 3.3.2). Ensuite, je présente deux exemples de contraintes globales `Gcc` et `Regular` ainsi que leurs versions relâchées avec préférences (cf. sections 3.3.3 et 3.3.4). Enfin, je montre en section 3.3.5 l'apport de notre cadre pour la modélisation et la résolution des problèmes d'emplois du temps de personnel hospitalier (NRPs).

3.3.1 Contraintes globales

Les contraintes rencontrées dans les applications réelles sont souvent des contraintes « métiers » qui ont une structure et une sémantique particulière, qu'il est utile d'exploiter pour fournir des algorithmes de filtrage dédiés plus efficace que les algorithmes de filtrage génériques. C'est le cas notamment des *contraintes globales* qui encapsulent un ensemble de propriétés faisant intervenir plusieurs, voire toutes les variables du problème.

Définition 3.8 (Contrainte globale)
Une *contrainte globale*, dénotée par $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$, est une contrainte ayant une sémantique précise qui permet d'exprimer des propriétés impliquant a priori un nombre arbitraire de variables X et pouvant inclure d'éventuels paramètres additionnels représentés par $\theta_1, \ldots, \theta_m$.

L'apport des contraintes globales ne situe pas uniquement d'un point de vue modélisation, en procurant une modélisation élégante. En effet, en exploitant la sémantique de la contrainte, des algorithmes de filtrage très performant peuvent être développés. La cohérence maintenue est souvent globale.

Définition 3.9 (Cohérence globale)
Une contrainte globale $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$ est dite *globalement cohérente ssi* pour toute valeur (x_i, a_j) il existe une instanciation complète $\mathcal{A} \in D^X$ et $\mathcal{A}[x_i] = a_j$ telle que GC soit satisfaite.

Établir la cohérence globale pour GC revient donc à éliminer toutes les valeurs des domaines des variables de X qui ne peuvent pas être étendues en un tuple autorisé par GC. La définition 3.10 permet de caractériser la viabilité d'une valeur, c-à-d quand une valeur donnée peut participer à une solution satisfaisant une contrainte.

Définition 3.10 (Viabilité d'une valeur (pour la cohérence globale))
Une valeur (x_i,a_j) est viable *ssi* pour toute contrainte $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$, ayant x_i dans sa portée, il existe une instanciation complète des variables de X étendant l'affectation $(x_i = a_j)$ et satisfaisant la contrainte GC.

L'efficacité des algorithmes de filtrage des contraintes globales repose bien souvent sur l'utilisation de modèles et algorithmes issus d'autres domaines de l'informatique comme par exemple la programmation linéaire, la théorie des graphes, les flots, etc. L'utilisation de ces modèles et algorithmes, déjà très étudiés par ailleurs, permet l'écriture de tests de cohérence et d'algorithmes de filtrage très efficaces en temps et peu coûteux en espace. La conception de tels algorithmes est au cœur de nos travaux sur la relaxation de certaines contraintes globales et l'extraction de motifs séquentiels (voir section 5.4).

3.3.2 Contraintes globales relaxées sans préférences

Pour capturer l'idée de *violation* (ou relaxation) d'une contrainte globale, la notion de *contrainte globale molle* (ou relaxée) a été introduite dans [Petit *et al.*, 2001]. Il s'agit d'une contrainte globale classique (dure) avec une variable de coût z qui quantifie la violation la contrainte selon une fonction de coût μ associée.

Définition 3.11 (Contrainte globale relaxée [Allouche *et al.*, 2015])
Une contrainte globale relaxée, dénotée $\textsc{Soft\_GC}(X \cup \{z\}, \mu, \theta_1, \ldots, \theta_m)$, est définie par une contrainte globale $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$ et une fonction $\mu : D^X \rightarrow \mathbb{R}^+$ telle que $\forall \mathcal{A} \in D^X$, $\mu(\mathcal{A}) = 0$ ssi $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$ est satisfaite. μ représente la *mesure de violation* associée à GC et z la *variable de coût* qui quantifie le degré de violation de GC.

Mesures de violation. D'après la définition 3.11, pour une même contrainte globale, il existera généralement plusieurs contraintes globales relaxées selon la mesure de violation retenue. Deux mesures de violation génériques ont été proposées dans [Petit *et al.*, 2001] :

— la mesure de violation basée variable,
— la mesure de violation basée décomposition.

i) Mesure de violation basée variable. La violation engendrée par une instanciation dépend du nombre de variables qu'il est nécessaire de réaffecter pour satisfaire cette contrainte.

Définition 3.12 (mesure de violation basée variable [Petit *et al.*, 2001])
Soit $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$ une contrainte globale, la mesure de violation basée variable μ_{var} quantifie pour une affectation le nombre minimal de variables de sa portée dont il faudrait changer la valeur pour que GC soit satisfaite.

ii) Mesure de violation basée décomposition. Cette mesure exploite la décomposition d'une contrainte globale en contraintes élémentaires (si elle existe) pour quantifier la violation d'une affectation.

Définition 3.13 (Mesure basée décomposition [Petit *et al.*, 2001])
Soit $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$ une contrainte globale et sa décomposition en contraintes élémentaires. La mesure de violation basée décomposition μ_{dec} quantifie pour une instanciation le nombre de contraintes binaires insatisfaites par la décomposition de GC.

Cohérence et filtrage. Quelque soit la contrainte globale GC et la mesure de violation μ, il est possible de définir le test de cohérence et l'algorithme de filtrage maintenant la cohérence globale de la façon suivante.

i) Test de cohérence. Soit $\textsc{Soft}\_\mathrm{GC}(X \cup \{z\}, \mu, \theta_1, \ldots, \theta_m)$ la version relâchée de $\mathrm{GC}(X, \theta_1, \ldots, \theta_m)$ selon μ. La contrainte $\textsc{Soft}\_\mathrm{GC}(X \cup \{z\}, \mu, \theta_1, \ldots, \theta_m)$ admet une solution ssi il existe une instanciation complète $\mathcal{A}$ des variables de X telle que $\mu(\mathcal{A}) \leq max(D_z)$.

ii) Filtrage. Le principe de l'algorithme de filtrage est de générer pour chaque valeur (x_i, a_j) un sous-problème dans lequel le domaine de la variable x_i est réduit à la valeur a_j et de tester si une solution existe pour ce sous-problème. Si ce n'est pas le cas, alors il est possible de filtrer cette valeur. La complexité temporelle d'un tel algorithme est en $O(m \times TC)$, avec TC la complexité pour déterminer l'existence d'une solution et $m = \sum_{x_i \in X} |D_i|$, la somme des cardinaux des domaines. Ainsi, s'il est possible de connaître l'existence d'une solution en temps polynomial, alors il est possible de maintenir la cohérence globale en temps polynomial [Bessiere *et al.*, 2007].

L'idée de la relaxation de contraintes globales a été ensuite étendue dans [Hoeve *et al.*, 2006] et a permis de définir plusieurs versions relâchées de contraintes globales (`soft-AllDifferent`, `soft-Gcc`, `soft-Regular`, etc) permettant de tirer bénéfice à la fois de la structure et de la sémantique de la contrainte ainsi que de sa mesure de violation pour fournir des algorithmes de filtrage très efficace. Toutefois, les différentes mesures de violation proposées manquaient d'expressivité pour modéliser de problèmes réels. En effet, ces propositions considéraient que toutes les violations sont de même importance (seul leur nombre importe). Dans la section suivante, nous étudions la relaxation de certaines contraintes globales dans un cadre où il est possible d'exprimer des préférences. Les algorithmes de propagation associés à ces contraintes sont obtenus par des représentations fines en terme de flots.

3.3.3 Relaxation de `Gcc` avec préférences

(Travail en collaboration avec Jean-Philippe Métivier et Patrice Boizumault ; publications associées : SAC'09 [Métivier *et al.*, 2009a]).

La contrainte globale de cardinalité `Gcc`. Cette contrainte impose qu'un ensemble de variables prenne ses valeurs de façon à respecter des bornes inférieures et supérieures sur le nombre de fois que ces valeurs peuvent êtres prises.

Définition 3.14 (`Gcc` (X, l, u) [Régin, 1996])
Soit $X = \{x_1, ..., x_n\}$ un ensemble de variables et $Doms$ l'union des domaines de X. Soit $a_j \in Doms$ et l_j et u_j les bornes inférieures et supérieures de a_j. `Gcc` $(X,[l_1, ..., l_m],[u_1, ..., u_m])$ admet une solution *ssi* il existe une affectation complète des variables de X telle que :

$$\forall a_j \in Doms,\ l_j \leq |\{x_i \in X \mid x_i = a_j\}| \leq u_j$$

La contrainte Gcc se modélise par un réseau de flot pour lequel l'existence d'un flot maximal de valeur n permet de tester la cohérence [Régin, 1996]. La recherche des composantes fortement connexes dans le graphe résiduel permet de filtrer toutes les valeurs non viables (i.e., les valeurs associées aux arcs n'appartenant à aucune composante connexe). Le test de cohérence se fait en $O(n \times m)$ [Ford et Fulkerson, 1956] et le filtrage en $O(m + n)$ [Tarjan, 1972] (avec n le nombre de sommets et m le nombre d'arcs du réseau).

Enfin, toute contrainte Gcc (X, l, u) peut se décomposer en une conjonction de contraintes atleast (X, a_j, l_j) et atmost (X, a_j, u_j) portant sur les valeurs de $Doms$ [Hentenryck *et al.*, 1992] :

$$\texttt{Gcc}(X, l, u) \equiv \bigwedge_{a_j \in Doms} (\texttt{atleast}(X, a_j, l_j) \wedge \texttt{atmost}(X, a_j, u_j))$$

La contrainte globale Σ-Gcc**.** La contrainte Σ-Gcc $(X \cup \{z\}, \mu^{\Sigma}_{dec}, l, u, \varphi^{atleast}, \varphi^{atmost})$ est la version relâchée de Gcc (X, l, u) selon la mesure de violation basée décomposition avec préférences μ^{Σ}_{dec}. Cette nouvelle contrainte autorise le non respect des bornes inférieures et supérieures moyennant un coût de violation qui est fonction de l'écart aux bornes et du poids associé à celles-ci.

À chaque valeur a_j est associé un poids $\varphi^{atleast}_j$ (resp. φ^{atmost}_j) à la borne inférieure l_j (resp. supérieure u_j). Ainsi, il est possible de distinguer les violations associées aux différentes valeurs d'une même contrainte Gcc (X, l, u), mais aussi entre le manque et l'excès associé à une même valeur. Le calcul du manque (resp. l'excès) se fait grâce à la fonction $s(X, a_j)$ (resp. $e(X, a_j)$).

$$s(X, a_j) = max(0, l_j - |\{x_i \mid x_i \in X, x_i = a_j\}|)$$
$$e(X, a_j) = max(0, |\{x_i \mid x_i \in X, x_i = a_j\}| - u_j)$$

La violation de la contrainte Σ-Gcc pour la mesure basée décomposition avec préférences μ^{Σ}_{dec} est définie par : $\mu^{\Sigma}_{dec}(X) = \sum_{a_j \in Doms} \Big(s(X, a_j) \times \varphi^{atleast}_j + e(X, a_j) \times \varphi^{atmost}_j\Big)$.

Une contrainte Σ-Gcc $(X \cup \{z\}, \mu^{\Sigma}_{dec}, l, u, \varphi^{atleast}, \varphi^{atmost})$ admet une solution *ssi* il existe une instanciation complète $\mathcal{A}$ telle que : $\mu^{\Sigma}_{dec}(\mathcal{A}) \leq max(D_z)$. La modélisation de cette contrainte s'effectue en ajoutant au réseau de Gcc des arcs de violation traduisant le manque ou l'excès pour chaque valeur de $Doms$.

L'existence d'un flot faisable de poids minimal inférieur à $max(D_z)$ dans le nouveau réseau permet de caractériser la cohérence de Σ-Gcc. De la même manière, on peut caractériser la viabilité d'une valeur (x_i, a_j) par l'existence d'un flot passant par l'arc (x_i, a_j) de poids inférieur à $max(D_z)$.

Remarque 1. *Il est à noter qu'ici le flot faisable n'est pas nécessairement de valeur n. En effet, il est possible que les demandes ne puissent être comblées grâce aux affectations des variables (soit les variables ne sont pas assez nombreuses, soit la structure du réseau rend impossible la satisfaction de toutes les bornes). Nous donnons ici une borne supérieure sur la valeur du flot : $n + \sum_{a_j \in Doms} l_j$.*

Nous avons également proposé une seconde mesure basée variable avec préférences μ^{Σ}_{var}. Chaque variable $x_i \in X$ reçoit un poids φ_i reflétant l'importance de celle-ci. μ^{Σ}_{var} mesure alors la somme minimale des poids des variables qu'il est nécessaire de réaffecter pour satisfaire la contrainte `Gcc` (X, l, u). La contrainte Σ-`Gcc` $(X \cup \{z\}, \mu^{\Sigma}_{var}, l, u, \varphi)$ est la version relâchée de la contrainte globale `Gcc` (X, l, u) selon la mesure de violation basée variable avec préférences μ^{Σ}_{var}.

Comme précédemment, la modélisation de Σ-`Gcc` selon la mesure μ^{Σ}_{var} s'effectue en rajoutant au réseau de `Gcc` des arcs de violation qui vont servir à représenter les réaffectations des variables de la contrainte. Test de cohérence et filtrage sont mis en œuvre de la même manière que pour la mesure μ^{Σ}_{dec}.

3.3.4 Relaxation de la contrainte `Regular` avec préférences

La contrainte globale `Regular`. Soit Π=$(Q, \Sigma, \delta, q_0, F)$ un automate fini déterministe, avec Q un ensemble fini d'états, Σ un alphabet, $\delta : Q \times \Sigma \to Q$ un ensemble de transitions, q_0 un état initial et $F \subseteq Q$ un ensemble d'états finaux. La contrainte `Regular` (X, Π) associée à l'automate Π impose qu'un mot constitué d'une séquence de n variables appartienne au langage reconnu par l'automate Π.

Définition 3.15 (`Regular` (X, Π) [Pesant, 2004a])
Soit Π=$(Q, \Sigma, \delta, q_0, F)$ un automate fini déterministe, $\mathcal{L}(\Pi)$ le langage régulier associé à Π et X une séquences de n variables. La contrainte `Regular` (X, Π) admet une solution ssi $\exists A \in D_{x_1} \times ... \times D_{x_n}$ t.q. $A \in \mathcal{L}(\Pi)$.

La contrainte `Regular` (X, Π) est modélisée par un graphe en couches dont les sommets de chaque couche correspondent aux états de l'automate Π et les arcs représentent les transitions de l'automate [Pesant, 2004a]. Le graphe en couches est défini de la manière suivante :

$$
\begin{array}{l}
V = \{s\} \cup V_0 \cup ... \cup V_n \cup \{t\} \\
A = A_s \cup A_1 \cup ... \cup A_n \cup A_t \\
\text{avec} \\
\forall i \in [0..n],\ V_i = \{q^i_l \mid q_l \in Q\} \\
A_s = \{(s \to q^0_0)\} \\
\forall i \in [1..n],\ A_i = \{(q^{i-1}_l \to q^i_m, a_j) \mid a_j \in D_{x_i} \text{ et } \delta(q_l, a_j) = q_m\} \\
A_t = \{(q^n_l \to t) \mid q_l \in F\}
\end{array}
$$

Dans ce graphe, tout chemin du sommet représentant l'état initial q^0_0 dans la première couche vers un sommet modélisant un état final q^n_i (avec $q_i \in F$) dans la dernière couche correspond à une solution pour `Regular` (X, Π). De la même manière, on peut caractériser la viabilité d'une valeur (x_i, a_j) par l'existence d'un tel chemin passant par l'arc (x_i, a_j). Le test de cohérence et le filtrage sont mis en œuvre par un parcours en largeur du graphe en $O(n \times |Q| \times |\Sigma|)$.

La contrainte globale Σ-`Regular`. La contrainte Σ-`Regular` $(X \cup \{z\}, \Pi, W, \mu^{\Sigma}_{hm})$ est la version relâchée de la contrainte `Regular` (X, Π) selon la mesure de violation μ^{Σ}_{hm} et correspondant à la version pondérée de la *distance de Hamming*. Cette nouvelle contrainte modélise le fait que certaines transitions inattendues de l'automate engendrent un coût si elles sont utilisées. Ainsi, à chaque substitution d'un symbole a par un symbole b est associé un poids φ_{ab} reflétant le coût de celle-ci.

Soit W une matrice contenant le coût associé aux substitutions d'un symbole par un autre, la distance de Hamming pondérée, notée $hm^w(m_1, m_2)$, mesure pour deux mots de même longueur, la somme des poids φ_{ab} des symboles qui différent à une même position (le symbole a appartenant au mot m_1 et le symbole b à m_2). La violation de la contrainte Σ-`Regular` pour la mesure μ^{Σ}_{hm} est définie par : $\mu^{\Sigma}_{hm}(X) = min\{hm^w(D, X) \mid D = D_{x_1} \times ... \times D_{x_n} \text{ t.q. } D \in L(\Pi)\}$.

La contrainte Σ-`Regular` $(X \cup \{z\}, \Pi, W, \mu^{\Sigma}_{hm})$ admet une solution *ssi* il existe une instanciation complète $\mathcal{A}$ telle que : $\mu^{\Sigma}_{hm}(\mathcal{A}) \leq max(D_z)$. La modélisation de cette contrainte s'effectue en ajoutant au graphe en couches de `Regular` un arc de violation pour chaque valeur $a_k \in D_{x_i} \backslash \{a_j\}$ traduisant la substitution de a_j par a_k. Le poids de cet arc est φ_{a_j,a_k}.

L'existence d'un chemin de s à t de poids inférieur ou égal à $max(D_z)$ dans le nouveau graphe en couches permet de caractériser la cohérence de Σ-`Regular`. De la même manière, on peut caractériser la viabilité d'une valeur (x_i, a_j) par l'existence d'au moins un chemin reliant s à t dans le graphe en couches associé de poids inférieur ou égal à $max(D_z)$ et utilisant un arc étiqueté par a_j reliant les couches V_{i-1} et V_i. Le test de cohérence et le filtrage sont mis en œuvre par un parcours en largeur du graphe en $O(n \times |Q| \times |\Sigma|)$ dans le pire cas.

Exprimer Σ-`Regular` avec un `costRegular`. `costRegular` (X, Π, z) est une version pondérée de `Regular` (X, Π) qui permet de modéliser le fait que certaines transitions de l'automate sont pondérées. D'un point de vue modélisation, il suffit de reporter les poids de ces transitions directement sur les arcs associés dans le graphe en couches de `Regular` (X, Π) [Demassey *et al.*, 2006]. L'existence d'un s–t chemin de poids inférieur ou égal à $max(D_z)$ permet de tester la cohérence de la contrainte.

Soit l'automate Π associé à la contrainte Σ-`Regular` $(X \cup \{z\}, \Pi, W, \mu^{\Sigma}_{hm})$. Il est possible d'encoder de manière plus élégante la mesure de violation μ^{Σ}_{hm} à l'aide de la contrainte `costRegular` (X, Π, z), en ajoutant à l'automate initial Π les transitions représentant les différentes substitutions possibles : pour chaque transition $\delta(q_l, a_j) = q_m$ de Π et pour chaque symbole $a_k \in \Sigma \backslash \{a_j\}$ (Σ étant l'alphabet utilisé par Π), on ajoute une nouvelle transition $\delta(q_l, a_k) = q_m$ de poids égal à φ_{a_j,a_k}. Soit Π' ce nouvel automate. Ainsi, la relaxation de `Regular` (X, Π) selon la mesure de violation μ^{Σ}_{hm} s'obtient en imposant la contrainte `costRegular` sur Π' [Métivier *et al.*, 2009b].

3.3.5 Applications des contraintes globales relaxées aux NRPs

Le problème de planification d'emplois du temps d'infirmières (NRPs : *Nurse Rostering Problem*) consiste à affecter, pour chaque infirmière, une équipe (*shift*) sur une période donnée (*planning horizon*) tout en respectant des règles spécifiant les besoins de l'hôpital et la législation du travail.

En raison de leur complexité et de leur importance dans les hôpitaux modernes, les NRPs ont été largement étudiés en Recherche Opérationnelle (RO) et en Intelligence Artificielle (IA) depuis plus de 40 ans. De tels problèmes sont généralement difficiles à résoudre car de grande taille et sur-contraints [auf'm Hofe, 2000, Qu et He, 2008]. En effet, un grand nombre de règles différentes et des préférences spécifiques des infirmières doivent être satisfaites pour garantir un planning de bonne qualité dans la pratique. D'un point de vue résolution, de nombreuses approches ont été proposées pour résoudre les `NRP`s. Toutefois, la plupart des ces approches nécessitent des heuristiques ad'hoc.

Contribution. Sur ce problème éminemment sur-contraint et complexe, nous avons montré l'intérêt des contraintes globales relaxées pour offrir une méthode générique de résolution, avec deux avantages principaux : (i) la modélisation de ces problèmes est concise et élégante ; (ii) l'apport d'une approche générique vs les approches ad'hoc développées depuis une quarantaine d'années pour résoudre chacun de ces problèmes. L'annexe A donne un exemple de modélisation CSP de l'instance `GPOST`.

L'utilisation de méthodes de recherche locale comme `VNS/LDS+CP` a permis de résoudre efficacement plusieurs instances réelles difficiles du site ASAP (Automated Scheduling, Optimisation and Planning) de l'université de Nottingham (`http://www.cs.nott.ac.uk/~tec/NRP/`). Les résultats expérimentaux montrent que notre approche est compétitive avec des approches complètement ad'hoc.

3.4 Fonctions de coût globales décomposables

(Travail en collaboration avec David Allouche, Christian Bessière, Patrice Boizumault, Simon de Givry, Patricia Gutierrez, Jean-Philippe Métivier et Thomas Schiex ; publications associées : JFPC'12 [Allouche *et al.*, 2012b] et AAAI'12 [Allouche *et al.*, 2012a]).

Récemment, Jimmy H.M. Lee et Ka Lun Leung [Lee et Leung, 2012] ont montré pour la première fois que les techniques de filtrage dédiées aux WCSP peuvent être combinées avec les mécanismes de filtrage des contraintes globales relaxées, pour définir des *fonctions de coût globales* qui peuvent s'exprimer sous la forme d'un problème de flot maximum de coût minimum dans un graphe dirigé particulier. Ce travail a ouvert la voie à plus d'expressivité du formalisme WCSP.

Dans le cadre du projet ANR FICOLOFO, nous avons introduit la notion de *décomposition de fonctions de coût globales* [Allouche *et al.*, 2012a] et nous avons montré comment les contraintes globales décomposables pouvaient être relâchées sous la forme de fonctions de coût globales décomposables, ayant la même structure de décomposition. L'objectif était de permettre une intégration simplifiée des contraintes globales (relaxées) dans les réseaux de fonctions de coût tout en exploitant des contraintes de bas niveau.

Il existe un moyen simple de dériver les fonctions de coûts décomposables à partir des contraintes globales décomposables connues, en relâchant chacune des contraintes issues de la décomposition. Ainsi, de la même manière que pour le cadre de la relaxation de contraintes globales, il est possible de dériver des fonctions de coût décomposables à partir des décompositions de contraintes globales existantes. Pour bien illustrer ce principe, nous allons prendre comme exemple la contrainte globale `Among` [Beldiceanu et Contejean, 1994a].

La contrainte globale `Among` restreint le nombre d'occurrences de certaines valeurs dans une séquence de n variables [Beldiceanu et Contejean, 1994a].

Définition 3.16 (`Among` [Beldiceanu et Contejean, 1994a])
Soient $X = \langle x_1, x_2, ..., x_n \rangle$ une séquence de n variables, V un ensemble de valeurs et D^X le produit cartésien des domaines des variables de X. Soient l et u deux entiers tel que $0 \leq l \leq u \leq n$, la contrainte $\texttt{Among}(X, V, l, u)$ impose que chaque valeur de V doit apparaître au moins l fois et au plus u fois dans X :

$$\texttt{Among}(X, V, l, u) = \{\mathcal{A} \in \mathcal{D}^X \mid l \leq \mid \{i, \mathcal{A}[x_i] \in V\} \mid \leq u\}$$

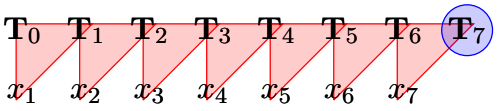


FIGURE 3.5 – L'hypergraphe d'une décomposition Berge-acyclique de `Among` sur une séquence de $n = 7$ variables.

Cette contrainte peut se décomposer en un réseau de contraintes $(X \cup \{T_0, ..., T_n\}, \mathcal{C})$, où les variables additionnelles T_i ont pour domaine D_{T_i}=[0..i]. Chaque variable T_i représente le nombre de fois qu'une valeur de V est prise par la séquence de variables $\{x_1, ..., x_i\}$. L'ensemble des contraintes $\mathcal{C}$ contient deux contraintes unaires c_0 et c_n et n contraintes ternaires de la forme $c_{\{T_{i-1},x_i,T_i\}}$. La contrainte unaire c_0 force T_0 à prendre la valeur 0, alors que la contrainte unaire c_n impose que $l \leq T_n \leq u$. Chaque contrainte ternaire encode les sommes cumulées sur les variables supplémentaires. L'hypergraphe associé à la décomposition est donné par la figure 3.5.

En exploitant les fonctions de manque et d'excès introduites en section 3.3.3, il est possible de relaxer cette décomposition comme suit :

— pour $i = 0$,

$$w_{\{T_0\}}(\mathcal{A}) = \begin{cases} 0 & \text{si } T_0 = 0 \\ \top & \text{sinon} \end{cases}$$

— pour $i \in [1..n]$,

$$w_{\{T_{i-1},x_i,T_i\}}(\mathcal{A}) = \begin{cases} 0 & \text{si } \mathcal{A}[x_i] \in V \text{ and } T_i = T_{i-1} + 1 \\ 0 & \text{si } \mathcal{A}[x_i] \notin V \text{ and } T_i = T_{i-1} \\ \top & \text{sinon} \end{cases}$$

— Soit $\ell(\mathcal{A}) = |\{i, \mathcal{A}[x_i] \in V\}|$,

$$w_{\{T_n\}}(\mathcal{A}) = \texttt{Max}(0, l - \ell(\mathcal{A}), \ell(\mathcal{A}) - u)$$

Dans les CSP classiques, il est bien connu que si une contrainte globale admettait une décomposition *Berge-acyclique*, l'établissement de la cohérence globale (GAC) sur la décomposition établissait également GAC sur la contrainte globale (appelée également version *monolithique*) [Beeri *et al.*, 1983]. Enfin, dans [Allouche *et al.*, 2012a], nous avons montré que le filtrage par T-DAC d'une décomposition Berge-acyclique d'une fonction de coût ou de la fonction de coût globale d'origine produisaient le même minorant $w_\emptyset$.

3.5 Conclusion

Dans ce chapitre, j'ai présenté nos contributions portant sur la résolution des WCSP. Ce problème est NP-difficile et différentes méthodes complètes, à base de recherche arborescente, ou approchées, à base de recherche locale, ont été proposées pour le résoudre. L'idée d'exploiter la décomposition arborescente du graphe de contraintes pour mieux guider le choix des voisinages dans les méthodes de recherche de type `VNS` est pertinente et originale. Elle constitue à mes yeux une avancée importante dans la résolution approchée des WCSP.

De façon complémentaire au formalisme WCSP, nous avons défini un cadre unifié de relaxations pour différentes contraintes globales, permettant de prendre en compte de manière fine des préférences et des coûts de violation des contraintes globales. Les algorithmes de propagation associés à ces contraintes sont obtenus par des représentations fines en termes de flots. La modélisation d'un problème réel éminemment sur-contraint et complexe a procuré une jolie justification à ce cadre. Enfin, bien que la relaxation de contraintes globales est un sujet très étudié depuis 2006, il s'agit de la toute première véritable application dans ce domaine.

Deuxième partie

Programmation par contraintes pour la fouille de données

Chapitre 4

Contexte : Extraction de motifs ensemblistes et de motifs séquentiels

La découverte de connaissances à partir des données est un domaine de recherche important pour de nombreuses applications stratégiques. Son caractère fondamental s'accentue avec la disponibilité de plus en plus croissante de gisements de données issues de divers domaines (e.g. web, télécommunications, biologie).

Les méthodes dites de fouille de données à base de motifs ont pour objectif la découverte d'informations nouvelles à partir de bases de données [Agrawal *et al.*, 1993]. Le cœur du processus de fouille est la recherche de régularités dans les données ou motifs cachés. Par exemple, à partir de situations biologiques décrites par un ensemble de gènes, un motif est un ensemble de gènes qui se retrouvent fréquemment dans de nombreuses situations biologiques.

Outre l'extraction de motifs ensemblistes, la fouille de données présente aussi l'intérêt de découvrir des régularités dans des données se présentant sous forme de séquences ; on appelle motifs séquentiels de telles régularités [Agrawal et Srikant, 1995]. Nous verrons que cette dimension est aussi au cœur de nos contributions.

Plan du chapitre. Tout d'abord, j'introduis les notions et notations liées à l'extraction de motifs ensemblistes. Ensuite, j'aborde la problématique d'extraction d'ensembles de motifs, c'est à dire des motifs fondés sur des motifs locaux. Enfin, je termine par un rappel des principales notions liées à l'extraction de motifs séquentiels.

4.1 Fouille de motifs ensemblistes

Cette section introduit les termes et notions classiquement utilisées en extraction de motifs dans un cadre ensembliste, c'est-à-dire sans prendre en compte une relation d'ordre entre les données.

4.1.1 Cadre formel et définitions

La version de base de l'extraction de motifs fréquents permet de faire la fouille dans une relation (table) d'une base données transactionnelle dont les valeurs sont des booléens (indiquant

la présence ou l'absence d'une propriété) :

Définition 4.1 (Base de données transactionnelle)
Une base de données transactionnelle $\boldsymbol{r}$ est un triplet $(\mathcal{I}, \mathcal{R}, \mathcal{T})$ où :
— $\mathcal{I} = \{i_1, \ldots, i_n\}$ est l'ensemble des attributs (items),
— $\mathcal{T} = \{t_1, \ldots, t_m\}$ est l'ensemble des objets (transactions),
— $\mathcal{R}$ une relation sur $\mathcal{T} \times \mathcal{I}$ qui permet d'indiquer si l'objet t possède l'attribut i (noté $t\,\mathcal{R}\,i$).

Considérant la base transactionnelle $\boldsymbol{r}$ de l'exemple 4.1 où 6 transactions étiquetées $t_1, \ldots, t_6$ sont décrites par 5 items $A, B, \ldots, E$. Dans cet exemple, chaque transaction appartient à une seule classe et est étiquetée par un item de classe (soit j_1, soit j_2). On notera $\mathcal{T}_1$ l'ensemble des transactions de classe 1 et $\mathcal{T}_2$ celui de classe 2.

Exemple 4.1. Base de données transactionnelle

Classe	$\mathcal{T}$ \ $\mathcal{I}$	A	B	C	D	E
1	t_1	1	0	1	1	0
	t_2	0	1	1	0	1
	t_3	1	1	1	0	0
2	t_4	0	1	0	0	1
	t_5	1	1	1	0	1
	t_6	0	1	1	0	1

$\mathcal{T}$	Classe	Items
t_1	j_1	A C D
t_2	j_1	B C E
t_3	j_1	A B C
t_4	j_2	B E
t_5	j_2	A B C E
t_6	j_2	B C E

Table 4.1 – Deux représentations de la base de données transactionnelle $\boldsymbol{r}$.

L'objectif majeur de l'extraction de motifs est la découverte de relations utiles entre les objets de $\mathcal{T}$, chaque objet étant décrit par un sous-ensemble des items de $\mathcal{I}$. Ces relations, qui traduisent un comportement ou rendent compte d'un phénomène dans les données, reposent sur des *motifs* (aussi appelés *itemsets* dans le cas des données transactionnelles).

Définition 4.2 (Motif, taille)
Un *motif ensembliste d'items* ou *itemset* est un sous ensemble de $\mathcal{I}$. La *taille* d'un motif p est le nombre d'items que p contient (i.e. sa cardinalité) : $\texttt{taille}(p) = |p|$. On dira qu'une transaction $t \in \mathcal{T}$ supporte un motif p si, $\forall i \in p$, $t\,\mathcal{R}\,i$ (ou $p \subseteq t$).

Définition 4.3 (Couverture)
Étant donné un motif p et un ensemble de transactions $\mathcal{T}$, la couverture de p est l'ensemble des transactions qui le supportent : $\texttt{cover}_{\mathcal{T}}(p) = \{t \in \mathcal{T} \mid \ p \subseteq t\}$

Définition 4.4 (Langage)
Le langage de motifs $\mathcal{L}_{\mathcal{I}}$ est l'espace de recherche des motifs construit sur l'alphabet $\mathcal{I}$, autrement dit c'est l'ensemble des parties de $\mathcal{I}$ (i.e. $2^{\mathcal{I}} \setminus \emptyset$).

Pour la base transactionnelle de l'exemple 4.1, nous avons donc :
— 5 motifs de taille 1 : A, B, C, D et E.
— 10 motifs de taille 2 : AB, AC, AD, AE, BC, BD, BE, CD, CE et DE.
— 10 motifs de taille 3 : ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE et CDE.
— 5 motifs de taille 4 : $ABCD$, $ABCE$, $ABDE$, $ACDE$ et $BCDE$.

— 1 motif de taille 5 : $ABCDE$.

Parmi cet ensemble de $2^{|\mathcal{I}|} - 1$ motifs, on s'intéresse à ceux qui apparaissent fréquemment :

Définition 4.5 (Fréquence)
La fréquence absolue[25] d'un motif p est le cardinal de sa couverture : $\texttt{freq}_{\mathcal{T}}(p) = |\texttt{cover}_{\mathcal{T}}(p)|$.

4.1.2 Extraction de motifs sous contraintes locales

Le processus d'extraction de motifs peut retourner une collection trop importante de motifs pour être exploitée. L'extraction de motifs sous contraintes est généralement considéré comme une solution pour améliorer à la fois la qualité des motifs extraits et le processus de fouille. Les contraintes peuvent être définies sur des motifs individuels, on parle de contraintes *locales* et les motifs extraits sont appelés *motifs locaux* [Hand, 2002], ou bien elles requièrent l'utilisation d'autres motifs, on parle de contraintes *d'ensembles de motifs* (cf. section 4.1.5).

Un exemple classique de contrainte locale est la contrainte des *motifs fréquents* qui permet de sélectionner les motifs dont la fréquence est supérieure à un seuil donnée $minfr$ (i.e. $\texttt{freq}_{\mathcal{T}}(p) \geq minfr$). La contrainte de taille minimale spécifie la taille minimale (en nombre d'items) des motifs extraits (i.e. $\texttt{taille}(p) \geq \omega$).

Exemple 4.2. Dans l'exemple du tableau 4.1, pour $minfr = 3$, les motifs fréquents sont (nous indiquons entre $\langle . \rangle$ les valeurs de fréquence) : A $\langle 3 \rangle$, B $\langle 5 \rangle$, C $\langle 5 \rangle$, E $\langle 4 \rangle$, AC $\langle 3 \rangle$, BC $\langle 4 \rangle$, BE $\langle 4 \rangle$, CE $\langle 3 \rangle$ et BCE $\langle 3 \rangle$.

Outre les mesures de fréquence et de taille, il existe d'autres mesures d'intérêt portant sur des motifs locaux [Geng et Hamilton, 2006], comme *l'aire* d'un motif, définie par le produit de sa fréquence par sa taille, ou encore le *taux de croissance* d'un motif qui est une mesure classique de recherche de contraste [Novak *et al.*, 2009] (i.e., motifs dont la fréquence varie fortement d'un jeu de données à un autre).

4.1.3 Espace de recherche et représentations condensées de motifs

Le problème d'extraction de motifs est une tâche difficile du point de vue algorithmique de par la grande taille de l'espace de recherche. Pour les motifs ensemblistes, pour $|\mathcal{I}| = n$, la taille de l'espace de recherche est $2^n - 1$.

Une relation de spécialisation [Mitchell, 1982] structure le langage $\mathcal{L}_{\mathcal{I}}$ et facilite la recherche des motifs.

Définition 4.6 (Spécialisation)
Une relation de spécialisation $\preceq$ est un ordre partiel défini sur les motifs de $\mathcal{L}_{\mathcal{I}}$. Un motif p_1 est plus *spécifique* qu'un motif p_2 si $p2 \preceq p_1$.

Un exemple classique de *relation de spécialisation* est l'inclusion ensembliste $\subseteq$ entre deux motifs ensemblistes. Si un motif p_2 est inclus dans un motif p_1, on dit que p_1 est plus spécifique que p_2.

La principale propriété exploitée par la plupart des algorithmes d'extraction de motifs pour réduire l'espace de recherche est la propriété *d'anti-monotonie*.

25. Quand la fréquence est utilisée pour exprimer en pourcentage la fraction d'occurrences relativement au nombre total d'objets, on parlera alors de fréquence relative.

Définition 4.7 (Contrainte anti-monotone)
Une contrainte c est dite anti-monotone, si et seulement si, pour tout motif p satisfaisant c, tous ses sous-motifs satisfont également c.

L'(anti-)monotonie fournit une importante propriété d'élagage de l'espace de recherche :

Proposition 4.1 (Élagage fondé sur les contraintes (anti-)monotones)
Si un motif p ne satisfait pas une contrainte anti-monotone, alors aucune spécialisation de p ne pourra la vérifier.

Autrement dit, si un motif p ne satisfait pas une contrainte anti-monotone c, il est alors certain que toutes ses spécialisations ne vérifient pas c et l'exploration de l'espace de recherche peut être stoppée à partir de p. La contrainte de fréquence minimale est une contrainte anti-monotone.

Les *représentations condensées* de motifs ont été introduites pour rendre efficace le processus d'extraction et réduire le nombre de motifs obtenus. Il existe deux types de représentations condensées : les représentations exactes et celles dites approximatives. Avec une représentation exacte, il est possible de régénérer pour chaque motif les valeurs exactes des mesures associées à c (par exemple la fréquence dans le cas des motifs fréquents), alors que dans le cas d'une représentation condensée approximative seule une approximation de cette valeur peut être dérivée.

Pour les motifs ensemblistes et la contrainte de fréquence minimale, les représentations condensées les plus classiques sont celles fondées sur les *motifs fermés* [Pasquier *et al.*, 1999], les *motifs libres* [Boulicaut *et al.*, 2000] et les *motifs maximaux* [Gouda et Zaki, 2005]. Dans ce qui suit, nous nous intéressons uniquement aux motifs fermés et aux motifs maximaux.

Définition 4.8 (Motif fermé fréquent)
Un motif p est dit fermé, noté $\texttt{fermé}_{\mathcal{T}}(p)$, si et seulement si, il est fréquent et que tous ses sur-ensembles stricts ont une fréquence strictement inférieure à celle de p :

$$\texttt{fermé}_{\mathcal{T}}(p) \iff \texttt{freq}_{\mathcal{T}}(p) \geq minfr \ \wedge \ \forall i \in \mathcal{I} \setminus p : \texttt{freq}_{\mathcal{T}}(p \cup \{i\}) < \texttt{freq}_{\mathcal{T}}(p)$$

Tous les motifs ayant la même fermeture ont la même fréquence. Ainsi, la fermeture forme des classes d'équivalence de fréquence. En ne retenant qu'un représentant par classe d'équivalence, cette représentation conserve peu de motifs et est dite condensée. De plus, elle est exacte. En effet, la fréquence d'un motif quelconque est celle du plus petit fermé le contenant.

Définition 4.9 (Motif maximal fréquent)
Un motif p est dit maximal, noté $\texttt{maximal}_{\mathcal{T}}(p)$, si et seulement si, il est fréquent et que tous ses sur-ensembles sont peu fréquents :

$$\texttt{maximal}_{\mathcal{T}}(p) \iff \texttt{freq}_{\mathcal{T}}(p) \geq minfr \ \wedge \ \forall i \in \mathcal{I} \setminus p : \texttt{freq}_{\mathcal{T}}(p \cup \{i\}) < minfr$$

Cette représentation est approximative car elle ne permet pas de dériver la fréquence exacte de chaque motif fréquent non maximal, mais seulement une borne inférieure.

4.1.4 Modélisation CSP de l'extraction de motifs ensemblistes

Les travaux fondateurs proposés par [De Raedt *et al.*, 2008] ont ouvert la voie à des méthodes déclaratives proposant de modéliser et de résoudre des problèmes d'extraction de motifs sous contraintes à l'aide de la programmation par contraintes. Cette modélisation repose sur la représentation d'une tâche d'extraction via deux ensembles de variables de domaine binaire :

— l'ensemble $\{X_1, .., X_n\}$ qui d'écrit les items présents dans le motif recherché p (avec n le nombre d'attributs du jeu de données) :

$$\forall i \in \mathcal{I},\ (X_i = 1)\ \ ssi\ \ (i \in p)$$

— l'ensemble $\{T_1, .., T_m\}$ qui décrit l'ensemble des transactions couvrant ce motif (avec m le nombre de transactions du jeu de données) :

$$\forall t \in \mathcal{T},\ (T_t = 1)\ \ ssi\ \ (p \subseteq t)$$

En plus de ces deux ensembles de variables, un ensemble de contraintes réifiées [26] permettant de lier les deux ensembles de variables à la base de données transactionnelle $\boldsymbol{r}$ est ajouté. Ces contraintes permettent d'établir le lien entre les motifs recherchés et leur couverture :

$$\forall t \in \mathcal{T}, (T_t = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} X_i \times (1 - d_{t,i}) = 0 \tag{4.1}$$

L'équation 4.1 traduit le fait qu'une variable T_t ne vaudra 1 et donc que la transaction t ne sera support de p que si et seulement si tous les X_i correspondant à des items n'appartenant pas à cette transaction valent 0 et donc que ces items n'appartiennent pas au motif p.

Une fois ce lien entre les variables et la base transactionnelle réalisé, il devient naturel d'exprimer des tâches de fouille en ajoutant des contraintes pour modéliser celles-ci. Par exemple, la contrainte de fréquence minimale $\texttt{freq}(p) \geq minfr$ (où $minfr$ est un seuil) se formule par :

$$\sum_{t \in \mathcal{T}} T_t \geq minfr \tag{4.2}$$

celle de taille maximale (par rapport à un seuil Ω) par :

$$\sum_{i \in \mathcal{I}} X_i \leq \Omega \tag{4.3}$$

La contrainte $\texttt{fermé}(p)$ impose que le motif p soit fermé par rapport à la fréquence :

$$\forall i \in \mathcal{I},\ (X_i = 1)\ \Longleftrightarrow\ \sum_{t \in \mathcal{T}} T_t(1 - d_{t,i}) = 0 \tag{4.4}$$

L'équation 4.4 correspond à la contrainte de couverture de la transposée de la base, c'est-à-dire que poser une contrainte de fermeture revient à poser une contrainte de couverture en considérant les transactions comme des items et les items comme des transactions.

Discussions. En opposition à la fouille d'itemsets sous contraintes classique, la PPC utilisant cet encodage présente plusieurs avantages comme la possibilité d'utiliser des solveurs génériques ou la déclarativité et l'expressivité de la modélisation des contraintes. Elle dispose néanmoins de faiblesses puisqu'elle est plus lente et plus gourmande en espace à cause des m contraintes réifiées de taille au plus $n + 1$ dont le maintien entraîne une forte détérioration du temps de calcul.

26. Une contrainte réifiée permet d'associer une variable booléenne à une contrainte donnée.

4.1.5 Motifs fondés sur des motifs locaux

Une autre voie très prometteuse pour réduire le nombre de motifs extraits est la découverte des motifs de plus haut niveau reposant sur des caractéristiques qui impliquent plusieurs motifs locaux et donnant au final un sens global à l'ensemble de motifs qui est retourné. L'idée sous-jacente est d'exprimer l'intérêt d'un motif en fonction des autres motifs extraits.

Dans [Zimmermann, 2009], une caractérisation de l'extraction d'ensemble de motifs (appelés *pattern sets*) sous forme de contraintes est proposée. Ces contraintes sont définies comme étant des *contraintes sur des ensembles de motifs.* L'intérêt exprimé par la contrainte porte ici sur un *ensemble* de motifs (par exemple, couverture d'un ensemble de motifs, ...).

Il existe de nombreux exemples de motifs qui peuvent être définis avec des contraintes d'ensembles de motifs comme par exemple les top-k motifs [Fu *et al.*, 2000, Han *et al.*, 2002]. Plusieurs autres exemples peuvent être trouvés dans la thèse de Willy Ugarte [Ugarte, 2014].

Les top-k motifs. Les top-k motifs [Fu *et al.*, 2000, Han *et al.*, 2002] sont un problème classique qui consiste à extraire les k meilleurs motifs par rapport à une mesure d'intérêt $\mathtt{m} : \mathcal{L}_{\mathcal{I}} \to \mathbb{R}$.

4.2 Fouille de motifs séquentiels

Contrairement à la fouille de motifs ensemblistes, la fouille de motifs séquentiels permet de prendre en compte une relation d'ordre entre les données [Agrawal et Srikant, 1995]. Ceci permet une plus grande précision dans les résultats, mais rend la tâche d'extraction plus complexe. Ce point sera davantage détaillé en section 5.4.

4.2.1 Séquences d'items

Soit $\mathcal{I}$ un ensemble fini de n éléments (*items*). Le langage de séquences correspond à $\mathcal{L}_{\mathcal{I}} = \mathcal{I}^n$ où $n \in \mathbb{N}^*$.

Définition 4.10 (séquence, base de séquences)
Une séquence s définie sur $\mathcal{L}_{\mathcal{I}}$ est une liste ordonnée d'éléments $\langle s_1 s_2 \dots s_n \rangle$, où $s_i \in \mathcal{I}$ est un item, $1 \leq i \leq n$. n est appelé la longueur de la séquence s. Une base de séquences SDB est un ensemble de tuples (sid, s), où s est la séquence et sid représente son identifiant.

Le concept clé en fouille de séquences est la relation de sous-séquence. Une séquence s est sous-séquence d'une autre séquence s' s'il existe une correspondance de chaque élément de s avec le même symbole dans la séquence de s' telle que l'ordre soit respecté.

Définition 4.11 (sous-séquence, la relation $\preceq$)
Une séquence $s = \langle s_1 \dots s_m \rangle$ est une sous-séquence d'une séquence $s' = \langle s'_1 \dots s'_n \rangle$ (ou s' est une *sur-séquence* de s), notée $(s \preceq s')$, ssi $m \leq n$ et il existe des entiers $1 \leq j_1 \leq \dots \leq j_m \leq n$ tels que $s_i = s'_{j_i}$ pour tout $1 \leq i \leq m$. Dans ce contexte, le tuple d'entiers $\mathcal{O} = (j_1, \dots, j_m)$ représente une *occurrence* de s dans s'. On dit aussi que s est contenue dans s'. Un tuple (sid', s') contient une séquence s si et seulement si, $s \preceq s'$.

Définition 4.12 (couverture, support)
Soient SDB une base de séquences et p un motif. La couverture de p dans SDB est l'ensemble

sid	Séquence
1	$\langle ABCBC \rangle$
2	$\langle BABC \rangle$
3	$\langle AB \rangle$
4	$\langle BCD \rangle$

TABLE 4.2 – SDB_1 : Un exemple de base de séquences.

de tous les tuples de SDB qui contiennent p : $\texttt{cover}^{27}{}_{SDB}(p)=\{(sid, s) \in SDB \mid p \preceq s\}$, et son support est défini par $\texttt{freq}_{SDB}(p) = |\texttt{cover}_{SDB}(p)|$.

Définition 4.13 (MOTIF SÉQUENTIEL)
Soit un seuil de support minimum $minfr$. On dit qu'un motif p est fréquent dans une base SDB, si son support contient au moins $minfr$ éléments : $\texttt{freq}_{SDB}(p) \geq minfr$. p est appelé motif séquentiel [Agrawal et Srikant, 1995].

Exemple 4.3. La Table 4.2 représente une base composée de 4 séquences, avec $\mathcal{I} = \{A, B, C, D\}$. Soit la séquence $p = \langle AC \rangle$, alors $cover_{SDB1}(p) = \{(1, s_1), (2, s_2)\}$. Si on considère $minfr = 2$, alors $p = \langle AC \rangle$ est un motif séquentiel car $\texttt{freq}_{SDB_1}(p) \geq 2$.

Définition 4.14 (EXTRACTION DE MOTIFS SÉQUENTIELS (EMS))
Soit un seuil de support minimum $minfr \geq 1$ et une base de séquences SDB, le problème d'extraction des motifs séquentiels consiste à trouver tous les motifs p tel que $\texttt{freq}_{SDB}(p) \geq minfr$.

4.2.2 Contraintes

Comme pour les motifs ensemblistes, une démarche complémentaire pour la découverte de motifs utiles, est l'utilisation de contraintes [Dong et Pei, 2007]. Un exemple très classique de contraintes, que nous avons déjà introduite, est celle de fréquence minimale qui permet de restreindre l'ensemble des motifs extraits aux motifs séquentiels selon un seuil. Le paradigme de l'extraction de motifs séquentiels sous contraintes offre de multiples contraintes. Nous identifions ci-dessous trois catégories de contraintes :

(i) **Contraintes sur le motif.** Celles-ci imposent des restrictions sur la structure du motif. Les contraintes de taille, d'expressions régulières ou d'item font partie de cette catégorie.
   — *Contrainte de longueur minimale.* Cette contrainte, notée $\texttt{taille-min}(p, \ell_{min})$, spécifie la taille minimale (en nombre d'items) des motifs extraits (i.e. les motifs extraits doivent contenir au moins ℓ_{min} items).

   — *Contrainte de longueur maximale.* Cette contrainte, notée $\texttt{taille-max}(p, \ell_{max})$, spécifie la taille maximale (en nombre d'items) des motifs extraits (i.e. les motifs extraits doivent contenir au plus ℓ_{max} items).

27. Nous désignerons indifféremment par $\texttt{cover}(p)$, la couverture d'un motif ensembliste ou séquentiel. Toutefois, pour les motifs ensemblistes, on utilisera la notation $\mathcal{T}$, alors que pour les motifs séquentiels, c'est la notation SDB qui sera utilisée.

— *Contrainte d'item.* Cette contrainte spécifie le sous-ensemble d'items qui doivent apparaître dans les motifs extraits : $\texttt{item}(p,t) \equiv (\exists i \in 1..|p|, p_i = t)$.

— *Contrainte d'expression régulière.* Cette contrainte, notée $\texttt{reg}(p, exp)$, assure que le motif p doit être reconnu par un automate d'états finis associé à l'expression régulière exp [Garofalakis *et al.*, 2002].

(ii) **Contraintes sur la couverture.** Un exemple classique de contrainte sur la couverture est la contrainte de fréquence minimale.

— *Contrainte de contraste.* Étant donné deux classes de séquences SDB^+ (positive) et SDB^- (négative), deux seuils $minfr_1$ et $minfr_2$, un motif p est dit *discriminant*, si et seulement si, $\texttt{freq}_{SDB^+}(p) \geq minfr_1$ et $\texttt{freq}_{SDB^-}(p) \leq minfr_2$.

(iii) **Contraintes sur la relation de sous-séquence.** Ces contraintes imposent une restriction sur l'intervalle de temps séparant deux items dans la séquence. Des exemples de telles contraintes sont la durée et gap.

— *Contrainte de gap.* Un motif séquentiel avec contrainte de gap $gap[M,N]$ est un motif tel qu'au minimum M items et au maximum N items sont présents entre chaque item voisin du motif dans les séquences correspondantes. Formellement, une séquence $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ est sous-séquence de $s = \langle s_1 \dots s_n \rangle$, sous la contrainte de gap $gap[M,N]$, ssi $m \leq n$ et il existe des entiers $1 \leq j_1 \leq \dots \leq j_m \leq n$, tels que $\alpha_i = s_{j_i}$ pour tout $1 \leq i \leq m$, et $\forall k \in \{1, ..., m-1\}, M \leq j_{k+1} - j_k - 1 \leq N$.

— *Contrainte de durée.* Cette contrainte est définie uniquement si une date est attribuée à chaque item dans la base de séquences. Un motif séquentiel avec contrainte de durée $Dur[M,N]$ est un motif tel que la distance entre le premier et le dernier item du motif dans les séquences correspondantes doit être comprise entre M et N. Formellement, une séquence $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ est sous-séquence de $s = \langle s_1 \dots s_n \rangle$, sous la contrainte de durée $Dur[M,N]$, ssi $m \leq n$ et il existe des entiers $1 \leq j_1 \leq \dots \leq j_m \leq n$, tels que $\alpha_i = s_{j_i}$ pour tout $1 \leq i \leq m$, et $M \leq date(s_{j_1}) - date(s_{j_m}) \leq N$.

4.2.3 Extraction de motifs séquentiels

Il existe dans la littérature plusieurs méthodes spécialisées pour l'extraction de motifs séquentiels. `GSP` [Srikant et Agrawal, 1996] fut le premier algorithme proposé. Il se base sur l'approche générer et tester, i.e., création de candidats, suivie du test de ces candidats pour confirmer leur fréquence dans la base. Plus tard, deux grandes catégories de méthodes ont été proposées :

1. Les algorithmes de recherche en profondeur basés sur un format vertical pour la représentation de la base de séquences. e.g. `SPADE` [Zaki, 2001] ou `SPAM` [Ayres *et al.*, 2002].
2. Les algorithmes basés sur l'extension de motifs en exploitant le principe de la *projection préfixée* tels que `PrefixSpan` [Pei *et al.*, 2001].

Projection préfixée et bases projetées. Nous donnons ci-dessous les définitions nécessaires pour introduire le principe de la *projection préfixée* [Pei *et al.*, 2001]. Ce principe est au cœur de notre contribution présentée à la section 5.4.3 (cf. aussi l'annexe C).

Définition 4.15 (PRÉFIXE, PROJECTION, SUFFIXE)
Soient $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ et $\beta = \langle \beta_1 \dots \beta_n \rangle$ deux séquences (avec $m \leq n$).

— La séquence α est un préfixe de β ssi $\forall i \in [1..m], \alpha_i = \beta_i$.

— La séquence $\beta = \langle \beta_1 \dots \beta_n \rangle$ est une projection de la séquence s par rapport à α, ssi (1) $\beta \preceq s$, (2) α est un préfixe de β et (3) il n'existe pas de sur-séquence propre β' de β telle que $\beta' \preceq s$ et α est un préfixe de β'.
— La séquence $\gamma = \langle \beta_{m+1} \dots \beta_n \rangle$ est appelée suffixe de s par rapport au préfixe α. Ainsi, nous avons $\beta = concat(\alpha, \gamma)$, avec "$concat$" est l'opérateur standard de concaténation.

Exemple 4.4. Considérons l'exemple 4.3 et soit la séquence $s_1 = \langle ABCBC \rangle$ de SDB_1. La sequence $\alpha = \langle AC \rangle$ est un préfixe de la séquence $\beta = \langle ACBC \rangle$ et $\gamma = \langle BC \rangle$ est son suffixe. La séquence $\beta = \langle ACBC \rangle$ est la projection de s_1 selon α.

Définition 4.16 (Base de séquences projetées)
Soit SDB une base de séquences. La base projetée (ou α-projection) de SDB par rapport à α, notée $SDB|_\alpha$, est l'ensemble de tous les suffixes des séquences de SDB ayant pour préfixe α.

Exemple 4.5. Considérons à nouveau la base de séquences de la Table 4.2, et les préfixes $\langle A \rangle$, $\langle AB \rangle$ et $\langle ABC \rangle$. Nous avons :
— $SDB_1|_{<A>} = \{(1, \langle BCBC \rangle), (2, \langle BC \rangle), (3, \langle B \rangle)\}$.
— $SDB_1|_{<AB>} = \{(1, \langle CBC \rangle), (2, \langle C \rangle), (3, \langle \rangle)\}$.
— $SDB_1|_{<ABC>} = \{(1, \langle BC \rangle), (2, \langle \rangle)\}$.

Pei et al. [Pei *et al.*, 2001] ont proposé un algorithme efficace, appelé `PrefixSpan`, pour l'extraction de motifs séquentiels basé sur le principe de la *projection préfixée*. L'algorithme construit des bases de séquences intermédiaires, qui sont des projections de la base d'origine déduites à partir des préfixes communs relevés. Ensuite, dans chaque base projetée, `PrefixSpan` cherche à faire croître la taille des motifs séquentiels découverts, en appliquant la même méthode de manière récursive.

Exemple 4.6. Prenons l'exemple de la base de séquences de la Table 4.2. Supposons que $minfr = 2$. `PrefixSpan` commence par trouver les items fréquents. Pour cela, une passe sur la base de séquences SDB_1 va permettre de collecter le nombre de séquences supportant chaque item rencontré et donc d'évaluer le support des items de la base. Les items trouvés sont (sous la forme <item> :support) : $\langle A \rangle$: 3, $\langle B \rangle$: 4 et $\langle C \rangle$: 3. Ensuite, les séquences de SDB_1 sont projetées en trois sous-ensembles disjoints, puisqu'il y a trois préfixes de taille 1 (i.e. les trois items fréquents). Ces sous-ensembles seront : (1) les motifs séquentiels ayant pour préfixe $\langle A \rangle$, (2) ceux ayant pour préfixe $\langle B \rangle$ et (3) ceux ayant pour préfixe $\langle C \rangle$. Par exemple, la base projetée de SDB_1 selon le préfixe $\langle A \rangle$, notée $SDB_1|_{\langle A \rangle}$, est composée de 3 suffixes : $\{(1, \langle BCBC \rangle)$, $(2, \langle BC \rangle)$, $(3, \langle B \rangle)\}$. Une passe sur $SDB_1|_{<A>}$ permet alors d'obtenir les motifs séquentiels de longueur 2 ayant $\langle A \rangle$ pour préfixe commun : $\langle AB \rangle$: 3 et $\langle AC \rangle$: 2. Ainsi, de façon récursive, $SDB_1|_{<A>}$ peut être partitionnée en deux sous-ensembles : (1) les motifs séquentiels ayant pour préfixe $\langle AB \rangle$ et (2) ceux ayant pour préfixe $\langle AC \rangle$. Ces motifs peuvent alors former de nouvelles bases projetées, et chacune de ces bases peut être utilisée en entrée de l'algorithme, toujours de manière récursive. Ce processus de α-projection des bases projetées se termine lorsque aucun super-motif séquentiel ne peut être obtenu.

La proposition 4.2 établit une propriété pour calculer le support d'une séquence γ dans $SDB|_\alpha$ [Pei *et al.*, 2001].

Proposition 4.2 (Calcul de support)
Pour toute séquence γ dans SDB ayant pour préfixe α et pour suffixe β, avec $\gamma = concat(\alpha, \beta)$, $sup_{SDB}(\gamma) = sup_{SDB|_\alpha}(\beta)$.

Cette proposition garantit que seules les séquences dans SDB obtenues par extension du motif α seront considérées pour le calcul du support de la séquence γ. En outre, seuls les suffixes de $SDB|_\alpha$ doivent être considérés pour le calcul de ce support.

Chapitre 5

Apports de la PPC pour la fouille de données

L'utilisation de la programmation par contraintes pour exprimer des problèmes de fouille de données possède de nombreux avantages. Le premier est d'offrir à l'utilisateur un moyen simple et déclaratif pour modéliser ses problèmes. Le second est de proposer une approche générique de résolution qui permet à l'utilisateur de ne plus devoir se préoccuper de l'écriture d'un algorithme spécifique pour chaque tâche de fouille. Je présente dans ce chapitre mes principales contributions portant sur les apports de la PPC pour la fouille de données.

Plan du chapitre. La section 5.1 présente la notion de contrainte souple de seuil, introduite dans la thèse de Willy Ugarte, et montre comment de telles contraintes peuvent être mises en œuvre pour l'extraction des top-k motifs souples. La section 5.2 introduit la souplesse dans la relation de dominance et montre l'apport de la programmation par contraintes (PPC) pour offrir un cadre unifié de manipulation de la souplesse dans le problème d'extraction des skypatterns. La section 5.3 propose une nouvelle structure originale, appelée *cube de skypatterns*, qui réunit tous les skypatterns possibles suivant toutes les combinaisons possibles de mesures et présente deux approches pour construire un tel cube. La section 5.4 traite de l'extraction de motifs séquentiels sous contraintes dans un cadre déclaratif et générique de résolution. Les sections 5.5 et 5.6 résument mes contributions dans les autres axes de recherche auxquels je me suis intéressé : aide à la localisation de fautes dans un programme (travaux de thèse de Mehdi Maamar) et le clustering conceptuel sous contraintes en PL-0/1 (travaux de thèse de Abdelkader Ouali). Enfin, nous concluons par une discussion sur nos contributions les plus marquantes.

5.1 Contraintes souples de seuil

(Travail en collaboration avec Willy Ugarte, Patrice Boizumault, Bruno Crémilleux et Alban Lepailleur ; publications associées : JIIS'15 [Ugarte *et al.*, 2015c], DS'12 [Ugarte *et al.*, 2012a], JFPC'12 [Ugarte *et al.*, 2012b]).

Contexte. Une limite importante des méthodes actuelles d'extraction de motifs sous contraintes est la difficulté, pour l'utilisateur, d'exprimer ses préférences. Par ailleurs, le manque de flexibilité et de souplesse du cadre actuel où les contraintes sont usuellement rigides et avec de nombreux paramètres difficiles à régler, fait qu'une solution potentiellement intéressante n'est pas prise en

compte dès qu'une contrainte et/ou sa valeur seuil n'est pas vérifiée. Ce problème est encore plus ardu lorsque plusieurs seuils doivent être fixés simultanément dans une même requête.

Contribution. Pour capturer l'idée de *violation* (ou relaxation) d'une contrainte de seuil, nous avons introduit la notion de *contrainte souple de seuil*. Il s'agit d'une contrainte classique (dure) avec une variable de coût z_i qui quantifie la violation de la contrainte selon une fonction de coût μ (appelée aussi mesure de violation). La mesure de violation permet de mesurer l'écart au seuil. Ainsi, il est possible d'accepter, comme solutions, les motifs qui ne satisfont pas la requête, mais qui sont "proches". Ensuite, nous avons étendu cette notion de souplesse à l'extraction de top-k *motifs souples* (i.e. les k meilleurs motifs qui ne satisfont pas forcément les contraintes sur les mesures fournies par l'utilisateur).

5.1.1 Mesures de violation

Comme pour les contraintes globales relaxées, il existera généralement plusieurs contraintes souples de seuil selon la mesure de violation retenue. Nous avons proposé trois mesures de violation permettant de quantifier la violation sur des contraintes portant sur des mesures. Ces mesures peuvent être quelconques.

Dans ce qui suit, nous noterons par $\mathtt{m}$ une mesure quelconque et par α un seuil minimal ou maximal pour cette mesure. Soit $max_{\mathtt{m}}$ la valeur maximale pour la mesure $\mathtt{m}$[28] et $c(p)$ la contrainte portant sur $\mathtt{m}$, définie soit par $c(p) \equiv \mathtt{m}(p) \geq \alpha$ ou par $c(p) \equiv \mathtt{m}(p) \leq \alpha$.

i) μ_1 : L'écart absolu au seuil. La mesure de violation μ_1 consiste à mesurer, pour chaque motif p, l'écart absolu de $\mathtt{m}(p)$ au seuil :

$$c(p) \equiv \mathtt{m}(p) \geq \alpha \qquad \rightarrow \mu_1(p) = \begin{cases} 0 & si \;\; \mathtt{m}(p) \geq \alpha \\ \alpha - \mathtt{m}(p) & sinon \end{cases}$$

$$c(p) \equiv \mathtt{m}(p) \leq \alpha \qquad \rightarrow \mu_1(p) = \begin{cases} 0 & si \;\; \mathtt{m}(p) \leq \alpha \\ \mathtt{m}(p) - \alpha & sinon \end{cases}$$

ii) μ_2 : L'écart relatif au seuil. La mesure de violation μ_2 est un raffinement de la mesure μ_1, permettant de cumuler les violations de plusieurs contraintes de seuil. Elle permet de mesurer, pour chaque motif p, l'écart relatif de $\mathtt{m}(p)$ au seuil :

$$c(p) \equiv \mathtt{m}(p) \geq \alpha \qquad \rightarrow \mu_2(p) = \begin{cases} 0 & si \;\; \mathtt{m}(p) \geq \alpha \\ \frac{\alpha - \mathtt{m}(p)}{\alpha} & sinon \end{cases}$$

$$c(p) \equiv \mathtt{m}(p) \leq \alpha \qquad \rightarrow \mu_2(p) = \begin{cases} 0 & si \;\; \mathtt{m}(p) \leq \alpha \\ \frac{\mathtt{m}(p) - \alpha}{max_{\mathtt{m}} - \alpha} & sinon \end{cases}$$

iii) μ_3 : L'écart relatif restreint au seuil. L'idée de la mesure de violation μ_3 est de contrôler l'écart au seuil en restreignant la violation de la contrainte $c(p) \equiv \mathtt{m}(p) \geq \alpha$ (resp. $c(p) \equiv \mathtt{m}(p) \leq \alpha$) à l'intervalle $[(1-\epsilon) \times \alpha, \alpha]$ (resp. $[\alpha, (1+\epsilon) \times \alpha]$), avec ϵ ($\epsilon \in [0..1]$) un paramètre qui définit la largeur de l'intervalle. Ainsi, si l'écart relatif de $\mathtt{m}(p)$ au seuil n'est pas compris dans l'intervalle de relaxation, alors on considère que la contrainte est insatisfaite.

28. Pour $\mathtt{m} = \mathtt{freq}$, $max_{\mathtt{m}} = |\mathcal{T}|$; pour $\mathtt{m} = \mathtt{taille}$, $max_{\mathtt{m}} = |\mathcal{I}|$, etc.

$$c(p) \equiv \mathtt{m}(p) \geq \alpha \qquad \rightarrow \mu_3(p) = \begin{cases} 0 & si \;\; \mathtt{m}(p) \geq \alpha \\ \frac{\alpha - \mathtt{m}(p)}{\alpha} & si \;\; (1-\epsilon) \times \alpha \leq \mathtt{m}(p) \leq \alpha \\ \infty & sinon \end{cases}$$

$$c(p) \equiv \mathtt{m}(p) \leq \alpha \qquad \rightarrow \mu_3(p) = \begin{cases} 0 & si \;\; \mathtt{m}(p) \leq \alpha \\ \frac{\mathtt{m}(p) - \alpha}{\alpha} & si \;\; \alpha \leq \mathtt{m}(p) \leq (1+\epsilon) \times \alpha \\ \infty & sinon \end{cases}$$

5.1.2 Mise en œuvre

La mise en œuvre des contraintes souples de seuil dans un solveur CSP repose sur une transformation de ces contraintes en des contraintes dures équivalentes. Cette transformation est obtenue en appliquant la relaxation disjonctive proposée par Thierry Petit [Petit, 2002]. L'exemple ci-dessous montre un exemple de transformation pour $c(p) \equiv (\mathtt{m}(p) \geq \alpha)$ selon les mesures μ_2 et μ_3 :

- la relaxation $c(p)$ selon μ_2 est :

$$c'_i(p) \equiv \left[z = \mathtt{Max}\left(0, \frac{\alpha - \mathtt{m}(p)}{\alpha}\right)\right]$$

- la relaxation $c(p)$ selon μ_3 est :

$$c'(p) \equiv [\mathtt{m}(p) \geq (1-\epsilon) \times \alpha] \wedge \left[z = \mathtt{Max}\left(0, \frac{\alpha - \mathtt{m}(p)}{\alpha}\right)\right]$$

Extraction des motifs souples. Soit une requête souple $\mathtt{q}(p) = \bigwedge_{i=1}^{n} c_i(p)$, et μ_i $(i = 1, n)$ une mesure de violation associée à $c_i(p)$. Pour quantifier la violation de $\mathtt{q}(p)$, il suffit de cumuler les violations individuelles des toutes les contraintes souples $c_i(p)$. Soit la variable de coût $z = \sum z_i$ représentant le cumul des violations, où z_i est la variable de coût associée à chaque contrainte souple c_i. Pour contrôler la violation totale (i.e. cumul des violations des contraintes souples composant la requête $\mathtt{q}(p)$), un seuil de violation maximal autorisé λ ($\lambda \in [0..1]$) est imposé. L'objectif est de trouver tous les motifs souples vérifiant la requête $\mathtt{q}(p)$ dont la violation ne dépasse le seuil λ.

L'extraction de l'ensemble des motifs souples satisfaisant la requête $\mathtt{q}(p)$, peut être effectuée en résolvant la requête dure correspondante où toutes les contraintes souples de seuil sont transformées en contraintes dures équivalentes et en imposant la nouvelle contrainte ($\sum z_i \leq \lambda$) permettant de contrôler la violation totale.

5.1.3 top-k motifs souples

Nous avons également étendu cette notion de souplesse à l'extraction de top-k *motifs souples* (i.e. les k meilleurs motifs qui ne satisfont pas forcément les contraintes sur les mesures fournies par l'utilisateur). Pour cela, nous avons défini une nouvelle mesure d'intérêt sur les motifs extraits de façon à pouvoir les comparer.

Intérêt d'un motif pour une contrainte souple de seuil. Les mesures de violation introduites précédemment ne permettent pas de prendre en compte le degré de satisfaction d'une

Algorithme 8 : Extraction des top-k motifs souples.

Données Φ : CSP, $\mathcal{T}$: la base transactionnelle, $k(\geq 1)$: un entier
Sorties $\mathcal{M}$: top-k motifs souples $\langle p_1, p_2, \ldots, p_k \rangle$
$\mathcal{M} \leftarrow \emptyset \quad i \leftarrow 1 \quad continue \leftarrow vrai \quad$ **Store** $\leftarrow \mathtt{q}'(p)$;
tant que $(i \leq k \wedge p \neq \emptyset)$ **faire**
  si $(p = \mathtt{SolveNext}(\Phi, \mathcal{T}))$ **alors**
    `ajouter` $(p, \mathcal{M})$;
    $i \leftarrow i + 1$;
tant que $(p \neq \emptyset)$ **faire**
  Store $\leftarrow$ **Store** $\cup \{p \rhd_{\theta_\mathtt{q}} p_k\}$;
  $p \leftarrow \mathtt{SolveNext}(\Phi, \mathcal{T})$;
  `retirer`$(p_k, \mathcal{M})$;
  `ajouter` $(p, \mathcal{M})$;
retourner $\mathcal{M}$;

contrainte de seuil $c(p)$. En effet, si $c(p)$ est satisfaite, alors $\mu(p) = 0$. Or un motif p dont la valeur $\mathtt{m}(p)$ est très grande par rapport au seuil sera considéré comme plus intéressant qu'un motif p' dont la valeur $\mathtt{m}(p')$ est légèrement supérieure au seuil. Pour tenir compte du degré de satisfaction ou de violation d'une contrainte de seuil $c_i(p)$, nous avons introduit la mesure d'intérêt $\theta_i : \mathcal{L}_\mathcal{I} \rightarrow [-1..1]$ pour un motif p, notée $\theta_i(p)$:

$$\textit{pour} \qquad c_i(p) \equiv \mathtt{m}(p) \geq \alpha \qquad \theta_i(p) = \begin{cases} \frac{\mathtt{m}(p)-\alpha}{max_\mathtt{m}-\alpha} & si \;\; \mathtt{m}(p) \geq \alpha \\ \\ -\mu(p) & sinon \end{cases}$$

$$\textit{pour} \qquad c_i(p) \equiv \mathtt{m}(p) \leq \alpha \qquad \theta_i(p) = \begin{cases} \frac{\alpha-\mathtt{m}(p)}{\alpha} & si \;\; \mathtt{m}(p) \leq \alpha \\ \\ -\mu(p) & sinon \end{cases}$$

Soit $\theta_\mathtt{q}(p)$ l'intérêt d'un motif p pour une requête $\mathtt{q}(p)$, défini comme étant le cumul des intérêts de p pour les contraintes souples qui la composent. En exploitant une relation de préférence $\rhd_{\theta_\mathtt{q}}$ [29] sur l'ensemble des motifs souples satisfaisant la requête $\mathtt{q}(p)$, nous avons introduit la notion de *top-k motif souple* :

Définition 5.1 (TOP-K MOTIF SOUPLE)
Soit un entier positif k, un motif p est un top-k motif souple selon $\rhd_{\theta_\mathtt{q}}$, si et seulement si, il n'existe pas plus de $(k-1)$ motifs souples satisfaisant la requête $\mathtt{q}(p)$ et dont l'intérêt est plus élevé que celui de p.

Extraction des top-k motifs souples. L'idée principale est d'exploiter la relation de préférence $\rhd_{\theta_\mathtt{q}}$ entre les motifs pour produire un raffinement successif sur les motifs extraits grâce à des contraintes postées dynamiquement au cours de la recherche. Soit $\mathtt{q}(p)$ une requête, et $\mathtt{q}'(p)$ la requête dure équivalente associée. L'algorithme 8 détaille le calcul des top-k motifs souples pour $\mathtt{q}'(p)$ selon $\rhd_{\theta_\mathtt{q}}$. Il prend comme entrées le CSP Φ initial, la base transactionnelle $\mathcal{T}$, un entier positif k et retourne la liste $\mathcal{M}$ des *top-k* motifs souples. On notera par `Store` l'ensemble courant

29. Le motif p est strictement préféré au motif p' selon $\theta_\mathtt{q}$, dénoté par $p \rhd_{\theta_q} p'$, si et seulement si $\theta_\mathtt{q}(p) > \theta_\mathtt{q}(p')$.

des contraintes de Φ (i.e. le store des contraintes). Au départ, `Store` est initialisé à $\mathtt{q}'(p)$ (cf. ligne 3). Le processus d'extraction est réalisé en deux étapes principales :

1. Tout d'abord, les k premiers motifs souples $(p_1, p_2, \ldots, p_k)$ solutions de la requête $\mathtt{q}'(p)$ sont calculés et classés en fonction de $\rhd_{\theta_\mathtt{q}}$ dans la liste $\mathcal{M}$, grâce à la fonction $\mathtt{SolveNext}(\Phi, \mathcal{T})$ (cf. lignes 4-7).
2. Puis, une fois les k premiers motifs souples trouvés, nous imposons que toute nouvelle solution à la requête $\mathtt{q}'(p)$ doit être strictement préférée (au sens de $\rhd_{\theta_\mathtt{q}}$) à la dernière solution p_k en postant dynamiquement la contrainte $(\theta_\mathtt{q}(p) > \theta(p_k))$. Ensuite, dès qu'une nouvelle solution p est obtenue, telle que $\theta_\mathtt{q}(p) > \theta_\mathtt{q}(p_k)$, alors :
   — p est inséré dans la liste $\mathcal{M}$ des top-k solutions et p_k est retiré de $\mathcal{M}$ (cf. lignes 11-12),
   — la nouvelle contrainte est dynamiquement ajoutée au store des contraintes (cf. ligne 9) afin de garantir que le prochain motif recherché doit être meilleur selon la relation $\rhd_{\theta_\mathtt{q}}$.

Résultats obtenus. Notre approche concernant l'extraction des top-k motifs souples a été évaluée à travers une application en chémoinformatique sur la découverte de fragments moléculaires toxicophores. Le but de cette étude, qui s'inscrit dans le cadre d'une plus large collaboration avec le laboratoire CERMN [30], porte sur l'apport de l'utilisation de la souplesse pour la découverte de toxicophores. Les résultats obtenus montrent clairement l'intérêt de la souplesse pour l'obtention de nouvelles connaissances dans ce domaine.

5.2 Skypatterns souples

(Travail en collaboration avec Willy Ugarte, Patrice Boizumault, Bruno Crémilleux et Alban Lepailleur ; publications associées : AIJ [Ugarte *et al.*, 2015a], AKDM5 [Ugarte *et al.*, 2015b], CPAIOR'14 [Ugarte *et al.*, 2014c], EGC'13 [Ugarte *et al.*, 2013]).

Contexte. Bien que la notion de contrainte souple de seuil, introduite en section 5.1, a permis de s'attaquer à *la rigidité du cadre actuel*, il n'en demeure pas moins que le *problème de seuillage* [31] constitue un frein majeur au processus de fouille de données. Par ailleurs, face au problème du "trop de motifs", il n'est pas toujours possible d'examiner manuellement l'ensemble des motifs produits et d'en sélectionner ceux qui sont potentiellement intéressants. L'utilisateur aimerait pouvoir spécifier des préférences entre les motifs afin de découvrir des motifs capturant une forme d'intérêt global sur l'ensemble des mesures. Les requêtes skylines [Börzsönyi *et al.*, 2001], introduites dans les bases de données multidimensionnelles, permettent d'exprimer les préférences de l'utilisateur à partir d'une relation de dominance. Cette notion a été récemment étendue à l'extraction de motifs pour définir la notion de skypattern [Soulet *et al.*, 2011]. Les skypatterns permettent à l'utilisateur d'exprimer des préférences via une relation de Pareto dominance et ne nécessitent donc pas de seuils pour les contraintes portant sur les mesures. Néanmoins, le cadre actuel est rigide et des motifs pouvant en réalité s'avérer intéressants (les motifs dominés proches des skypatterns) ne sont pas proposés. C'est le cas lorsque la requête implique des mesures conflictuelles, et dont le résultat d'interrogation peut être infructueux.

Contribution. Notre originalité consiste en l'introduction de la souplesse dans la relation de dominance ainsi que la proposition d'un cadre unifié basé sur la PPC pour la manipulation de la souplesse dans le problème d'extraction des skypatterns.

30. Centre d'Etudes et de Recherche sur le Médicament de Normandie.
31. Comment fixer les valeurs de seuils liés aux contraintes ?

Trans.	Items					
t_1		B			E	F
t_2		B	C	D		
t_3	A				E	F
t_4	A	B	C	D	E	
t_5		B	C	D	E	
t_6		B	C	D	E	F
t_7	A	B	C	D	E	F

Item	Prix
A	30
B	40
C	10
D	40
E	70
F	55

TABLE 5.1 – Jeu de données transactionnel $\boldsymbol{r}$.

5.2.1 Extraction des Skypatterns

Cette section présente la problématique de l'extraction des skypatterns. Nous commençons par définir la notion de dominance.

Définition 5.2 (DOMINANCE)
Étant donné un ensemble de mesures M, un motif p_1 domine un autre motif p_2 par rapport à M (dénoté par $p_1 \succ_M p_2$) si et seulement si $\forall m \in M, m(p_1) \geq m(p_2)$ et $\exists m' \in M, m'(p_1) > m'(p_2)$.

Exemple 5.1. Considérons l'exemple de la Table 5.1 et supposons que $M = \{\texttt{freq}, \texttt{aire}\}$. Le motif BCD domine le motif BC car $\texttt{freq}(BCD) = \texttt{freq}(BC) = 5$ et $\texttt{aire}(BCD) > \texttt{aire}(BC)$.

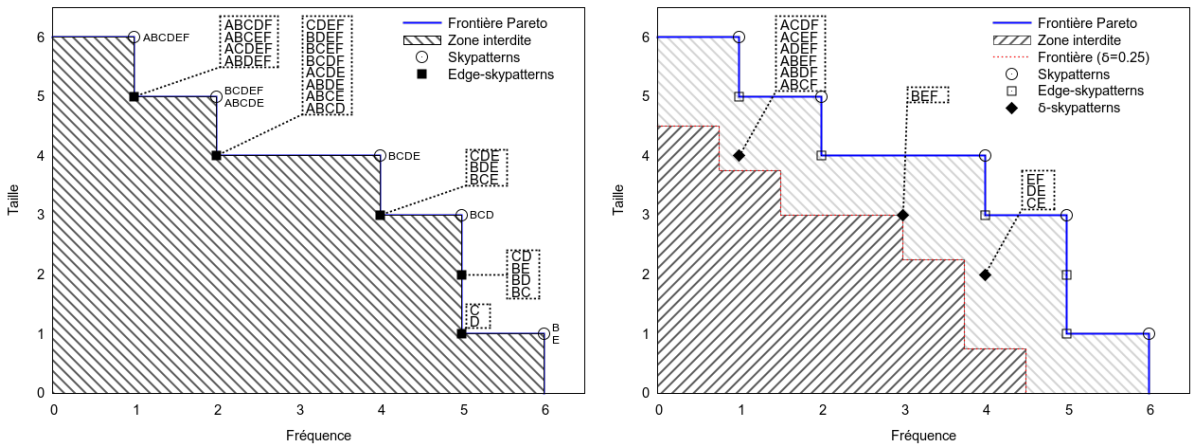
Lorsqu'un motif p_1 domine un autre motif p_2 par rapport à M, cela signifie que p_1 est équivalent ou "meilleur" que p_2 pour toutes les mesures choisies. Ainsi, les motifs dominés par d'autres (au moins un) ne sont pas pertinents et sont éliminés du résultat par l'opérateur Sky.

Définition 5.3 (OPÉRATEUR SKYPATTERN)
Étant donné un ensemble de motifs P et un ensemble de mesures M, un skypattern de P, par rapport à M, est un motif de P non dominé selon M. L'opérateur $Sky(M, P)$ renvoie tous les skypatterns de P selon M : $Sky(M, P) = \{p \in P \mid \nexists\, y \in P, y \succ_M p\}$

Le problème de l'extraction des skypatterns consiste à évaluer la requête $Sky(M, \mathcal{L}_\mathcal{I})$. Pour l'exemple de la Table 5.1, $Sky(M, \mathcal{L}_\mathcal{I}) = \{ABCDEF, BCDEF, ABCDE, BCDE, BCD, B, E\}$, avec $M = \{\texttt{freq}, \texttt{taille}\}$. Ils sont représentés à la figure 5.1. L'aire hachurée correspond à la zone interdite (i.e. zone où il ne peut pas y avoir de skypatterns). La zone de dominance est au-dessus de la ligne bleu, appelée bordure de l'aire de dominance.

Extraction de skypatterns à l'aide des représentations condensées de motifs. L'extraction des skypatterns est un problème difficile en raison du nombre très élevé de motifs candidats (i.e. $|\mathcal{L}_\mathcal{I}|$) et une énumération naïve de $\mathcal{L}_\mathcal{I}$ n'est pas réaliste. Par exemple, avec 1000 items une approche naïve aurait besoin de calculer $(2^{1000} - 1) \times |M|$ valeurs de mesures, et ensuite de comparer les 2^{1000} motifs entre eux.

Dans [Soulet *et al.*, 2011], les auteurs ont montré que tout skypattern selon M peut être obtenu à partir d'une représentation condensée de motifs selon M. Comme l'ensemble représentatifs des skypatterns P est beaucoup plus petit que $\mathcal{L}_\mathcal{I}$, le processus d'extraction devient alors faisable. L'originalité de l'approche proposée dans [Soulet *et al.*, 2011] est d'identifier un petit ensemble de mesures $M' \subseteq M$ (où M' est appelé "ensemble de mesures skylineables" de M) tel que tout



(a) Edge-skypatterns.

(b) δ-skypatterns avec $\delta = 0.25$.

FIGURE 5.1 – Soft-skypatterns extraits de l'exemple de la Table 5.1.

skypattern selon M peut être obtenu à partir de la représentation condensée selon M'. Comme la représentation condensée selon M' comporte (beaucoup) moins de motifs que celle selon M, il s'ensuit un gain de temps pour l'extraction des motifs. Néanmoins, la difficulté ici est d'identifier un ensemble de mesures M' qui soient condensables. En effet, pour des mesures comme l'aire ou la longueur, la représentation condensée est égale à $|\mathcal{L}_\mathcal{I}|$ (i.e. mesure non condensable). Par conséquent, calculer une représentation concise pour chaque mesure $m \in M$ rendrait le processus d'extraction inefficace.

En exploitant ce principe de la skylineabilité, [Soulet *et al.*, 2011] ont proposé AETHERIS. Le processus d'extraction de AETHERIS s'effectue en deux étapes. Tout d'abord, les motifs de la représentation condensée de $\mathcal{L}_\mathcal{I}$ selon M' sont extraits. Ensuite, la production de l'ensemble représentatif des skypatterns s'obtient en appliquant l'opérateur Sky selon M sur les motifs de la représentation condensée.

5.2.2 Extraction de Skypatterns souples

Comme pour les contraintes portant sur des mesures avec seuils, nous avons introduit la souplesse dans le processus d'extraction de skypatterns afin de découvrir des motifs intéressants capturant un effet global sur l'ensemble des mesures mais ne vérifiant pas de façon stricte la relation de dominance. À cette effet, nous avons défini deux types de skypatterns souples : les *edge-skypatterns* et les *δ-skypatterns*.

Edge-skypatterns. Une première façon de relâcher la dominance est de considérer les points qui se trouvent sur la bordure de l'aire de dominance (ligne en blue à la figure 5.1). Ils sont définis à partir de la notion de dominance stricte.

Définition 5.4 (DOMINANCE STRICTE)
Étant donné un ensemble de mesures M, un motif p_1 domine strictement un autre motif p_2 par rapport à M (dénoté par $p_1 \gg_M p_2$) si et seulement si $\forall m \in M$, $m(p_1) > m(p_2)$.

Définition 5.5 (Opérateur Edge-skypattern)
Étant donné un ensemble de motifs P et un ensemble de mesures M, un edge-skypattern de P, par rapport à M, est un motif de P non strictement dominé par rapport à M. L'opérateur $Edge\text{-}Sky(M, P)$ renvoie tous les edge-skypatterns de P par rapport à M : $Edge\text{-}Sky(M, P) = \{p \in P \mid \nexists\, y \in P, y \gg_M p\}$

Le problème de l'extraction des edges-skypatterns consiste à évaluer la requête $Edge\text{-}Sky(M, \mathcal{L}_\mathcal{I})$. La figure 5.1a illustre les $28 = 7 + (4 + 8 + 3 + 4 + 2)$ edge-skypatterns extraits de l'exemple de la Table 5.1. Notons que $Sky(M, P) \subseteq Edge\text{-}Sky(M, P)$.

δ-skypatterns. Une seconde façon de relâcher la dominance est de considérer les points de la zone interdite (i.e. zone hachurée) qui sont proches de la bordure de l'aire de dominance, au sens d'une mesure de distance δ, avec $0 < \delta \leq 1$. La notion de δ-dominance permet de capturer ce type de motifs.

Définition 5.6 (δ-Dominance)
Étant donné un ensemble de mesures M, un motif p_1 δ-domine un autre motif p_2 par rapport à M (dénoté $p_1 \succ^\delta_M p_2$), si et seulement si $\forall m \in M, (1 - \delta) \times m(p_1) > m(p_2)$.

Définition 5.7 (Opérateur δ-skypattern)
Étant donné un ensemble de motifs P et un ensemble de mesures M, un δ-skypattern de P par rapport à M est un motif de P non δ-dominé par rapport à M. L'opérateur $\delta\text{-}Sky(M, P)$ détermine tous les δ-skypatterns par rapport à M : $\delta\text{-}Sky(M, P) = \{p \in P \mid \nexists\, y \in P, y \succ^\delta_M p\}$

Le problème de l'extraction des δ-skypatterns consiste à évaluer la requête $\delta\text{-}Sky(M, \mathcal{L}_\mathcal{I})$. En appliquant l'opérateur $\delta\text{-}Sky(M, P)$ sur notre exemple, avec $\delta = 0.25$, nous obtenons 10=6+3+1 nouveaux motifs (cf. figure 5.1b). Notons que $Edge\text{-}Sky(M, P) \subseteq \delta\text{-}Sky(M, P)$.

5.2.3 Mise en œuvre de l'extraction à l'aide des CSP dynamiques

Contrairement à Aetheris, nous proposons d'appliquer localement l'opérateur Sky durant l'extraction. En effet, dès qu'un motif est un candidat skypattern, l'espace de recherche dominé par ce motif peut être éliminé. Pour cela, on définit un CSP dynamique (DCSP) où, chaque fois qu'une solution est trouvée, une nouvelle contrainte conduisant à réduire l'espace de recherche est postée dynamiquement. Comme pour Aetheris, nous générons uniquement les motifs de la représentation condensée par rapport à M' où M' est l'ensemble des mesures skylineables de M.

Extraction des skypatterns. Soit M un ensemble de mesures, et p la variable représentant le motif recherché. Considérons le DCSP [32] $\Phi_1, \Phi_2, \ldots, \Phi_n$ où chaque $\Phi_i = (\{p\}, \mathcal{L}_\mathcal{I}, \mathtt{q}_i(p))$ avec :
— $\mathtt{q}_1(p) = \mathtt{fermé}_{M'}(p)$
— $\mathtt{q}_{i+1}(p) = \mathtt{q}_i(p) \wedge \phi(s_i, p)$ où s_i est la solution à la requête $\mathtt{q}_i(p)$.

La contrainte $\mathtt{fermé}_{M'}(p)$ impose que p doit être un motif fermé par rapport à M' (cf. principe de la skylineabilité). Pour contraindre que la prochaine solution à la requête $\mathtt{q}_{i+1}(p)$ ne doit pas être dominée par s_i (i.e. solution de la requête $\mathtt{q}_i(p)$), une nouvelle contrainte $\phi(s_i, p)$ est postée dynamiquement. Ce processus s'arrête pour la valeur n telle que la requête $\mathtt{q}_{n+1}(p)$ n'a pas de solution. Soit $Cand$ l'ensemble des solutions successives obtenues en résolvant le DCSP $\Phi_1, \Phi_2, \ldots, \Phi_n$. $Cand$ est alors filtré afin de ne conserver que les motifs qui sont des skypatterns.

32. Les modifications des CSP sont effectuées uniquement par l'ajout de nouvelles contraintes.

Extraction des skypatterns souples. Comme pour les skypatterns, le processus d'extraction des skypatterns souples est réalisé en deux étapes principales :

1. Extraction de l'ensemble *Cand* des motifs candidats à l'aide des CSP dynamiques ;
2. Filtrage de tous les candidats qui ne seraient pas des skypatterns souples.

Le seul changement concerne la contrainte $\phi(s_i, p)$ postée dynamiquement. Celle-ci dépend du type de skypatterns souples et de sa relation de dominance [Ugarte *et al.*, 2014c].

Résultats obtenus. Une étude expérimentale approfondie menée sur plusieurs jeux de donnés de l'*UCI*[33] ainsi que sur une étude de cas en chémoinformatique ont montré l'efficacité de AETHERIS et de notre approche (dénommée CP+SKY) aussi bien pour l'extraction de skypatterns durs que souples. Malgré le caractère générique apporté par le cadre CSP et le fait que AETHERIS tire bénéfice des stratégies d'élagage basées sur l'anti-monotonie pour extraire des motifs, CP+SKY demeure très compétitive face à AETHERIS et même le surpasse dans certains cas.

5.3 Cube de Skypatterns

(Travail en collaboration avec Willy UGARTE, Patrice BOIZUMAULT et Bruno CRÉMILLEUX ; publications associées : ECAI'14 [Ugarte *et al.*, 2014b] et ICATI'14 [Ugarte *et al.*, 2014a] ; article repris en annexe B).

Dans la pratique, les utilisateurs ne connaissent pas le rôle exact de chaque mesure et préfèrent rechercher des motifs dominants selon différentes combinaisons de mesures. Sur le même modèle que les cubes de skylines dans les bases de données, l'utilisateur aimerait pouvoir explorer et affiner le rôle de chaque mesure afin d'observer le comportement des skypatterns à travers l'espace multidimensionnel et ainsi analyser et comprendre les différents facteurs de dominance. Pour répondre à cette problématique, nous avons introduit une nouvelle structure originale, appelée *cube de skypatterns*, qui réunit tous les skypatterns possibles suivant toutes les combinaisons possibles de mesures.

Définition 5.8 (CUBE DE SKYPATTERNS)
Soit M un ensemble de mesures, le cube de skypatterns sur M est l'ensemble de tous les skypatterns dans tous les sous-ensembles non-vides M_u de M :

$$\textit{Skypattern-Cube}(M) = \{(M_u, \textit{Sky}(M_u, \mathcal{L}_\mathcal{I})) \mid \ M_u \subseteq M, M_u \neq \emptyset\}$$

La structure du cube de skypatterns peut être représentée par un treillis. Chaque nœud du treillis est appelé cuboïde. Les différents cuboïdes sont regroupés par niveau en fonction de leur nombre de mesures. Ces niveaux sont numérotés en partant du bas du treillis (cuboïdes portant sur une seule mesure) et en remontant vers le sommet (cuboïde suivant toutes les mesures possibles).

Construction du cube de skypatterns. Une approche base-line pour calculer le cube de skypatterns serait d'appliquer la méthode AETHERIS ou CP+SKY aux $2^{|M|} - 1$ sous-ensembles non-vides de M. Toutefois, dans la pratique, une telle approche devient vite inefficace quand le

33. http://www.ics.uci.edu/~mlearn/MLRepository.html.

nombre de cuboïdes devient très grand. Pour calculer de manière efficace le cube de skypatterns, nous avons proposé deux approches différentes :

1. **une approche ascendante** [Ugarte *et al.*, 2014b], qui permet de construire le cube niveau par niveau. Les ensembles de skypatterns d'un nœud père du niveau i du treillis sont obtenus en appliquant des règles de dérivation aux ensembles de skypatterns de ses nœuds fils obtenus au niveau précédent $(i-1)$, ceci sans effectuer aucun test de dominance. Les skypatterns manquants d'un nœud père (i.e. ceux non-dérivables à partir de ses nœuds fils) sont calculés à la volée à l'aide des CSP dynamiques. Une condition suffisante permet d'éviter de lancer le calcul des non-dérivables alors que l'on peut déterminer a priori qu'il n'en existe aucun. L'annexe B présente de manière détaillée cette approche.
2. **une approche par relaxation** [Ugarte *et al.*, 2014a] fondée sur le fait que chaque cuboïde peut être approximé par un ensemble unique, $\mathit{Edge\text{-}Skypattern}(M)$, qui est l'ensemble des edge-skypatterns par rapport à M. Le problème de calcul du cube de skypatterns est alors transformé en un problème (équivalent) de calcul d'un cube de skylines en $|M|$ dimensions.

À notre connaissance, les deux approches proposées sont les seules méthodes non-naïve et efficace pour la construction d'un cube de skypatterns.

Résultats obtenus. Les expérimentations menées sur un jeu de données réel de grande taille, extrait à partir des données de mutagénicité [Hansen *et al.*, 2009] (un problème majeur dans l'évaluation des risques des substances chimiques) fourni par le CERMN, montrent que les deux approches ascendante et par relaxation nécessitent respectivement 13 heures et 42 minutes pour calculer le cube avec un nombre de mesures $|M| = 11$, alors que l'approche base-line a besoin de plus de 400 heures de calcul. Une analyse qualitative des résultats obtenus montre que nos règles de dérivation (pour l'approche ascendante) permettent de déduire environ $80-90\%$ des skypatterns. Par ailleurs, la qualité de notre approximation du cube de skypatterns par $\mathit{Edge\text{-}Skypattern}(M)$ reste meilleure comparée à une approximation par les motifs fermés.

5.4 Approches PPC pour la fouille de motifs séquentiels

L'utilisation de la PPC pour la fouille de motifs séquentiels offre un cadre déclaratif et générique de résolution qui permet à l'utilisateur de ne plus devoir se préoccuper de l'écriture d'algorithmes spécifiques pour la prise en compte de nouvelles contraintes. Toutefois, peu de travaux ont été effectués dans cette direction en raison de l'ordre d'apparition des items au sein du motif qui rend la tâche plus complexe que l'extraction d'itemsets (ou motifs ensemblistes).

Je présente ci-dessous une synthèse de nos contributions pour l'extraction de motifs séquentiels sous contraintes – à partir d'une base de séquences – dans un cadre déclaratif permettant de traiter simultanément des contraintes de nature quelconque.

5.4.1 Un premier modèle CSP pour l'extraction de motifs séquentiels

(Travail en collaboration avec Jean-Philippe Métivier et Thierry Charnois ; publications associées : Atelier LML'13 [Métivier *et al.*, 2013], EGC'14 [Metivier *et al.*, 2014]).

Dans [Métivier *et al.*, 2013], nous avons proposé un premier modèle CSP pour l'extraction de motifs séquentiels. Pour chaque séquence s de SDB, une contrainte réifiée est définie indiquant si le motif à extraire p est sous-séquence (ou non) de s : $(S_s = 1) \Leftrightarrow (p \preceq s)$.

Pour modéliser la relation de sous-séquence ($p \preceq s$), chaque séquence est encodée par un automate d'état fini qui capture toutes les sous-séquences de la séquence en question. Si le motif séquentiel peut être reconnu par l'automate associé à une séquence, alors le motif couvre la séquence. Nous avons proposé de modéliser cette contrainte de couverture à l'aide d'une contrainte `Regular`. Nous avons montré que de nombreuses contraintes usuelles de la fouille de motifs séquentiels peuvent être simplement modélisées (la contrainte gap à l'aide de `Regular`, la contrainte sur la longueur du motif séquentiel ...).

5.4.2 Extraction de motifs séquentiels avec wildcards

(Travail en collaboration avec Amina Kemmar, Willy Ugarte, Yahia Lebbah, Patrice Boizumault, Thierry Charnois et Bruno Crémilleux ; publication : ICTAI'14 [Kemmar *et al.*, 2014]).

En nous appuyons sur la notion de motif séquentiel avec wildcards, nous avons proposé dans [Kemmar *et al.*, 2014] un premier modèle CSP pour l'extraction de motifs séquentiels avec wildcards explicites dans une base de séquences. Un wildcard est un symbole spécial qui peut remplacer n'importe quel item (attribut) dans une séquence. Dans ce modèle, les motifs séquentiels avec items non contigus sont modélisés en utilisant les wildcards comme des jokers. Si l'on considère l'exemple de la Table 4.2, les deux motifs $\langle A \diamond C \rangle$ et $\langle A \diamond \diamond C \rangle$ sont considérés comme différents. Nous avons montré comment modéliser de nombreuses contraintes définies sur les motifs à extraire. Ces contraintes portent sur deux catégories de motifs : des contraintes définies sur des motifs locaux (e.g. fréquence, longueur, expression régulière, gap), ou des contraintes définies sur des ensembles de motifs comme les top-k motifs ou encore les sous-groupes pertinents.

Bilan. Toutes ces propositions utilisent des **contraintes réifiées** pour encoder la base de séquences. Pour chaque séquence s de SDB, une contrainte réifiée est définie indiquant si le motif à extraire p est une sous-séquence (ou non) de s : $(S_s = 1) \Leftrightarrow (p \preceq s)$. Cet encodage permet d'exprimer simplement la contrainte de fréquence : $\texttt{freq}_{SDB}(p) = \sum_{s \in SDB} S_s$. Mais, il présente un inconvénient majeur car il nécessite $m = \#SDB$ contraintes réifiées pour encoder toute la base de séquences. Ceci constitue une limitation forte sur la taille des bases qui peuvent être traitées.

La plupart de ces propositions encodent la **relation de sous-séquence** ($p \preceq s$) en utilisant un ensemble de variables $Pos_{s,j}$ ($s \in SDB$ et $1 \leq j \leq \ell$) pour déterminer l'occurrence $\mathcal{O}$ de p dans s (cf. définition 4.11). Lorsque seules quelques occurrences sont possibles, comme dans cas de séquences avec wildcards explicites, l'encodage peut être réalisé avec des contraintes disjonctives sur toutes les occurrences possibles (voir [Kemmar *et al.*, 2014] pour plus de détails). Mais pour les séquences standards, le nombre de ces occurrences devient exponentiel, interdisant ainsi tout encodage direct et rendant l'algorithme d'énumération très coûteux en termes de temps de calcul. En outre, il nécessite un grand nombre de variables supplémentaires ($m \times \ell$).

Afin de remédier à cet inconvénient, Negrevergne *et al.* [Négrevergne et Guns, 2015] ont proposé une contrainte globale `exists-embedding` pour encoder la relation de sous-séquence. Ils ont exploité le principe de la *fréquence projetée* pour garder seulement les items fréquents qui apparaissent après le préfixe courant, grâce à l'introduction de variables supplémentaires (une par séquence). Une procédure de recherche spécifique (basée sur les domaines des variables supplémentaires) est utilisée pour éviter d'énumérer les items non fréquents. Mais encore une fois, cet encodage repose sur des contraintes réifiées et requiert m contraintes globales `exists-embedding`.

5.4.3 Une contrainte globale pour l'extraction de motifs séquentiels

(Travail en collaboration avec Amina Kemmar, Yahia Lebbah, Patrice Boizumault et Thierry Charnois ; publication associée : CP'15 [Kemmar *et al.*, 2015] ; article repris en annexe C).

Comme nous l'avons souligné précédemment, l'utilisation de la PPC pour exprimer des problèmes de fouille de données offre un cadre déclaratif et générique de résolution. Toutefois, la contrepartie à cette grande généricité est la relative faiblesse des approches déclaratives existantes qui ne passent pas à l'échelle comparées aux méthodes spécialisées. En effet, comme nous l'avons indiqué, l'encodage PPC, qui utilise une contrainte réifiée par séquence, constitue une limitation forte sur la taille des bases qui peuvent être traitées.

Pour pallier à tous ces inconvénients, nous avons proposé dans [Kemmar *et al.*, 2015] une nouvelle contrainte globale Prefix-Projection qui exploite le principe de la projection préfixée (cf. section 4.2.3) afin d'encoder dans une seule contrainte la relation de sous-séquence et la contrainte de fréquence. Notre encodage permet de mettre en œuvre différents types de contraintes (appartenance d'items, taille et expressions régulières) et de les combiner simultanément. Par rapport aux approches PPC existantes, Prefix-Projection ne requiert ni contraintes réifiées ni variables supplémentaires pour encoder la relation de sous-séquence. Les expérimentations menées montrent que notre approche surpasse clairement les approches PPC existantes et concurrence les méthodes spécialisées sur de grandes bases de séquences. Une présentation détaillée de la contrainte globale Prefix-Projection est donnée en annexe C.

5.4.4 Une contrainte globale pour l'EMS avec GAP

(Travail en collaboration avec Amina Kemmar, Yahia Lebbah, Patrice Boizumault et Thierry Charnois ; publication associée : CPAIOR'16 [Kemmar *et al.*, 2016] ; article repris en annexe D).

Une autre contrainte couramment employée en fouille de motifs séquentiels est la contrainte de gap. Cette contrainte permet de contraindre l'intervalle de temps entre deux items consécutifs d'un motif dans les séquences d'origines. L'extraction de motifs séquentiels sous la contrainte de gap est une tâche difficile car la propriété d'anti-monotonie n'est plus respectée.

Bien que Prefix-Projection est bien adaptée pour les contraintes sur le motif, celle-ci ne permet de prendre en compte les contraintes sur la relation de sous-séquence, comme la durée ou le gap (cf section 4.2.2). Pour y remédier, nous avons récemment proposé dans [Kemmar et al., 2016] une nouvelle extension de la contrainte globale Prefix-Projection, dénommée GAP-SEQ, permettant d'extraire les motifs séquentiels avec ou sans la contrainte de gap. Sans détailler (cf. annexe D), GAP-SEQ utilise un encodage concis et son filtrage exploite la propriété de *préfixe-anti-monotonie*[34]. L'idée générale est de calculer toutes les extensions valides à droite du motif, i.e., celles respectant la contrainte de gap, puis de générer de nouveaux candidats en étendant ce motif d'un item supplémentaire à droite. Cet item est sélectionné parmi ceux fréquents dans les extensions valides à droite du motif.

Les expérimentations menées sur différents jeux de données réels de grande taille montrent que GAP-SEQ surpasse clairement les approches PPC et la méthode de l'état de l'art `cSpade` pour l'extraction de motifs séquentiels sous la contrainte de gap. L'annexe D détaille cette nouvelle contrainte globale ainsi que son algorithme de filtrage.

34. Une contrainte c est dite préfixe-anti-monotone si pour chaque motif p satisfaisant c, tous les préfixes de p satisfont c également.

5.5 Aide à la localisation de fautes dans les programmes

(Travail en collaboration avec Mehdi Maamar, Nadjib Lazaar et Yahia Lebbah ; publications associées : ASE'15 [Maamar *et al.*, 2015], JFPC'16 [Maamar *et al.*, 2016]).

Contexte. Je me suis intéressé au problème d'aide à la localisation de fautes à partir de traces d'exécution. Cette question reste cruciale dans la mise au point de logiciels critiques. En effet, la trace d'exécution fournie par un model-checker est souvent longue et difficile à comprendre, et de ce fait d'un intérêt très limité pour le programmeur qui doit débugger son programme. La localisation des portions de code qui contiennent des fautes est donc souvent un processus difficile et coûteux, même pour des programmeurs expérimentés. Au cours de la dernière décennie, plusieurs techniques automatisées ont été proposées pour résoudre ce problème. La plupart de ces techniques automatisées pour la localisation de fautes comparent deux types de traces d'exécution, les exécutions correctes (dites positives) et les exécutions erronées (dites négatives). Ces méthodes se basent sur une fonction de score pour évaluer le caractère suspect de chaque instruction dans le programme en exploitant les occurrences des instructions apparaissant dans les traces négatives et positives.

Contribution. Nous avons étudié les apports des méthodes de fouille pour l'aide à la localisation d'une faute dans les programmes à partir des traces d'exécution. Dans ce cadre, nous avons proposé dans [Maamar *et al.*, 2015] une première formalisation du problème comme un problème de classification supervisée (exécutions correctes versus exécutions erronées) en fouille de données et utilisé nos travaux sur l'extraction de top-k à l'aide des CSP dynamiques présentées en section 5.1.3 pour identifier les top-k suites d'instructions les plus suspectes.

5.5.1 Modélisation

Soit $L = \{e_1, ..., e_n\}$ l'ensemble des instructions composant le programme P et $T = \{tc_1, ..., tc_m\}$ l'ensemble des cas de test destinés à tester le programme P. Le problème de fouille associé est formalisé comme suit : (i) chaque instruction de L correspond à un item, (ii) la couverture [35] de chaque cas de test tc_i forme une transaction dans l'ensemble des transactions $\mathcal{T}$. Par ailleurs, l'ensemble $\mathcal{T}$ est partitionné en deux sous-ensembles disjoints $\mathcal{T}^+$ et $\mathcal{T}^-$, où $\mathcal{T}^+$ (resp. $\mathcal{T}^-$) représente l'ensemble des couvertures des cas de test positifs (resp. négatifs).

En s'appuyant sur cette formalisation, la tâche de fouille consiste en l'extraction des k meilleurs motifs (top-k) satisfaisant un ensemble de contraintes modélisant les instructions les plus suspectes. Soit p le motif suspect recherché, comme dans [De Raedt *et al.*, 2008] (cf. section 4.1.4), nous introduisons deux ensembles de variables booléennes : n variables $\{X_1, ..., X_n\}$ pour représenter le motif et m variables $\{T_1, ..., T_m\}$ pour représenter son support. En outre, pour réduire la redondance dans les suites d'instructions extraites, nous imposons la contrainte de fermeture `fermé`(p).

5.5.2 top-k motifs suspects

L'intuition qui est derrière la grande majorité des méthodes de localisation de fautes est que les instructions qui apparaissent très souvent dans les traces d'exécution négatives sont considérées comme étant les plus suspectes, tandis que les instructions qui apparaissent seulement dans

35. L'ensemble des instructions de P exécutées (au moins une fois) par tc_i désigne sa couverture.

les traces d'exécutions positives sont considérées comme les plus innocentes [Jones et Harrold, 2005, Nessa *et al.*, 2008].

Pour extraire les motifs les plus suspects, nous avons introduit une relation de préférence entre les motifs, notée $\rhd_{\mathcal{S}}$. Cette relation de préférence exprime le fait que le motif p sera préféré à p', ssi p est plus fréquent que p' dans $\mathcal{T}^-$. Si les deux motifs couvrent exactement le même ensemble de transactions dans $\mathcal{T}^-$, alors le motif le moins fréquent dans $\mathcal{T}^+$ sera préféré. Sur le même principe que celui des top-k motifs souples, nous avons proposé la notion de *top-k motif suspect*.

Définition 5.9 (TOP-K MOTIF SUSPECT)
Étant donné une préférence $\rhd_{\mathcal{S}}$ et un entier positif k, un motif p est un top-k motif suspect selon $\rhd_{\mathcal{S}}$, si et seulement si, il n'existe pas plus de $(k-1)$ motifs suspects de $\mathcal{L}_{\mathcal{I}}$ qui sont préférés à p :

$$\#\{y \in \mathcal{L}_{\mathcal{I}} \mid \ y \neq p \wedge y \rhd_{\mathcal{S}} p\} < k$$

En plus d'être fermé, nous avons imposé que le motif p doit inclure au moins une instruction et apparaître au moins une fois dans $\mathcal{T}^-$.

5.5.3 Aide à la localisation d'une faute

L'approche proposée comporte deux étapes :

1. Extraction des top-k suites d'instructions les plus suspectes,
2. Classement des instructions issues des top-k motifs pour localiser la faute.

Pour extraire les top-k motifs suspects, nous avons fait appel aux contraintes postées dynamiquement au cours de la recherche (cf. section 5.1.3). A l'issue de cette première étape, on obtient une liste ordonnée des k meilleurs motifs $\mathcal{S} = \langle p_1, \ldots, p_k \rangle$. Chaque motif p_i représente un sous-ensemble d'instructions pouvant expliquer et localiser la faute. Enfin un post-traitement des top-k motifs suspects permet de retourner une liste contenant un classement précis des instructions susceptibles de localiser la faute.

Les expérimentations menées sur une série de programmes Siemens montrent que notre approche permet de proposer une localisation plus précise comparée à la technique de localisation la plus populaire proposée par Tarantula [Jones et Harrold, 2005].

5.6 Clustering conceptuel sous contraintes en PL-0/1

(Travail en collaboration avec Abdelkader OUALI, Yahia LEBBAH, Patrice BOIZUMAULT, Lakhdar LOUKIL et Albrecht ZIMMERMANN ; publications associées : IJCAI'16 [Ouali *et al.*, 2016b], JFPC'16 [Ouali *et al.*, 2016a]).

Contexte. Le clustering est une activité importante en fouille de données dont l'objectif est de partitionner un ensemble de transactions en groupes (clusters) relativement homogènes. Le **clustering conceptuel** consiste à fournir une description distincte de chaque cluster (groupe), c.-à-d. le concept caractérisant l'ensemble des transactions qu'il contient. Ce problème peut être

formulé comme : rechercher un ensemble de k clusters, où chacun est décrit par un motif fermé $c_1, c_2, ..., c_k$, et couvrant toutes les transactions sans aucun chevauchement entre les clusters.

Une fonction d'évaluation f est nécessaire pour évaluer la qualité du clustering. Ainsi, le clustering conceptuel revient à trouver un ensemble disjoint de clusters sur $\mathcal{T}$ qui optimise un certain critère donné sur la qualité du clustering.

Deux types de méthodes ont été proposées pour le clustering conceptuel : (1) les approches déclaratives qui reposent sur des solveurs génériques [Dao *et al.*, 2015, Metivier *et al.*, 2012, Guns *et al.*, 2013], et (2) les heuristiques [Perkowitz et Etzioni, 1999, Pensa *et al.*, 2005] qui fournissent des solutions de bonne qualité dans des temps de calcul jugés raisonnables. Toutefois, les méthodes actuelles proposées dans la littérature restent loin d'être des méthodes utilisées au niveau applicatif. En effet, les approches déclaratives souffrent d'un problème de passage à l'échelle pour des bases de données de grande taille, alors que les méthodes heuristiques sont instables quant à la qualité des solutions fournies car fortement influencées par les conditions d'initialisation, et nécessitent généralement de nombreuses exécutions.

Contribution. Nous avons proposé une approche hybride combinant à la fois la PLNE (Programmation Linéaire en Nombres Entiers) et la fouille de concepts. Notre approche est complète et tire profit du cadre général de la PLNE (en procurant un haut niveau de flexibilité et d'expressivité) et des heuristiques spécialisées pour l'exploration et l'extraction de motifs.

5.6.1 Modélisation en PL-0/1

L'approche que nous proposons pour résoudre le problème de clustering conceptuel sous contraintes exploite à la fois l'efficacité des extracteurs de motifs utilisés par la communauté de la fouille de données tels que LCM [Uno *et al.*, 2004] et le cadre déclaratif de la programmation linéaire (PL). Notre approche procède en deux étapes :

1. Extraction des motifs sous contraintes locales par une méthode de fouille.
2. Modélisation du problème de clustering sous contraintes globales en termes de programme linéaire en nombres entiers et sa résolution par une méthode complète.

maximize or minimize	(1) $\quad z = \sum_{c \in \mathcal{C}} v_c x_c$
subject to	(2) $\quad \sum_{c \in \mathcal{C}} a_{t,c} x_c = 1 \quad \forall t \in \mathcal{T}$
	(3) $\quad k_{min} \leq \sum_{c \in \mathcal{C}} x_c \leq k_{max}$
	$x_c \in \{0,1\}, \quad c \in \mathcal{C}$

FIGURE 5.2 – Modèle PL-0/1 pour le clustering conceptuel .

Soient $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ un ensemble de m transactions définies sur un ensemble $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ de n items. Et soit $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ l'ensemble des clusters (motifs fermés sous contraintes locales) fournis par un extracteur de fouille. La figure 5.2 donne le modèle PL-0/1 proposé pour le problème de clustering conceptuel. On associe à un cluster représenté par un

motif clos $c \in \mathcal{C}$ une variable binaire x_c. ($x_c = 1$) ssi le motif clos c est choisi dans le clustering. $a_{t,c}$ est une matrice binaire $m \times p$ où ($a_{t,c} = 1$) ssi la transaction t appartient au cluster représenté par le motif clos c. La fonction objective (1) est définie en associant à chaque cluster c une valeur v_c exprimant l'intérêt (à maximiser) ou le coût (à minimiser) du cluster c. La contrainte (2) impose que toute transaction t doit être couverte par un et un seul cluster c. La contrainte (3) permet à l'utilisateur de contrôler le nombre k de clusters en spécifiant une borne inférieure k_{min} et/ou une borne supérieure k_{max}. D'autres contraintes, permettant de modéliser d'autres variantes de clustering, comme le soft clustering, le co-clustering, et le soft co-clustering, sont également proposées dans [Ouali *et al.*, 2016b]. Par ailleurs, comme le montre la contrainte (3), la relaxation de k octroie à notre approche une plus grande flexibilité, ce qui n'est pas le cas pour les approches déclaratives actuelles, où la valeur de k doit être fixée au départ.

Résultats obtenus. Nous avons mené plusieurs expérimentations permettant, (1) de comparer (en termes de temps CPU) notre approche avec celle proposée dans [Guns *et al.*, 2013], et (2) d'évaluer la qualité des clusters résultants et leurs descriptions obtenues par notre approche avec celles qui résultent des approches heuristiques existantes [Perkowitz et Etzioni, 1999, Pensa *et al.*, 2005]. Les expérimentations montrent, d'une part, la flexibilité de notre modèle qui permet d'attaquer une grande variété de problèmes de clustering de motifs sous contraintes, et d'autre part, le passage à l'échelle du clustering quand le nombre de clusters exigé augmente. Par ailleurs, les expérimentations montrent comment il peut être difficile de contrôler les résultats des heuristiques : les clusterings retournés par les approches heuristiques ne couvrent jamais toutes les transactions du jeu de données, et ont souvent des clusters qui se chevauchent, contrairement à notre approche, qui offre plus de garantie sur les solutions fournies. Afin d'avoir une comparaison équitable avec les approches heuristiques, nous avons exécuté notre approche avec des configurations permettant la même quantité de non-couverture : d'un point de vue qualitatif, les clusterings trouvés par notre approche sont proches des clusterings obtenus par les heuristiques.

5.7 Conclusion

Dans ce chapitre, j'ai présenté plusieurs résultats montrant les apports de la programmation par contraintes pour la fouille de données.

Le but dans cet axe de mes recherche était double. D'une part, répondre à deux verrous importants en fouille de données : *l'introduction de souplesse dans le processus d'extraction de motifs et l'utilisation de préférences par l'utilisateur*. Ces deux verrous se rejoignent dans l'idée générale de réduire l'impact du paramétrage des méthodes de fouille de données. L'introduction du cube des skypatterns, qui permet d'étudier les différents ensembles de skypatterns suivant différentes combinaisons possibles de mesures, et sa construction fondée sur une approche ascendante, constitue de mon point de vue une contribution originale et élégante qui ouvre la voie vers de nouvelles méthodes de fouille permettant une meilleure interaction/prise en compte des préférences de l'utilisateur. Par ailleurs, Ce mode d'interaction permet aussi d'envisager de nouvelles perspectives très prometteuses comme la prise en compte de l'évolution des données ou encore la navigation à travers le cube de skypatterns.

D'autre part, le but était de répondre au verrou du passage à l'échelle des approches déclaratives existantes qui reposent sur un encodage booléen. La taille de cet encodage dépend du nombre de transactions du jeu de données et constitue un frein majeur pour traiter des jeux

de grande taille. L'originalité de nos travaux consistait à introduire des techniques utilisées en fouille de données pour améliorer les encodages PPC pour la fouille de séquences. Malgré les multiples défis algorithmiques posés par la fouille de séquences, ce travail fondateur a contribué à repenser la manière dont les approches PPC pouvaient tirer bénéfice de méthodes de fouille. Il à permis de montrer, pour la première fois, qu'une approche PPC pouvait concurrencer (et même surpasser) les algorithmes spécialisés de l'état de l'art.

Ces premiers résultats ne sont que les prémisses de travaux qu'ils restent encore à mener pour mieux prendre en compte les préférences des utilisateurs et améliorer le processus d'extraction de manière significative, tant d'un point de vue quantitatif (i.e efficacité) que qualitatif (i.e. produire des résultats utilisables et si possible compréhensibles par l'utilisateur).

Bilan et perspectives

Le formalisme WCSP est une alternative intéressante pour traiter des problèmes d'optimisation complexes. Dans ce cadre, nous avons conçu et mis en œuvre des méthodes hybrides pour la résolution de problèmes d'optimisation sous contraintes modélisés sous formes de WCSP, faisant coopérer des approches complètes procédant par "séparation et propagation" avec des approches incomplètes procédant par recherche locale. Nous avons en particulier proposé d'exploiter la structure du graphe sous-jacent à un réseau de contraintes dans la résolution. Nous avons ainsi proposé un schéma générique de recherche locale qui exploite la décomposition arborescente pour guider efficacement l'exploration de l'espace de recherche. L'originalité de ce schéma est d'utiliser la décomposition arborescente comme une carte du problème afin de construire des voisinages pertinents. Plusieurs variantes séquentielles et parallèles de ce schéma générique ont été également proposées. Toutes les méthodes développées ont été intégrées et sont désormais disponibles dans le solveur `toulbar2`. Par ailleurs, pour accroître l'expressivité du cadre WCSP, nous avons introduit la notion de décomposition de fonctions de coût globales afin de permettre une intégration simplifiée des contraintes globales (relaxées) dans les réseaux de fonctions de coût. Enfin, nous avons proposé un cadre unifié de relaxations pour différentes contraintes globales existantes permettant de prendre en compte de manière fine des préférences ou des coûts de violation de ces contraintes. Les algorithmes de propagation associés à ces contraintes sont obtenus par des représentations fines en termes de flots dans des graphes. Nous avons montré comment les contraintes globales relaxées pouvaient coder élégamment et efficacement un problème réel complexe (NRPs : *Nurse Rostering Problems*).

Les premiers travaux initiés dans l'équipe sur les méthodes déclaratives proposant de modéliser et de résoudre des problèmes d'extraction de motifs n-aires a été un facteur stimulant pour développer de nouvelles activités de recherche dans cette direction. Nos travaux avaient pour ambition de faire face à deux verrous majeurs des approches déclaratives actuelles que sont la définition de méthodes efficaces sur de grands ensembles de données, avec un point de vue original consistant à introduire des techniques utilisées en fouille de données pour améliorer les performances de la PPC pour la fouille et la sélection de motifs potentiellement intéressants reposant sur des caractéristiques qui impliquent plusieurs motifs locaux comme les motifs Pareto-optimaux ou les top-k motifs souples. Enfin, nous avons proposé un cadre général mettant en œuvre les contraintes souples de seuil dans un extracteur de motifs en utilisant la relaxation de contraintes en PPC.

Mes perspectives de recherche s'inscrivent dans le prolongement de mes travaux sur les approches déclaratives pour la fouille de données. Notre objectif est de fournir de nouvelles méthodes de fouille de données génériques, permettant d'apporter des réponses concrètes aux trois verrous suivants :

1. le passage à l'échelle des approches déclaratives,
2. la prise en compte des préférences des utilisateurs durant le processus d'extraction,
3. l'extraction efficace d'ensembles de motifs et l'introduction de mécanismes d'optimisation dans le processus de fouille.

Encodages PPC pour la fouille de grands ensembles de données. Un premier axe de mon projet de recherche est de créer une rupture dans la problématique de la fouille de grands ensembles de données en utilisant des approches déclaratives. Une première piste consistera à proposer des représentations plus concises permettant de traiter des jeux de données de grande taille. Une seconde piste très prometteuse consistera à tirer pleinement profit des mécanismes d'élagage des algorithmes spécialisés pour améliorer les performances des approches PPC pour la fouille. Cette piste, qui a été explorée avec succès pour la fouille de motifs séquentiels, s'est poursuivie dans la cadre de la fouille de motifs ensemblistes. Dans ce cadre, nous avons proposé une nouvelle contrainte globale pour l'extraction de motifs fréquents fermés. Grâce aux règles de filtrage proposées, cette contrainte globale assure la GAC avec une complexité cubique en temps et quadratique en espace [Lazaar *et al.*, 2016]. Les résultats expérimentaux sur plusieurs jeux de données réels montrent que notre approche parvient à passer à l'échelle et surpasse clairement l'approche proposée dans [De Raedt *et al.*, 2008] qui est devenue de facto le standard pour de nombreuses tâches de fouille. Notre objectif est de tirer parti de cette nouvelle modélisation pour s'attaquer au problème des k-pattern sets, en imposant des contraintes sur un ensemble de k motifs. Notre ambition à court terme est d'élaborer de nouvelles contraintes globales pour l'extraction de motifs plus riches tels que les motifs fermés, les sous-groupes pertinents ou encore les skypatterns. Cela devrait répondre, en grande partie, à un problème récurrent en fouille de données, celui de la sélection des motifs pertinents parmi le grand nombre de motifs découverts.

Fouille de données interactive. Un autre défi majeur des algorithmes de fouille actuels est la prise en compte des préférences des utilisateurs dans le processus de fouille. Cela soulève la question de savoir comment placer l'utilisateur au cœur du processus d'extraction ? Un second axe de mon projet de recherche est de développer de nouvelles méthodes où l'utilisateur sera en mesure d'exprimer ses préférences sans être limités par les paramètres de la méthode afin de définir les motifs attendus. Pour réaliser cela, nous proposons d'explorer les directions suivantes :

— *Apprentissage des préférences utilisateurs.* L'objectif est de construire un système capable d'apprendre, en interaction avec un analyste, les préférences de celui-ci, à partir de motifs intéressants issus des données. Notre idée est d'alterner phases de fouille de motifs et phases d'apprentissage sur les motifs intéressants selon le scénario suivant. À partir d'une requête initiale de l'analyste, le système présente un premier ensemble de motifs a priori pertinents : (1) l'analyste sélectionne certains de ces motifs, les désignant comme réellement intéressants pour lui ; (2) le système considère ces motifs comme des exemples des préférences de l'expert et affine son "profil" (i.e. le système apprend les préférences de l'utilisateur) ; (3) une nouvelle collection de motifs est extraite en utilisant ces préférences mises à jour, celle-ci est présentée à l'utilisateur, et retour à l'étape (1).

  Pour apprendre les préférences de l'utilisateur, nous envisageons d'utiliser les techniques d'apprentissage fondées sur les réseaux de préférences. L'hypothèse de départ est que l'utilisateur a une préférence implicite entre toute paire de motifs, mais qu'il ne peut pas exprimer cette relation de préférence pour toutes les paires possibles. Nous souhaitons en particulier apprendre une relation de préférence sur des descripteurs pour exprimer l'intérêt des motifs selon une certaine fonction d'utilité.

— *Acquisition de contraintes.* Formuler explicitement les contraintes que doivent satisfaire les motifs à extraire est souvent une tâche difficile. Cela soulève la question de savoir s'il est possible d'apprendre (semi-)automatiquement ces contraintes ou leurs formulations à partir de données et des préférences de l'utilisateur.

— *Navigation dans le cube de skypatterns.* Les skypatterns, introduits à la section 5.2, constituent une autre façon intéressante pour l'utilisateur d'exprimer des préférences à l'aide d'une relation de dominance. De plus, grâce à la structure de cube de skypatterns, il devient possible d'observer l'évolution des ensembles de skypatterns à travers les nœuds du treillis et ainsi d'analyser et de comprendre les différents facteurs de dominance. Notre objectif est de proposer un ensemble d'opérateurs pour permettre à l'utilisateur final une navigation efficace à travers les skypatterns et/ou toutes les mesures possibles. Parmi les opérateurs possibles, nous pouvons citer :
  — L'ajout/retrait de mesures ;
  — L'ajout/retrait de données ;
  — Recommandation d'un ensemble de skypatterns similaires à un skypattern donné.

  Toutefois, afin de réduire le coût de calcul et d'explosion de l'espace de stockage lié au cube de skypatterns, nous proposons d'exploiter des représentations plus réduites pour faire de la navigation. Notre idée est d'éliminer les skypatterns considérés comme redondants des différents nœuds du cube. L'un des principaux intérêts de cette approche en plus de la réduction importante du coût de stockage est l'efficacité de la mise à jour des données du cube réduit.

Apports de la PL-0/1 pour la modélisation et l'extraction d'ensembles de motifs. Un problème récurrent en extraction de motifs est la sélection de motifs pertinents parmi le grand ensemble de motifs découverts. Pour réduire le nombre de motifs extraits et donc de faciliter l'analyse du résultat de la fouille est l'extraction de motifs de plus haut niveau reposant sur des caractéristiques qui impliquent plusieurs motifs locaux. Ces motifs sont appelées *ensembles de motifs* ou *pattern sets* (cf. section 4.1.5). Il existe deux grandes approches déclaratives pour l'extraction de motifs dont l'intérêt dépend des autres motifs extraits : les *motifs n-aires* [Khiari *et al.*, 2010] et les *k-pattern sets* [Guns *et al.*, 2013]. Toutefois, ces deux formalismes sont confrontés au même problème, qui est la capacité à pouvoir gérer de grands ensembles de données. De plus, ils peuvent parfois être restrictifs en ne considérant que des tâches spécifiques, ou en exigeant que le nombre de motifs soit fixé à l'avance. D'un point de vue fouille de données, ce projet de recherche a pour ambition (i) d'étudier les apports de la PL-0/1 pour la modélisation et l'extraction d'ensembles de motifs qui est un problème plus général que le clustering conceptuel et (ii) d'introduire des mécanismes d'optimisation dans les processus de fouille de données afin de se focaliser sur les meilleurs motifs. Cette démarche est une façon à la fois de limiter le nombre de motifs et de se concentrer sur l'ensemble des motifs a priori les plus utiles.

D'un point de vue résolution, pour faire face aux grands ensembles de données, nous proposons d'étudier les apports de la *génération de colonnes* lors de l'utilisation du PL-0/1. En effet, pour les grands ensembles de données, les programmes linéaires correspondants sont trop volumineux pour tenir compte de toutes les variables de décision explicitement. L'idée principale de la génération de colonnes est de commencer la résolution par un PL initial, constitué d'un nombre restreint de variables (ou colonnes), choisies par exemple de manière heuristique, puis de générer au fur et à mesure des variables *améliorantes*, c'est-à-dire des variables qui n'ont pas été mise dans le PL initialement mais qui sont susceptibles d'améliorer la fonction objective. Ce travail est

actuellement amorcé dans le cadre de nos travaux sur le clustering conceptuel.

Annexes

Annexe A

Modélisation CSP de l'instance GPOST

A.1 Description du problème

Le site du groupe de recherche du ASAP (Automated Scheduling, Optimisation and Planning) de l'université de Nottingham (`http://www.cs.nott.ac.uk/~tec/NRP/`) recense une large collection de NRPs. L'exemple que nous présentons correspond à l'instance `GPOST` du site ASAP.

Exemple A.2. Le personnel est composé de huit infirmières (4 à temps plein (A,B,C,D) et 4 à mi-temps (E,F,G,H)) pouvant travailler dans les deux équipes suivantes : l'équipe du matin M (8h00 à 17h00) et l'équipe de nuit N (20h00 à 7h00). Une infirmière engagée à plein-temps doit travailler en moyenne 36 heures par semaine, et une infirmière à mi-temps doit travailler en moyenne 18 heures. Une infirmière ne peut travailler que dans une équipe par jour. L'équipe de nuit du vendredi est considérée comme une équipe de week-end. Le but est de trouver un planning pour une période de quatre semaines qui satisfasse les contraintes d'intégrité et respecte au mieux les contraintes de préférence.

— **Contraintes d'intégrité**

(H1) Les équipes du matin et de nuit sont respectivement composées de 3 et 1 infirmières.

(H2) Chaque infirmière à plein-temps doit travailler 18 jours sur les 4 semaines du planning, alors qu'une infirmière à mi-temps doit travailler 10 jours.

(H3) Chaque infirmière doit travailler au plus quatre nuits, dont au plus 3 sont consécutives.

(H4) Chaque infirmière ne doit pas travailler plus de 6 jours consécutifs.

(H5) Sur trois week-ends consécutifs au mois un doit être libre.

(H6) Après une nuit travaillée, une infirmière doit avoir 48 heures de repos.

— **Contraintes de préférence**

(S1) Une nuit de travail isolée pour une infirmière est pénalisée par un coût de 100.

(S2 Un jour de travail isolé pour une infirmière est pénalisé par un coût de 100.

(S3) Un jour de repos isolé pour une infirmière est pénalisé par un coût de 10.

(S4) Un week-end incomplet engendre une violation de coût 100.

(S5) Chaque infirmière à plein-temps doit travailler en équipe M entre 4 et 5 fois par semaine.

(S6) Chaque infirmière à plein-temps doit travailler en équipe N entre 4 et 5 fois par semaine.
(S7) Chaque infirmière à mi-temps doit travailler en équipe M entre 2 et 3 fois par semaine.
(S8) Chaque infirmière à mi-temps doit travailler en équipe N entre 2 et 3 fois par semaine.
(S9) Pour S5..S8, tout écart δ est pénalisé d'un coût de δ^2.
(S10) Chaque infirmière à plein-temps doit travailler de 4 à 6 jours consécutifs.
(S11) Chaque infirmière à mi-temps doit travailler de 2 à 3 jours consécutifs.
(S12) Pour S10..S11, tout écart δ est pénalisé d'un coût de δ^2.

— **Contraintes d'infirmières**
(I1) L'infirmière A travaille en équipe M les deux premières journées.
(I2) L'infirmière C travaille en équipe M les deux premières journées.
(I3) L'infirmière E travaille en équipe M les deux premières journées.
(I4) L'infirmière D travaille en équipe N les deux premières journées.

A.2 Modélisation sous forme d'un CSP

Soit l'ensemble des infirmières $I = FI \cup PI$ avec $FI = \{A, B, C, D\}$ représentant l'ensemble des infirmières à plein-temps et $PI = \{E, F, G, H\}$ désignant celles à mi-temps. Soit également l'ensemble des jours du planning $J = [1..28]$ et J_{we} le sous-ensemble représentant les numéros de semaines du planning.

Variables et domaines. À chaque infirmière $i \in I$ et chaque jour $j \in J$, est associé une variable x_j^i. Tous les domaines initiaux sont identiques et contiennent les équipes du matin M et de nuit N. Afin de pouvoir modéliser les périodes de repos des infirmières, une troisième équipe R est ajoutée. Ainsi, on a :

$$X = \{x_j^i \mid i \in I, j \in J\}$$
$$\forall i \in I, \forall j \in J,\ D_j^i = \{M, N, R\}$$

Contraintes. Nous montrons à présent comment il est possible d'exploiter les contraintes globales présentées précédemment ainsi que leurs versions relâchées avec préférences pour modéliser les contraintes d'intégrité et de préférence des infirmières.

i) Contraintes d'équipes

La contrainte d'équipe de l'exemple A.2 (règle (H1)) est une des contrainte d'intégrité et peut donc être modélisée par une contrainte `Gcc`. Les bornes inférieures l et supérieures u sont décrites selon l'ordre suivant : M, N, R.

$$\forall j \in J, \quad \texttt{Gcc}(\{x_j^A..x_j^H\}, [3,1,0], [3,1,8]) \quad (H1)$$

Dans ces contraintes, les bornes inférieures et supérieures pour les infirmières en repos sont fixées respectivement à 0 et 8. Aucune restriction n'étant faite sur ces bornes, elles sont fixées à leur valeur extrême.

ii) Contraintes sur les charges de travail

Les contraintes sur les charges de travail de l'exemple A.2 (règles (H2) et (H3)) sont des contraintes d'intégrité et peuvent donc être modélisées par des contraintes `Gcc`.

$$\forall i \in FI, \quad \texttt{Gcc}(\{x_1^i..x_{28}^i\}, [0,0,10], [18,4,10]) \quad (H2)(H3)$$
$$\forall i \in PI, \quad \texttt{Gcc}(\{x_1^i..x_{28}^i\}, [0,0,18], [10,4,18]) \quad (H2)(H3)$$

iii) Contraintes restreignant les affectations aux équipes de travail

Les règles (S5),(S6) et (S7),(S8) peuvent être agglomérées en une seule contrainte Σ-`Gcc`. Toutefois, en raison de la croissance quadratique de la violation, il n'est pas possible de modéliser cette violation avec la mesure μ_{dec}^{Σ}. Nous proposons de fixer les bornes inférieures et supérieures des deux équipes M et N à leurs valeurs minimales et maximales et de contraindre le nombre de repos en conséquence. Ainsi, il suffit de quantifier la violation sur le nombre de repos en utilisant une mesure quadratique noté μ_{quad}.

$$\forall i \in FI,\ w \in J_{we},\ \Sigma\text{-}\texttt{Gcc}([x_{1+w*7}^i..x_{7+w*7}^i], z_i^{\alpha}, \mu_{quad}, [0,0,2], [7,7,3]) \quad (S5)(S6)$$
$$\forall i \in PI,\ w \in J_{we},\ \Sigma\text{-}\texttt{Gcc}([x_{1+w*7}^i..x_{7+w*7}^i], z_i^{\beta}, \mu_{quad}, [0,0,4], [7,7,5]) \quad (S7)(S8)$$

iv) Les contraintes de séquence

Le dernier ensemble de règles porte sur un ensemble d'enchaînements d'équipes qui ne doivent pas (ou le moins possible) être présents dans le planning final.

L'automate de la figure A.2 permet de modéliser à la fois les contraintes d'intégrité (H3) et (H6), ainsi que la contrainte de préférence (S1) sur les enchaînements de l'équipe de nuit.

$$\forall i \in I,\ \texttt{costRegular}([x_1^i..x_{28}^i], \Pi_1, z_i^{\gamma}) \quad (H3), (H6), (S1)$$

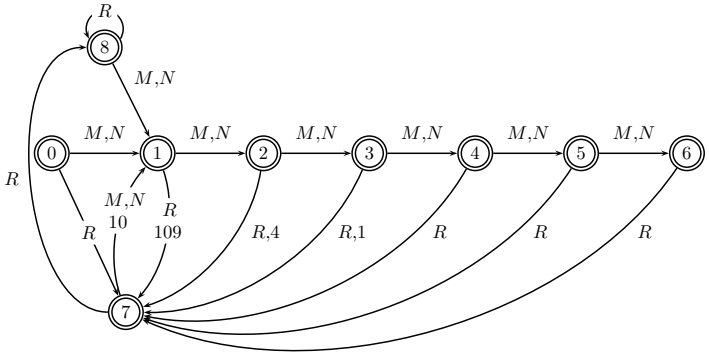


FIGURE A.1 – Automate Π_2 associé aux règles (H4),(S2), (S3) et (S10).

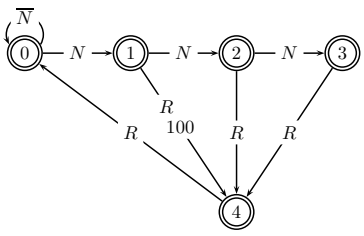


FIGURE A.2 – Automate Π_1 associé aux règles (H3), (H6) et (S1).

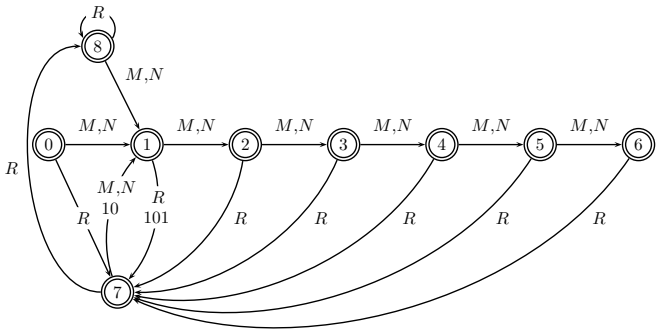


FIGURE A.3 – Automate Π_3 associé aux règles (H4),(S2), (S3) et (S11).

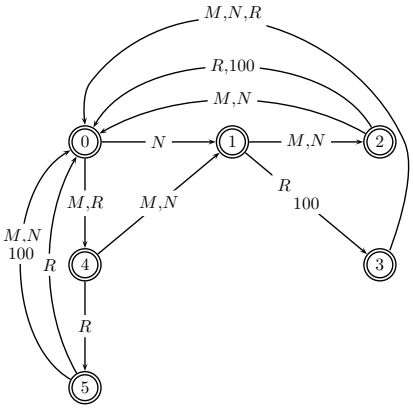


FIGURE A.4 – Automate Π_4 associé à la règle (S4).

La règle d'intégrité (H4) et les règles de préférence (S2), (S3) et (S10), S(11) sur les enchaînements d'équipes de travail peuvent être agglomérées en un même automate et modélisées par une seule contrainte `costRegular`.

$$\forall i \in FI,\ \texttt{costRegular}([x_1^i..x_{28}^i], \Pi_2, z_i^\delta) \quad (H4),(S2),(S3),(S10)$$
$$\forall i \in PI,\ \texttt{costRegular}([x_1^i..x_{28}^i], \Pi_3, z_i^\varepsilon) \quad (H4),(S2),(S3),(S11)$$

L'automate de la figure A.4 est associé à la contrainte de séquence qui porte sur les week-ends (S4).

$$\forall i \in I,\ \texttt{costRegular}([x_5^i, x_6^i, x_7^i, .., x_{26}^i, x_{27}^i, x_{28}^i], \Pi_4, z_i^\zeta) \quad (S4)$$

L'automate de la figure A.5 est associé à la contrainte de séquence (H5) sur l'enchaînement des week-ends.

$$\forall i \in I,\ \texttt{Regular}([x_5^i, x_6^i, x_7^i, .., x_{26}^i, x_{27}^i, x_{28}^i], \Pi_5, z_i^\eta) \quad (H5)$$

v) Les contraintes d'infirmières

Ces règles correspondent généralement à des requêtes personnelles (i.e. congés souhaités) et peuvent être modélisées par des contraintes unaires d'égalité ou de différence.

Variable objectif. La variable objectif Z correspond à la somme de toutes les variables de coûts :

$$Z = \sum_{i \in I} (z_i^\alpha + z_i^\beta + z_i^\gamma + z_i^\delta z_i^\varepsilon) + z_i^\zeta) + z_i^\eta)$$

Comme le montre la figure A.6, la modélisation d'un NRP à l'aide des contraintes globales relaxées est concise et élégante. Plusieurs règles de même nature peuvent être agglomérées en une seule contrainte.

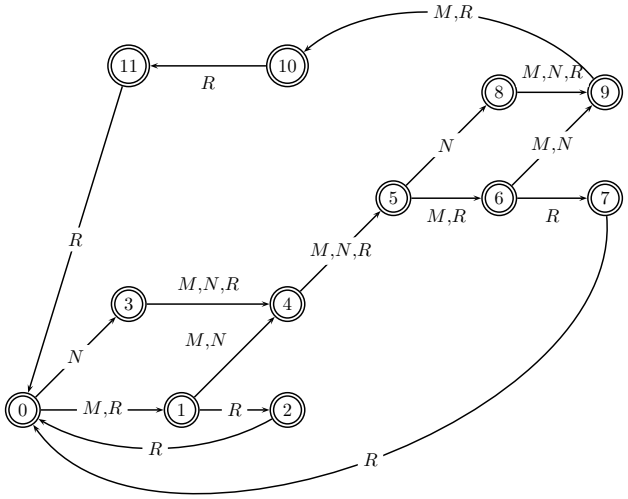


FIGURE A.5 – Automate Π_5 associé à la règle (H5).

- Variables et domaines :	
$I = [A..H], FI = [A..D], PI = [E..H]$	
$X = \{x_j^i \mid i \in I, j \in J\}$	
$\forall i \in I, \forall j \in J,\ D_j^i = \{M, N, R\}$	
- Constraintes d'équipes :	
$\forall j \in J,\ \texttt{Gcc}(\{x_j^A..x_j^H\}, [3,1,0], [3,1,8])$	$(H1)$
- Constraintes sur la charge de travail :	
$\forall i \in FI,\ \texttt{Gcc}(\{x_1^i..x_{28}^i\}, [0,0,10], [18,4,10])$	$(H2..3)$
$\forall i \in PI,\ \texttt{Gcc}(\{x_1^i..x_{28}^i\}, [0,0,18], [10,4,18])$	$(H2..3)$
$\forall i \in FI,\ w \in J_{we},\ \Sigma\texttt{-Gcc}([x_{1+w*7}^i..x_{7+w*7}^i], z_i^\alpha, \mu_{quad}, [0,0,2], [7,7,3])$	$(S5..6)$
$\forall i \in PI,\ w \in J_{we},\ \Sigma\texttt{-Gcc}([x_{1+w*7}^i..x_{7+w*7}^i], z_i^\beta, \mu_{quad}, [0,0,4], [7,7,5])$	$(S7..8)$
- Constraintes de séquence :	
$\forall i \in I,\ \texttt{costRegular}([x_1^i..x_{28}^i], \Pi_1, z_i^\gamma)$	$(H3, H6, S1)$
$\forall i \in FI,\ \texttt{costRegular}([x_1^i..x_{28}^i], \Pi_2, z_i^\delta)$	$(H4, S2, S3, S10)$
$\forall i \in PI,\ \texttt{costRegular}([x_1^i..x_{28}^i], \Pi_3, z_i^\varepsilon)$	$(H4, S2, S3, S11)$
$\forall i \in I,\ \texttt{costRegular}([x_5^i, x_6^i, x_7^i, .., x_{26}^i, x_{27}^i, x_{28}^i], \Pi_4, z_i^\zeta)$	$(S4)$
$\forall i \in I,\ \texttt{Regular}([x_5^i, x_6^i, x_7^i, .., x_{26}^i, x_{27}^i, x_{28}^i], \Pi_5, z_i^\eta)$	$(H5)$
- Variables de coût :	
$Z = \sum_{i \in I}(z_i^\alpha + z_i^\beta + z_i^\gamma + z_i^\delta + z_i^\varepsilon) + z_i^\zeta + z_i^\eta)$	

FIGURE A.6 – Modélisation CSP de l'instance `GPOST` de l'exemple A.2.

Annexe B

Computing Skypattern Cubes

Article [Ugarte *et al.*, 2014b] : paru dans les actes du congrès ECAI en 2014

Auteurs : Willy Ugarte, Patrice Boizumault, Samir Loudni, Bruno Crémilleux

Abstract

We introduce skypattern cubes and propose an efficient bottom-up approach to compute them. Our approach relies on derivation rules collecting skypatterns of a parent node from its child nodes without any dominance test. Non-derivable skypatterns are computed on the fly thanks to Dynamic CSP. The bottom-up principle enables to provide a concise representation of the cube based on skypattern equivalence classes without any supplementary effort. Experiments show the effectiveness of our proposal.

B.1 Introduction

The notion of skyline queries [Börzsönyi *et al.*, 2001] has been recently integrated into the pattern discovery paradigm to mine skyline patterns (called *skypatterns*) [Soulet *et al.*, 2011, Ugarte *et al.*, 2014c]. Given a set of measures, skypatterns are patterns based on a Pareto-dominance relation for which no measure can be improved without degrading the others. As an example, a user may prefer a pattern with a high frequency, large size and a high confidence. In this example, a pattern x_i dominates another pattern x_j if $freq(x_j) \geq freq(x_i)$, $size(x_j) \geq size(x_i)$, $confidence(x_j) \geq confidence(x_i)$ where at least one strict inequality holds. Given a set of patterns, the skypattern set contains the patterns that are not dominated by any other pattern. Skypatterns are highly interesting because they do not require any threshold on the measures and the dominance relation gives to the skypatterns a global interest with semantics easily understood by the user.

In practice, users do not know the exact role of each measure which be used and it is difficult to beforehand select the most appropriate set of measures. Users would like to keep all the measures potentially useful, look what happens on a skypattern set by removing or adding a measure to evaluate the impact of measures and then converge to a proper skypattern set. Similarly to the notion of the skyline cube in the database [Raïssi *et al.*, 2010], users would like to have available the *skypattern cube*. Each element of the cube is a *node* which associates to a subset of the

measures its skypattern set. By comparing two neighboring nodes, which are differentiated by adding or removing one measure, users can observe the new skypatterns and the ones which die out. It greatly helps to better understand the role of the measures. Moreover, users can spot that different subsets of measures have the same skypattern set : such an equivalence class over subsets of measures shows useless measures (i.e., measures that can be added to a set of measures without changing the skypattern set). To sum up, the cube is the proper structure to enable various user queries in an efficient manner and to discover the most interesting skypattern sets.

More formally, given a set M of n measures, the 2^n-1 possible non-empty skypattern subsets should be precomputed to efficiently handle various queries of users. A baseline method to build the skypattern cube needs the computing of the skypatterns on every measure subset and incurs a prohibitive cost. Therefore the problem of efficient computing of the skypattern cube is the focus of this paper.

For computing the skypattern cube, we propose a bottom-up approach motivated by the following observations. First, we formally give two derivation rules providing an easy way to automatically infer a large proportion of the skypatterns of a *parent node* from the skypattern sets of its *child nodes* without any dominance test (if k measures are associated to a parent node, its child nodes are the nodes defined by the $\binom{k}{k-1}$ subsets of $k-1$ measures). For the new skypatterns of a parent node (i.e., skypatterns which are not skypatterns in its child nodes), we give an efficient technique based on dynamic CSP to mine them on the fly. We show that in practice the number of new skypatterns remains low. Second, we demonstrate how the bottom-up principle enables us to determine skypattern equivalence classes without any supplementary effort. This result has the advantage to provide a more concise cube, highlighting the measures giving the same skypattern set. Third, experiments conducted on real-life datasets show the practical effectiveness achieved by our formal results. To sum up, to the best of our knowledge, we designed the first method to build the skypattern cube without enumerating and mining all the possible skypatterns.

This paper is organized as follows. After introducing the background in Section B.2, we present in Section B.3 the formal properties to automatically infer skypatterns and build the concise representation of the cube. Section B.4 describes our CSP method to mine the new skypatterns. We discuss related work in Section B.5. Section B.6 presents the experiments and we conclude in Section B.7.

B.2 Skypattern Cube

B.2.1 Context and Definitions

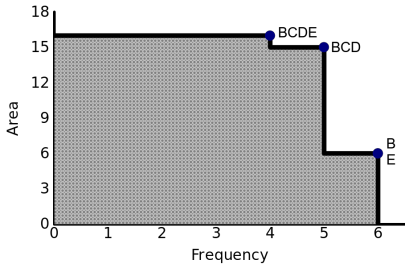
Let $\mathcal{I}$ be a set of distinct literals called *items*. An itemset (or pattern) is a non-null subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L}_\mathcal{I} = 2^\mathcal{I} \backslash \emptyset$. A transactional dataset $\mathcal{T}$ is a multiset of patterns of $\mathcal{L}_\mathcal{I}$. Fig. B.1a depicts a transactional dataset $\mathcal{T}$ where each transaction (or pattern) t_i is described by items denoted $A,\ldots,F$. The traditional example is a supermarket database in which each transaction corresponds to a customer and every item in the transaction is a product bought by the customer. An attribute (*price*) is associated to each product (see Fig. B.1a).

Constraint-based pattern mining aims at extracting all patterns x of $\mathcal{L}_\mathcal{I}$ satisfying a query $q(x)$ (conjunction of constraints) which is usually called *theory* [Mannila et Toivonen, 1997] :

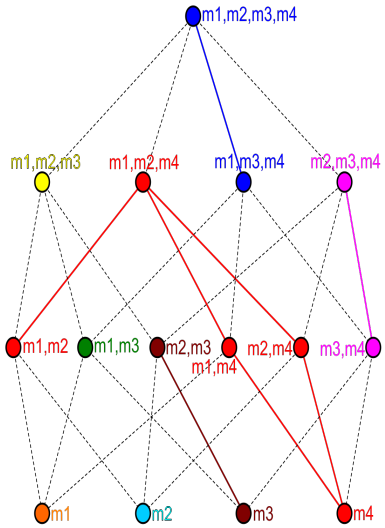
Trans.	Items					
t_1		B			E	F
t_2		B	C	D		
t_3	A				E	F
t_4	A	B	C	D	E	
t_5		B	C	D	E	
t_6		B	C	D	E	F
t_7	A	B	C	D	E	F

Item	A	B	C	D	E	F
Price	30	40	10	40	70	55

(a) Transactional dataset $\mathcal{T}$.



(b) Skypatterns for M=$\{freq, area\}$.



(c) Lattice associated to M.

Subset of M	**Skypattern set**
$\{m_1, m_2, m_3, m_4\}$	{BCDE, BCD, BDE, EF, BE, E}
$\{m_1, m_2, m_3\}$	{BCDE, BCD, BE, E}
$\{m_1, m_2, m_4\}$	{E}
$\{m_1, m_3, m_4\}$	{BCDE, BCD, BDE, EF, BE, E}
$\{m_2, m_3, m_4\}$	{BCDE, BDE, EF, E}
$\{m_1, m_2\}$	{E}
$\{m_1, m_3\}$	{BCDE, BCD, B, E}
$\{m_1, m_4\}$	{E}
$\{m_2, m_3\}$	{BCDE}
$\{m_2, m_4\}$	{E}
$\{m_3, m_4\}$	{BCDE, BDE, EF, E}
$\{m_1\}$	{B, E}
$\{m_2\}$	{ABCDEF, ABCEF, ABDEF, ABEF, ABCDE, ABCE, ABDE, ABE, ACDEF, ACEF, ACDE, ACE, ADEF, ADE, AEF, AE, BCDEF, BCEF, CDEF, CEF, BCDE, BCE, CDE, CE, BDEF, DEF, BDE, DE, BEF, EF, BE, E}
$\{m_3\}$	{BCDE}
$\{m_4\}$	{E}

(d) Skypattern cube for M.

FIGURE B.1 – M=$\{m_1$: $freq, m_2$: max, m_3: $area, m_4$: $mean\}$.

$Th(q) = \{x \in \mathcal{L}_\mathcal{I} \mid q(x)\ is\ true\}$. A common example is the frequency measure leading to the minimal frequency constraint ($freq(x) \geq \theta$). The latter provides patterns x having a number of occurrences in the dataset exceeding a given minimal threshold θ. There are other usual measures for a pattern x :

- $area(x) = freq(x) \times size(x)$.
- $min(x.att)$ (resp. $max(x.att)$) is the smallest (resp. highest) value of the set of item values of x for attribute att.
- $mean(x) = (min(x.att) + max(x.att))/2$.

For the dataset in Fig. B.1a, $freq(BC)$=5, $area(BC)$=10 and $mean(BCD.price)$=25.

A pattern x_i is closed w.r.t. a measure m iff $\forall x_j \supsetneq x_i, m(x_j) \neq m(x_i)$. The set of closed patterns is a compact representation of the patterns (i.e we can derive all the patterns with their exact value for m from the closed ones). This definition is straightforwardly extended to a set of measures M.

B.2.2 Skypatterns

As stated above, skypatterns enable to express a user-preference point of view according to a dominance relation [Soulet *et al.*, 2011].

Définition B.10 (Pareto Dominance)
Given a set of measures M, a pattern x_i dominates another pattern x_j w.r.t. M (denoted by $x_i \succ_M x_j$), iff $\forall m \in M, m(x_i) \geq m(x_j)$ and $\exists m \in M, m(x_i) > m(x_j)$.

Définition B.11 (Skypattern and skypattern operator)
Given a set of measures M, a skypattern w.r.t. M is a pattern not dominated w.r.t. M. The skypattern operator $Skypattern(M)$ returns all the skypatterns w.r.t. M : $Skypattern(M) = \{x_i \in \mathcal{L}_\mathcal{I} \mid \nexists x_j \in \mathcal{L}_\mathcal{I}, x_j \succ_M x_i\}$.

In the sequel of this paper, we adopt that $Skypattern(M) = Skypattern(M)$.

Exemple B.3. From $\mathcal{T}$ and with M=$\{freq, area\}$, pattern BCD dominates pattern BC as $freq(BCD)$=$freq(BC)$=5 and $area(BCD)$>$area(BC)$ (cf. Fig. B.1a). Fig. B.1b provides a graphical representation of $Skypattern(M) = \{BCDE, BCD, B, E\}$. The shaded area is called the *forbidden area* since it cannot contain any skypattern. The other part is called the *dominance area*.

Let M be a set of measures. Two patterns x_i and x_j are *indistinct* w.r.t. M (denoted by $x_i =_M x_j$) iff $\forall m \in M, m(x_i) = m(x_j)$. Two patterns x_i and x_j are *incomparable* w.r.t. M (denoted by $x_i \prec\succ_M x_j$) iff $(x_i \not\succ_M x_j)$ and $(x_j \not\succ_M x_i)$ and $(x_i \neq_M x_j)$.

Définition B.12 (Incomparable Skypattern)
A pattern $x \in Skypattern(M)$ is incomparable w.r.t M iff $\forall x_i \in Skypattern(M)$ s.t. $x_i \neq x$, $x_i \prec\succ_M x$.

Définition B.13 (Indistinct Skypattern)
A pattern $x \in Skypattern(M)$ is indistinct w.r.t. M iff $\exists x_i \in Skypattern(M)$ s.t. $(x_i \neq x) \wedge (x_i =_M x)$.

Incomparable skypatterns and indistinct ones w.r.t. M constitute a partition of $Skypattern(M)$. Moreover, $=_M$ is an equivalence relation (i.e., the relation is reflexive, symmetric and transitive). So, indistinct skypatterns can be gathered into a group. This remark will be precious for the derivation rule on indistinct skypatterns.

Définition B.14 (Indistinct Skypattern Group (ISG))
$S \subseteq Skypattern(M)$ is an indistinct skypattern group w.r.t. M, iff $|S| \geq 2$ and $\forall x_i, x_j \in S$, $(x_i =_M x_j)$ and $\forall x_i \in S$, $\forall x_j \in Skypattern(M) \setminus S$, $x_i \prec\succ_M x_j)$.

Exemple B.4. For M=$\{freq, area\}$, $BCDE$ and BCD are incomparable. B and E are indistinct and belong to the same ISG.

B.2.3 Skypattern Cube

Let M be a set of measures. We define the skypattern cube over M which consists of the $2^{|M|}-1$ skypattern sets $Skypattern(M_u)$ on all possible non-empty subset $M_u \subseteq M$.

Définition B.15 (Skypattern Cube)
Let M be a set of measures. $SkyCube(M) = \{(M_u, Skypattern(M_u)) \mid M_u \subseteq M, M_u \neq \emptyset\}$.

Exemple B.5. Consider the dataset in Fig. B.1a. Fig. B.1c depicts the lattice associated to M. Fig. B.1d associates to each non-empty subset of M its skypattern set.

B.3 Derivation Rules and Concise Representation

This section presents our *bottom-up* approach for computing the skypattern cube. The key idea is to collect the skypatterns of a parent node from the skypatterns of its child nodes. Then we compute the missing skypatterns of the node (i.e., skypatterns that are not skypatterns in its child nodes and thus not yet mined). We show how a concise representation of the cube is straightforwardly provided.

B.3.1 Derivation Rules

Theorems B.1 states that all the incomparable skypatterns of a child node remain incomparable skypatterns in its parent nodes. Theorem B.2 exhibits the indistinct skypatterns of a child node that remain skypatterns in its parent nodes. These two theorems define two derivation rules that enable to derive a subset of skypatterns of a parent node.

Théorème B.1 (Incomparability Rule)
Let $M_u \subseteq M$. If x is an incomparable skypattern w.r.t. M_u then $\forall m \in M \backslash M_u, x \in Skypattern(M_u \cup \{m\})$. Moreover x is incomparable w.r.t. $M_u \cup \{m\}$.

Proof (By Contradiction) : Assume that x is an incomparable skypattern w.r.t. M_u, and $\exists m \in M \backslash M_u$ s.t. $x \notin Skypattern(M_u \cup \{m\})$. So, $\exists y \neq x \in \mathcal{L}_{\mathcal{I}}$, $(y \succ_{M_u \cup \{m\}} x)$ i.e. (1) $\forall m_i \in M_u \cup \{m\}, m_i(y) \geq m_i(x)$ and (2) $\exists m_j \in M_u \cup \{m\}, m_j(y) > m_j(x)$. For (2), there are two cases :

1. $(m_j{=}m)$. As $x \in Skypattern(M_u)$, $\forall m_i \in M_u, m_i(x) \geq m_i(y)$. From (1), we deduce : $\forall m_i \in M_u, m_i(x) = m_i(y)$. So x is indistinct w.r.t. M_u. It contradicts that x is incomparable w.r.t. M_u.
2. $(m_j \in M_u)$. From (1), we have $\forall m_i \in M_u, m_i(y) \geq m_i(x)$. As, $m_j(y) > m_j(x)$, we deduce that $y \succ_{M_u} x$. It contradicts that x is a skypattern w.r.t. M_u. □

Exemple B.6. Let $M_u{=}\{m_1, m_3\}$. $BCDE$ and BCD are incomparable w.r.t. M_u. Theorem B.1 enables to deduce that $BCDE$ and BCD belong to $Skypattern(M_u \cup \{m_2\})$ and $Skypattern(M_u \cup \{m_4\})$.

Théorème B.2 (ISG Rule)
Let $M_u \subseteq M$ and S an ISG w.r.t. M_u. $\forall m \in M \backslash M_u$, each skypattern $x \in S$ s.t. $m(x) = \underset{x_i \in S}{\texttt{Max}}\{m(x_i)\}$ is a skypattern w.r.t. $M_u \cup \{m\}$.

Proof (By Contradiction) : Assume that there exists an ISG S w.r.t. M_u s.t. $\exists m \in M \backslash M_u$ s.t. $\exists x \in S$ s.t. $m(x){=}\underset{x_i \in S}{\texttt{Max}}\{m(x_i)\}$ and $x \notin Skypattern(M_u \cup \{m\})$. So, $\exists y \neq x \in \mathcal{L}_{\mathcal{I}}$, $(y \succ_{M_u \cup \{m\}} x)$ i.e. (1) $\forall m_i \in M_u \cup \{m\}, m_i(y) \geq m_i(x)$ and (2) $\exists m_j \in M_u \cup \{m\}, m_j(y) > m_j(x)$. For (2), there are two cases :

1. $(m_j{=}m)$. As $x \in Skypattern(M_u)$, $\forall m_i \in M_u, m_i(x) \geq m_i(y)$. From (1), we deduce : $\forall m_i \in M_u, m_i(x) = m_i(y)$, i.e. $y \in S$. So, $m(y) \leq m(x)$ (as $m(x) = \underset{x_i \in S}{\texttt{Max}}\{m(x_i)\}$. It contradicts (2).
2. $(m_j \in M_u)$. From (1), we have $\forall m_i \in M_u, m_i(y) \geq m_i(x)$. As, $m_j(y) > m_j(x)$, we deduce that $y \succ_{M_u} x$. It contradicts that x is a skypattern w.r.t. M_u. □

Exemple B.7. $S = \{B, E\}$ is an ISG w.r.t. $\{m_1\}$. Theorem B.2 enables to deduce that :

— $E \in Skypattern(\{m_1, m_2\})$ since $m_2(E) = \underset{x_k \in S}{\texttt{Max}}\{m_2(x_k)\}$
— $E \in Skypattern(\{m_1, m_4\})$ since $m_4(E) = \underset{x_k \in S}{\texttt{Max}}\{m_4(x_k)\}$
— $B, E \in Skypattern(\{m_1, m_3\})$ since $m_3(B)$=$m_3(E)$=$\underset{x_k \in S}{\texttt{Max}}\{m_3(x_k)\}$

Corollary 1 Let S an ISG w.r.t. M_u, and $m \in M \setminus M_u$. Let $S' = \{x \in S \mid m(x) = \underset{x_i \in S}{\texttt{Max}}\{m(x_i)\}$. If S' is a singleton then the unique skypattern is incomparable w.r.t. $M_u \cup \{m\}$ else S' is an ISG w.r.t. $M_u \cup \{m\}$. Finally all $x \in S \setminus S'$ are not skypatterns for $M_u \cup \{m\}$.

B.3.2 Computing a Skypattern Cube

The skypatterns of a node are computed in two steps. First, we collect all the skypatterns which can be derived from its child nodes. Then the missing skypatterns (i.e., non-derivable skypatterns) are computed. We start by defining the *derivable skypatterns*.

Let $M_u \subseteq M$ and $m \in M \setminus M_u$. We define $inc(M_u)$ the set of incomparable skypatterns w.r.t M_u and $ind(M_u, m)$ the set of maximal indistinct skypatterns w.r.t a measure m :

— $inc(M_u) = \{x \in Skypattern(M_u) \mid x \; is \; incomparable \, w.r.t. \, M_u\}$

— $ind(M_u, m) = \underset{ISG \, S \subseteq Skypattern(M_u)}{\bigcup\{x \in S} \mid m(x) = \underset{x_k \in S}{\texttt{Max}}\{m(x_k)\}\}$

$derived(M_u)$ is the set of skypatterns of the node associated to M_u which can be derived from the skypatterns of its child nodes :
— $derived(M_u) = \underset{m \in M_u}{\bigcup} (inc(M_u \setminus \{m\}) \cup ind(M_u \setminus \{m\}, m))$.

First, it is obvious that : $derived(M_u) \subseteq Skypattern(M_u)$ (see Theorem B.1 and B.2). Experiments show that a large proportion of skypatterns are obtained by this way. Moreover, if a skypattern x in a parent node is also a skypattern in at least one of its child nodes, then x will be necessary collected by one of these rules. It is expressed by :
$derived(M_u) = (\underset{m \in M_u}{\bigcup} Skypattern(M_u \setminus \{m\})) \cap Skypattern(M_u)$.

This property shows the power of our derivation rules. (The proof is immediate since, for each node, incomparable and indistinct skypatterns constitute a partition.) However, $derived(M_u)$ can be strictly included in $Skypattern(M_u)$, i.e., some skypatterns are missing. It happens when a skypattern of a node is not a skypattern in any of its child nodes as illustrated by the following example.

Exemple B.8. As $BCDE$ is incomparable w.r.t. $\{m_3\}$, $BCDE \in Skypattern(\{m_1, m_3\})$. As B and E constitute an ISG w.r.t. $\{m_1\}$ and $m_3(B)$=$m_3(E)$, then $B, E \in Skypattern(\{m_1, m_3\})$. But, the derivation rules cannot deduce that $BCD \in Skypattern(\{m_1, m_3\})$.

We compute on the fly the non-derivable skypatterns thanks to a dynamic CSP method described in Section B.4. Moreover, we can go further by detecting a priori that $derived(M_u) = Skypattern(M_u)$ for some M_u and thus avoiding useless computation. Theorem B.3 states a sufficient condition ensuring that $derived(M_u) = Skypattern(M_u)$. Experiments show that this condition is effective in practice.

Algorithme 9 : Bottom-up approach for computing the cube

Input : M : a set of measures, $\mathcal{T}$: a dataset.
Output : The skypattern cube of dataset $\mathcal{T}$ w.r.t. M.

```
1 cube ← ∅;
2 foreach m ∈ M do
3     cube ← cube ∪ {({m}, Skypattern({m}))};
4 for i ← 2 to |M| do
5     foreach M_u ⊂ M s.t. | M_u | = i do
6         cube ← cube ∪ {(M_u, complete(derived(M_u)))};
7 return cube
```

Théorème B.3 (Non-Computing Sufficient Condition)
Let $M_u \subseteq M$. If $\exists m \in M_u$ s.t. $\min_{x \in derived(M_u)} \{m(x)\} = \texttt{Max}_{x \in derived(M_u)} \{m(x)\}$ then $Skypattern(M_u) = derived(M_u)$.

Proof (By Contradiction) : Let $m \in M_u$ a measure s.t. $\min_{x \in derived(M_u)} \{m(x)\} = \texttt{Max}_{x \in derived(M_u)} \{m(x)\}$
Assume that $\exists p \in Skypattern(M_u) \backslash derived(M_u)$, so $m(p) = \min_{x \in derived(M_u)} \{m(x)\} = \texttt{Max}_{x \in derived(M_u)} \{m(x)\}$.
Henceforth $m(p) = \texttt{Max}_{x \in \mathcal{L}_\mathcal{I}} \{m(x)\}$. Thus, $p \in Skypattern(\{m\})$, and :

(i) either p is incomparable w.r.t. $\{m\}$,
(ii) or p is indistinct w.r.t $\{m\}$ with maximal value for m.

From (i) and (ii), $p \in derived(M_u)$ leading to a contradiction. □

Finally, Algorithm 9 gives the pseudo-code of our bottom-up approach. It starts by computing $Skypattern(\{m\})$ for every $m \in M$ and then follows a level-wise strategy : from the lower level, each level of the lattice is constructed by applying the derivation rules and, if needed, the computing of the non derivable skypatterns (function *complete*).

B.3.3 Concise Representation of a Cube

Different subsets of measures may lead to a same set of skypatterns. This observation can be used to provide a concise representation of the cube without loss of information. We define an equivalence relation over subsets of measures having the same skypattern set :

Définition B.16 (Equivalence between sets of measures)
Let M_u and M_v two sets of measures. M_u and M_v are said to be equivalent iff $Skypattern(M_u) = Skypattern(M_v)$.

Exemple B.9. Equivalence classes on our running example are illustrated in Figure B.1c. There are 8 classes : 4 have a cardinality of 1, 3 have a cardinality of 2 and 1 has a cardinality of 5.

Theorem B.4 indicates if a new node built by the addition of a measure to a subset of measures M_u belongs or not to the equivalence class of M_u. It means that equivalence classes can be easily determined thanks to the bottom-up construction of the skypattern cube. In other words, when

our approach is running to extract the skypatterns of the cube, it can also provide a concise representation of the cube without supplementary work.

Théorème B.4 (Equivalence class)
Let $M_u \subseteq M$ and $m \in M \setminus M_u$. $Skypattern(M_u \cup \{m\}) = Skypattern(M_u)$ iff (1) all indistinct skypatterns w.r.t. M_u are indistinct skypatterns w.r.t. $M_u \cup \{m\}$ and (2) $Skypattern(M_u \cup \{m\})$ $= derived(M_u \cup \{m\})$.

Double inclusion. All incomparable w.r.t. M_u are incomparable w.r.t. $M_u \cup \{m\}$ (Theorem B.1). All indistinct w.r.t. M_u are indistinct w.r.t. $M_u \cup \{m\}$ according to (1). Since incomparable w.r.t. M_u and indistinct w.r.t. M_u form a partition of $Skypattern(M_u)$, $Skypattern(M_u) \subset Skypattern(M_u \cup \{m\})$.
According to (2), $Skypattern(M_u \cup \{m\}) = derived(M_u \cup \{m\})$. As derived skypatterns w.r.t. $M_u \cup \{m\}$ can only come from $Skypattern(M_u)$, $Skypattern(M_u \cup \{m\}) \subset Skypattern(M_u)$. □

B.4 Mining non-derivable Skypatterns using DCSP

This section describes how the non-derivable skypatterns can be mined using Dynamic CSP [Verfaillie et Jussien, 2005]. The main idea of our approach [Ugarte *et al.*, 2014c], taking benefit from cross-fertilization between CSP and data mining [Guns *et al.*, 2011], is to improve the mining step during the process thanks to constraints dynamically posted and stemming from the current set of the candidate skypatterns. The process stops when the forbidden area cannot be enlarged. Finally, the completeness of our approach is insured by the completeness of the CP solver.

B.4.1 Mining Skypatterns

A Constraint Satisfaction Problem (CSP) $P=(\mathcal{X}, \mathcal{D}, \mathcal{C})$ is defined by a set of variables $\mathcal{X}$, a domain $\mathcal{D}$, which maps every variable $x_i \in \mathcal{X}$ to a finite set of values $D(x_i)$, and a set of constraints $\mathcal{C}$.

A Dynamic CSP [Verfaillie et Jussien, 2005] is a sequence $P_1, P_2, ..., P_n$ of CSP, each one resulting from some changes in the definition of the previous one. These changes may affect every component in the problem definition : variables, domains and constraints. *For our approach, changes are only performed by adding new constraints.* Solving such dynamic CSP involves solving a single CSP with additional constraints posted during search. Each time a new solution is found, new constraints are imposed. Such constraints will survive backtracking and state that next solutions should verify both the current set of constraints and the added ones.

Constraints on the dominance relation are dynamically posted during the mining process. Variable x will denote the (unknown) skypattern we are looking for. Changes are only performed by adding new constraints. So, we consider the sequence $\Phi_1, \Phi_2, \ldots, \Phi_n$ of CSP where M is a set of measures, each $\Phi_i = (\{x\}, \mathcal{L}_{\mathcal{I}}, q_i(x))$ and :

— $q_1(x) = closed_{M'}(x)$
— $q_{i+1}(x) = q_i(x) \wedge \phi(s_i, x)$where s_i is the first solution to $q_i(x)$

First, the constraint $closed_{M'}(x)$ states that x must be a closed pattern w.r.t M', it allows to reduce the number of redundant patterns. Then, the constraint $\phi_i(x) \equiv \neg(s_i \succ_M x)$ states

that the next solution (which is searched) will not be dominated by s_i. Using a short induction proof, we can easily argue that query $q_{i+1}(x)$ looks for a pattern x that will not be dominated by any of the patterns s_1, s_2, ..., s_i.

Each time the first solution s_i to query $q_i(x)$ is found, a new constraint $\phi_i(x)$ is dynamically posted, leading to reduce the search space. This process stops when the forbidden area cannot be further extended (i.e. there exists n s.t. query $q_{n+1}(x)$ has no solution). For skypatterns, $\phi_i(x)$ states that $\neg(s_i \succ_M x)$:

$$\phi_i(x) \equiv \left(\bigvee_{m \in M} m(s_i) < m(x) \right) \vee \left(\bigwedge_{m \in M} m(s_i) = m(x) \right)$$

But, the n extracted patterns s_1, s_2, ..., s_n are not necessarily all skypatterns. Some of them can only be "intermediate" patterns simply used to enlarge the forbidden area. A post processing step must be performed to filter all candidate patterns s_i that are not skypatterns, i.e. for which there exists s_j $(1 \leq i < j \leq n)$ s.t. s_j dominates s_i. So mining skypatterns is achieved in a two-steps approach :

1. Compute the set $S = \{s_1, s_2, \ldots, s_n\}$ of candidates using Dynamic CSP.
2. Remove all patterns $s_i \in S$ that are not skypatterns.

While the number of candidates (n) could be very large, it remains reasonably-sized in practice (see [Ugarte *et al.*, 2014c]).

B.4.2 Mining the non-derivable Skypatterns

In order to find the non-derivable skypatterns, we proceed in the same way as Section B.4.1, by stating that any non-derivable skypattern could not de dominated by any derived skypattern.

Let $M_u \subseteq M$ and $derived(M_u)$ the subset of $Sky(M_u,)$ obtained using the two derivation rules. Consider the sequence $\Phi_1, \Phi_2, ..., \Phi_n$ of CSP where each $\Phi_i = (\{x\}, \mathcal{L}_\mathcal{I}, q_i(x))$ and :

— $q_1(x) = closed_{M'_u}(x) \wedge \Psi_{M_u}(x)$

— $q_{i+1}(x) = q_i(x) \wedge \neg(s_i \succ_{M_u} x)$ where s_i is the first solution to query $q_i(x)$

— $\Psi_{M_u}(x)$ states that x cannot be dominated w.r.t. M_u by any derived skypattern :

$$\Psi_{M_u}(x) = \bigwedge_{x_i \in derived(M_u)} \neg(x_i \succ_{M_u} x)$$

Exemple B.10. Consider example B.8. For M_u=$\{m_1, m_3\}$, $derived(M_u) = \{B, E, BCDE\}$. So $\Psi_{M_u}(x) = \neg(B \succ_{M_u} x) \wedge \neg(E \succ_{M_u} x) \wedge \neg(BCDE \succ_{M_u} x)$. The associated Dynamic CSP has a unique solution : $x = BCD$.

B.5 Related Work

Mining skypatterns is different from mining skylines [Börzsönyi *et al.*, 2001]. Skyline queries focus on the extraction of tuples of the dataset and assume that all skylines are in the dataset. The skypattern mining task consists in extracting patterns which are elements of the frontier defined by the given measures. The skypattern problem is clearly harder because the

search space for skypatterns is much larger than the search space for skylines : $O(2^{|\mathcal{I}|})$ instead of $O(|\mathcal{T}|)$ for skylines.

Two methods have been designed for mining skypatterns. `Aetheris` [Soulet *et al.*, 2011] takes benefit of theoretical relationships between pattern condensed representations and skypatterns. `Aetheris` proceeds in two steps : first, condensed representations of the whole set of patterns (i.e. closed patterns according to the considered set of measures) are extracted ; then, the sky operator (see Definition 5.3) is applied. `CP+SKY` [Ugarte *et al.*, 2014c] mines skypatterns using Dynamic CSP (see Section B.4.1). Both methods have the same efficiency, but `CP+SKY` also allows to mine soft skypatterns [Ugarte *et al.*, 2014c].

Computing the skyline cube efficiently. [Pei *et al.*, 2005, Pei *et al.*, 2006, Yuan *et al.*, 2005] proposed several strategies to share skyline computation in different subspaces, but they have to cope with the problem of enumerating skylines over all possible subspaces. [Pei *et al.*, 2007] proposed *Stellar*, which computes seed skylines groups in the full space, then extend them to build the final set of skyline groups and thus avoiding the computation of skylines in all the subspaces. But *Stellar* does not take profit from any parent-child relationships in the lattice. [Raïssi *et al.*, 2010] is able to decrease the number of domination tests by reducing the number of measure subspaces that needs to be searched. However, its complex strategy makes impossible the success of the full use of the parent-child relationships. Moreover, all of these techniques address skylines.

B.6 Experimental Evaluation

B.6.1 Skypattern Cubes for Mutagenicity Dataset

In this section, we report an experimental evaluation on a real-life dataset of large size extracted from mutagenicity data [Hansen *et al.*, 2009] (a major problem in risk assessment of chemicals). This dataset has $|\mathcal{T}|$=6,512 transactions encoding chemicals and $|\mathcal{I}|$=1,073 items [36] encoding frequent closed subgraphs previously extracted from $\mathcal{T}$ with a 2% relative frequency threshold. Chemists use up to $|M|$=11 measures, five of them are typically used in contrast mining (frequency and growth rate) and enable to express different kinds of background knowledge. The other six measures are related to topological, geometrical and chemical properties of the chemicals.

Experimental protocol. The implementation of `CP+SKY+CUBE` (bottom-up approach proposed in this paper) was carried out in `Gecode` by extending the CP-based pattern extractor developed by [Khiari *et al.*, 2010]. All experiments were conducted on a computer running Linux with a core i3 processor at 2.13 GHz.

(a) CPU-time analysis. We compare `CP+SKY+CUBE` with two other methods for computing a skypattern cube :

1. `Base-Line-Aetheris` applies `Aetheris` to the $2^{|M|}-1$ non empty subsets of M.
2. `Base-Line-CP+SKY` applies `CP+SKY` to the $2^{|M|}-1$ non empty subsets of M.

For the base-line methods, the CPU-time is the sum of CPU-times required for each non-empty subset of M. Fig. B.2 compares the performance of the three methods according to the number of measures $|M|$. The scale is logarithmic. For each method and for $|M|$=k, the reported

36. A chemical Ch contains an item A if Ch supports A, and A is a frequent subgraph of $\mathcal{T}$.

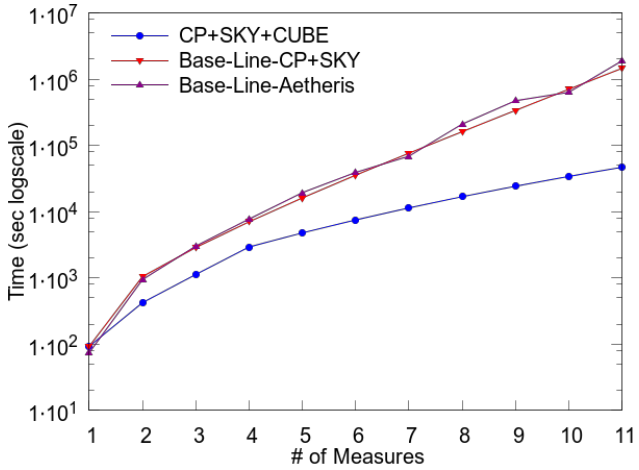


FIGURE B.2 – Comparing CPU-times for the 3 methods.

CPU-time is the average of CPU-times over all $\binom{11}{k}$ possible skypattern cubes. `Base-Line-Aetheris` and `Base-Line-CP+SKY` have a similar behavior since `Aetheris` and `CP+SKY` are equally effective (see Section B.5). `CP+SKY+CUBE` clearly outperforms the two base-line methods. For a small number of measures ($2 \leq |M| \leq 4$), the speed-up is 1.75 (see column 8, Table B.1). For $|M|=8$, there is an order of magnitude (speed-up value 9.57). For $|M|=11$, `CP+SKY+CUBE` requires about 13 hours to compute the skypattern cube, while the two baseline methods spent about 403 hours (speed-up value 31). Finally, if $|M|$ is increased by one (a new measure is added), the number of subsets to consider is doubled but we can see that our speed-up is multiplied by about 1.5 (see column 8, Table B.1).

(b) Space analysis. Column 1 (Table B.1) corresponds to the number of measures. Column 2 indicates the number of equivalence classes. Column 3 denotes the ratio between the number of equivalent classes and the total number of subsets of measure. Column 4 (resp. 5) reports the total number of skypatterns for the concise (resp. usual) representation (see Section B.3.3) and Column 6 gives their ratio. For $|M|=k$, reported values in columns (2), (4) and (5) represent the averages over all $\binom{11}{k}$ possible skypattern cubes. Our concise representation of the skypattern cube provides a substantial summarization and compression of skypattern sets. For instance, for $|M|=11$, there are 401 classes and a total number of $15,261$ skypatterns for the concise representation. For the usual representation, there are $2,047$ subsets of measure and and a total number of $87,374$ skypatterns. This leads to a substantial gain greater than 80%.

(c) Effectiveness of our derivation rules. In order to evaluate the effectiveness of the two derivation rules (cf. Section B.3.1), we measured the percentage of derived skypatterns (vs the total number of skypatterns) at each level of a cube. Reported values in Fig. B.3 are average values over all 11 possible cubes of 10 measures. For each level i ($2 \leq i \leq 10$), the proportion of incomparable and indistinct skypatterns is also depicted.

Our derivation rules are very efficient since they are able to deduce about $80-90\%$ of the skypatterns, except for the first levels. Moreover, when the number of measures increases, the number of indistinct skypatterns decreases (in percentage), while the number of incomparable skypatterns increases. Indeed, incomparable skypatterns of child nodes remain incomparable for a parent node (see Theorem B.1) whereas indistinct skypatterns may become any kind of

$\|M\|$	(2)	$\frac{(2)}{2^{\|M\|}-1}$	(4)	(5)	$\frac{(4)}{(5)}$	(7)	Speed-up
1	1	1.00	338	338	1.00	1m :33s	1.00
2	2	0.87	680	753	0.90	14m :39s	1.19
3	5	0.75	1,036	1,280	0.80	28m :12s	1.70
4	9	0.64	1,421	1,983	0.71	48m :43s	2.40
5	16	0.53	1,865	2,982	0.62	1h :19m :30s	3.37
6	28	0.44	2,424	4,526	0.53	2h :04m :45s	4.73
7	45	0.36	3,200	7,146	0.45	3h :09m :35s	6.69
8	73	0.29	4,386	12,015	0.36	4h :40m :03s	9.57
9	117	0.23	6,327	21,773	0.23	6h :43m :07s	13.93
10	213	0.20	9,619	42,386	0.23	9h :26m :42s	20.62
11	401	0.20	15,261	87,374	0.17	12h :59m :36s	30.97

(2) # of equivalence classes
(5) $\sum_{M_u \subseteq M} |\, Sky(M_u, |)$
(4) # of skypatterns for the concise representation
(7) CPU-Time for CP+SKY+CUBE

TABLE B.1 – Space analysis (left part) and CPU-time analysis (right part).

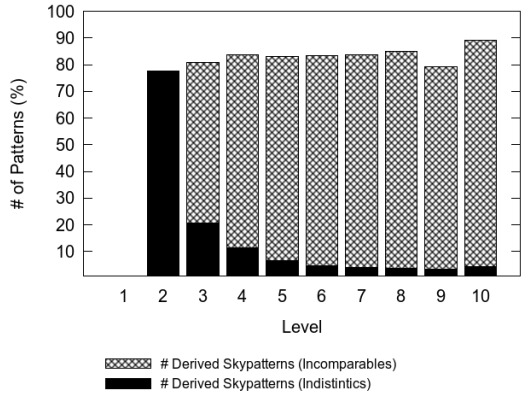


FIGURE B.3 – Effectiveness of our derivation rules.

skypatterns or dominated patterns (see Corollary 1).

(d) Effectiveness of our sufficient condition. Theorem B.3 gives our sufficient condition stating if, for a subset of measures M_u, $derived(M_u) = Sky(M_u,)$ without requiring any Dynamic CSP. To asses the effectiveness of this condition, we measured the percentage of success at each level of a cube. Reported values in Table B.2 are average values over all 11 possible cubes of 10 measures. Line 1 provides the level in the lattice. Line 2 indicates the number of nodes where our condition applies, while Line 3 reports the number of nodes where our condition should apply. Line 4 depicts their ratio (percentage of success). The more the number of measures increases, the more our sufficient condition becomes effective. From level 5 to level 10, the percentage of success increases from 73% to 100%.

Level	2	3	4	5	6	7	8	9	10
# of nodes where Th 3 applies	6.55	54.55	127.91	183.27	171.82	106.91	42.55	9.82	1.00
# of nodes where Th 3 should apply	29.45	102.55	194.73	252.00	210.00	120.00	45.00	10.00	1.00
Succ. ratio : (2)/(3)	0.22	0.53	0.66	0.73	0.82	0.89	0.95	0.98	1.00

TABLE B.2 – Effectiveness of our sufficient condition (Theorem B.3).

	CPU-Time			Speed-Up		
Dataset	(2)	(3)	(4)	$\frac{(2)}{(4)}$	$\frac{(3)}{(4)}$	(7)
austral	6m04s	4m15s	1m31s	3.98	2.79	0.82
cleve	1m53s	1m21s	21s	5.27	3.76	0.97
cmc	26s	2m23s	22s	1.20	6.41	0.90
crx	8m40s	5m37s	1m13s	7.12	4.61	0.89
german	2h34m18s	53m29s	14m03s	10.98	3.80	0.88
heart	1m46s	58s	19s	5.49	3.01	0.86
hepatic	6m12s	58s	19s	18.91	2.97	0.71
horse	10m34s	3m32s	58s	10.93	3.67	0.82
hypo	6h13m57s	51m46s	4m41s	79.75	11.04	0.79
lymph	4m32s	49s	11s	23.87	4.38	0.65
tic-tac-toe	1m10s	2m48s	41s	1.68	4.03	0.84
vehicle	34m01s	16m41s	2m55s	11.64	5.71	0.66
wine	1m00s	31s	13s	4.63	2.43	0.94
zoo	19s	8s	1s	10.43	4.82	0.87

(2) Base-line-Aetheris
(3) Base-line-CP+SKY
(4) CP+SKY+CUBE
(7) Succ. Ratio : $\frac{|derived(M_u)|}{|Sky(M_u,)|}$

TABLE B.3 – Results on UCI datasets with $|M|$=5.

B.6.2 Skypattern Cubes for UCI Datasets

Experiments were carried out on 14 various datasets from UCI [37] benchmarks. We considered 5 measures M=$\{freq, max, area, mean, growth\text{-}rate\}$. Measures using numeric values, like *mean*, were applied on attribute values that were randomly generated within the range [0..1]. Table B.3 summarizes the results we obtained.

(a) CPU-time analysis. Columns 2-4 compare the CPU-times of the three methods. `CP+SKY+CUBE` clearly dominates the two base-line methods. On half of the datasets, the speed-up is at least 10.43 (see column 5).

(b) Effectiveness of our derivation rules. Column 7 reports, for each dataset, the percentage of derived skypatterns. Reported values are the average values over levels i ($2 \leq i \leq 5$) of the cube. Our derivation rules are able to deduce about $79 - 97\%$ of the skypatterns, except for two datasets (lymph and vehicle).

37. `http://www.ics.uci.edu/~mlearn/MLRepository.html`

B.7 Conclusion

We have designed an efficient bottom-up method to compute skypattern cubes. Our derivation rules are able to collect a large part of the skypatterns of a parent node. Non-derivable skypatterns are computed on the fly thanks to Dynamic CSP. The bottom-up strategy makes easy the building of a concise representation of the cube according to skypattern equivalence classes. Experiments show the effectiveness of our proposal. Navigation through the cube is a highly promising perspective.

Annexe C

Prefix-Projection Global Constraint for Sequential Pattern Mining

Article [Kemmar *et al.*, 2015] : paru dans les actes du congrès CP en 2015

Auteurs : Amina Kemmar, Samir Loudni, Yahia Lebbah, Patrice Boizumault, Thierry Charnois

Abstract

Sequential pattern mining under constraints is a challenging data mining task. Many efficient ad hoc methods have been developed for mining sequential patterns, but they are all suffering from a lack of genericity. Recent works have investigated Constraint Programming (CP) methods, but they are not still effective because of their encoding. In this paper, we propose a global constraint based on the projected databases principle which remedies to this drawback. Experiments show that our approach clearly outperforms CP approaches and competes well with ad hoc methods on large datasets.

C.1 Introduction

Mining useful patterns in sequential data is a challenging task. Sequential pattern mining is among the most important and popular data mining task with many real applications such as the analysis of web click-streams, medical or biological data and textual data. For effectiveness and efficiency considerations, many authors have promoted the use of constraints to focus on the most promising patterns according to the interests given by the final user. In line with [Pei *et al.*, 2002], many efficient ad hoc methods have been developed but they suffer from a lack of genericity to handle and to push simultaneously sophisticated combination of various types of constraints. Indeed, new constraints have to be hand-coded and their combinations often require new implementations.

Recently, several proposals have investigated relationships between sequential pattern mining and constraint programming (CP) to revisit data mining tasks in a declarative and generic way [Coquery *et al.*, 2012, Métivier *et al.*, 2013, Kemmar *et al.*, 2014, Négrevergne et Guns, 2015]. The great advantage of these approaches is their flexibility. The user can model a problem and express his queries by specifying what constraints need to be satisfied. But, all these proposals

are not effective enough because of their CP encoding. Consequently, the design of new efficient declarative models for mining useful patterns in sequential data is clearly an important challenge for CP.

To address this challenge, we investigate in this paper the other side of the cross fertilization between data-mining and constraint programming, namely how the CP framework can benefit from the power of candidate pruning mechanisms used in sequential pattern mining. First, we introduce the global constraint PREFIX-PROJECTION for sequential pattern mining. PREFIX-PROJECTION uses a concise encoding and its filtering relies on the principle of projected databases [Pei *et al.*, 2001]. The key idea is to divide the initial database into smaller ones projected on the frequent subsequences obtained so far, then, mine locally frequent patterns in each projected database by growing a frequent prefix. This global constraint utilizes the principle of prefix-projected database to keep only locally frequent items alongside projected databases in order to remove infrequent ones from the domains of variables. Second, we show how the concise encoding allows for a straightforward implementation of the frequency constraint (PREFIX-PROJECTION constraint) and constraints on patterns such as size, item membership and regular expressions and the simultaneous combination of them. Finally, experiments show that our approach clearly outperforms CP approaches and competes well with ad hoc methods on large datasets for mining frequent sequential patterns or patterns under various constraints. It is worth noting that the experiments show that our approach achieves scalability while it is a major issue of CP approaches.

The paper is organized as follows. Section C.2 recalls preliminaries. Section C.3 provides a critical review of ad hoc methods and CP approaches for sequential pattern mining. Section C.4 presents the global constraint PREFIX-PROJECTION. Section C.5 reports experiments we performed. Finally, we conclude and draw some perspectives.

C.2 Preliminaries

This section presents background knowledge about sequential pattern mining and constraint satisfaction problems.

C.2.1 Sequential Patterns

Let $\mathcal{I}$ be a finite set of *items*. The language of sequences corresponds to $\mathcal{L}_\mathcal{I} = \mathcal{I}^n$ where $n \in \mathbb{N}^*$.

Définition C.17 (SEQUENCE, SEQUENCE DATABASE)
A sequence s over $\mathcal{L}_\mathcal{I}$ is an ordered list $\langle s_1 s_2 \dots s_n \rangle$, where s_i, $1 \leq i \leq n$, is an item. n is called the length of the sequence s. A sequence database SDB is a set of tuples (sid, s), where sid is a sequence identifier and s a sequence.

Définition C.18 (SUBSEQUENCE, $\preceq$ RELATION)
A sequence $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ is a subsequence of $s = \langle s_1 \dots s_n \rangle$, denoted by $(\alpha \preceq s)$, if $m \leq n$ and there exist integers $1 \leq j_1 \leq \dots \leq j_m \leq n$, such that $\alpha_i = s_{j_i}$ for all $1 \leq i \leq m$. We also say that α is contained in s or s is a super-sequence of α. For example, the sequence $\langle BABC \rangle$ is a super-sequence of $\langle AC \rangle$: $\langle AC \rangle \preceq \langle BABC \rangle$. A tuple (sid, s) contains a sequence α, if $\alpha \preceq s$.

sid	Sequence
1	$\langle ABCBC \rangle$
2	$\langle BABC \rangle$
3	$\langle AB \rangle$
4	$\langle BCD \rangle$

TABLE C.1 – SDB_1 : a sequence database example.

The cover of a sequence p in SDB is the set of all tuples in SDB in which p is contained. The support of a sequence p in SDB is the number of tuples in SDB which contain p.

Définition C.19 (COVERAGE, SUPPORT)
Let SDB be a sequence database and p a sequence. $\texttt{cover}_{SDB}(p)=\{(sid, s) \in SDB \mid p \preceq s\}$ and $\texttt{freq}_{SDB}(p) = \#\texttt{cover}_{SDB}(p)$.

Définition C.20 (SEQUENTIAL PATTERN)
Given a minimum support threshold $minfr$, every sequence p such that $\texttt{freq}_{SDB}(p) \geq minfr$ is called a sequential pattern [Agrawal et Srikant, 1995]. p is said to be frequent in SDB.

Exemple C.11. Table C.1 represents a sequence database of four sequences where the set of items is $\mathcal{I} = \{A, B, C, D\}$. Let the sequence $p = \langle AC \rangle$. We have $\texttt{cover}_{SDB1}(p) = \{(1, s_1), (2, s_2)\}$. If we consider $minfr = 2$, $p = \langle AC \rangle$ is a sequential pattern because $\texttt{freq}_{SDB_1}(p) \geq 2$.

Définition C.21 (SEQUENTIAL PATTERN MINING (SPM))
Given a sequence database SDB and a minimum support threshold $minfr$. The problem of sequential pattern mining is to find all patterns p such that $\texttt{freq}_{SDB}(p) \geq minfr$.

C.2.2 SPM under Constraints

In this section, we define the problem of mining sequential patterns in a sequence database satisfying user-defined constraints Then, we review the most usual constraints for the sequential mining problem [Pei *et al.*, 2002].

Problem statement. Given a constraint $C(p)$ on pattern p and a sequence database SDB, the problem of constraint-based pattern mining is to find the complete set of patterns satisfying $C(p)$. In the following, we present different types of constraints that we explicit in the context of sequence mining.

- The minimum size constraint $size(p, \ell_{min})$ states that the number of items of p must be greater than or equal to ℓ_{min}.

- The item constraint $item(p, t)$ states that an item t must belong (or not) to a pattern p.

- The regular expression constraint [Garofalakis *et al.*, 2002] $reg(p, exp)$ states that a pattern p must be accepted by the deterministic finite automata associated to the regular expression exp.

C.2.3 Projected Databases

We now present the definitions related to the concept of *projected databases* [Pei *et al.*, 2001].

Définition C.22 (PREFIX, PROJECTION, SUFFIX)
Let $\beta = \langle \beta_1 \dots \beta_n \rangle$ and $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ be two sequences, where $m \leq n$.
- Sequence α is called the prefix of β iff $\forall i \in [1..m], \alpha_i = \beta_i$.
- Sequence $\beta = \langle \beta_1 \dots \beta_n \rangle$ is called the projection of some sequence s w.r.t. α, iff (1) $\beta \preceq s$, (2) α is a prefix of β and (3) there exists no proper super-sequence β' of β such that $\beta' \preceq s$ and β' also has α as prefix.
- Sequence $\gamma = \langle \beta_{m+1} \dots \beta_n \rangle$ is called the suffix of s w.r.t. α. With the standard concatenation operator "*concat*", we have $\beta = concat(\alpha, \gamma)$.

Définition C.23 (PROJECTED DATABASE)
Let SDB be a sequence database, the α-projected database, denoted by $SDB|_\alpha$, is the collection of suffixes of sequences in SDB w.r.t. prefix α.

[Pei *et al.*, 2001] have proposed an efficient algorithm, called `PrefixSpan`, for mining sequential patterns based on the concept of *projected databases*. It proceeds by dividing the initial database into smaller ones projected on the frequent subsequences obtained so far ; only their corresponding suffixes are kept. Then, sequential patterns are mined in each projected database by exploring only locally frequent patterns.

Proposition C.1 (SUPPORT COUNT)
For any sequence γ in SDB with prefix α and suffix β s.t. $\gamma = concat(\alpha, \beta)$, $\texttt{freq}_{SDB}(\gamma) = \texttt{freq}_{SDB|_\alpha}(\beta)$.

Proposition C.1 establishes the support count of a sequence γ in $SDB|_\alpha$ [Pei *et al.*, 2001]. It ensures that only the sequences in SDB grown from α need to be considered for the support count of a sequence γ. Furthermore, only those suffixes with prefix α should be counted.

C.2.4 CSP and Global Constraints

A *Constraint Satisfaction Problem* (CSP) consists of a set X of n variables, a domain $\mathcal{D}$ mapping each variable $X_i \in X$ to a finite set of values $D(X_i)$, and a set of constraints $\mathcal{C}$. An assignment σ is a mapping from variables in X to values in their domains : $\forall X_i \in X, \sigma(X_i) \in D(X_i)$. A constraint $c \in \mathcal{C}$ is a subset of the cartesian product of the domains of the variables that are in c. The goal is to find an assignment such that all constraints are satisfied.

Domain consistency (DC). Constraint solvers typically use backtracking search to explore the space of partial assignments. At each assignment, filtering algorithms prune the search space by enforcing local consistency properties like domain consistency. A constraint c on X is domain consistent, if and only if, for every $X_i \in X$ and for every $d_i \in D(X_i)$, there is an assignment σ satisfying c such that $\sigma(X_i) = d_i$. Such an assignment is called a support.

Global constraints provide shorthands to often-used combinatorial substructures. We present two global constraints. Let $X = \langle X_1, X_2, ..., X_n \rangle$ be a sequence of n variables. Let V be a set of values, l and u be two integers s.t. $0 \leq l \leq u \leq n$, the constraint $\texttt{Among}(X, V, l, u)$ states that each value $a \in V$ should occur at least l times and at most u times in X [Beldiceanu et Contejean, 1994b]. Given a deterministic finite automaton A, the constraint $\texttt{Regular}(X, A)$ ensures that the sequence X is accepted by A [Pesant, 2004b].

C.3 Related works

This section provides a critical review of ad hoc methods and CP approaches for SPM.

C.3.1 Ad hoc Methods for SPM

`GSP` [Srikant et Agrawal, 1996] was the first algorithm proposed to extract sequential patterns. It uses a generate-and test approach. Later, two major classes of methods have been proposed :
- Depth-first search based on a vertical database format e.g. `cSpade` incorporating contraints (max-gap, max-span, length) [Zaki, 2000], `SPADE` [Zaki, 2001] or `SPAM` [Ayres *et al.*, 2002].
- Projected pattern growth such as `PrefixSpan` [Pei *et al.*, 2001] and its extensions, e.g. `CloSpan` for mining closed sequential patterns [Yan *et al.*, 2003] or `Gap-BIDE` [Li *et al.*, 2012] tackling the gap constraint.

In [Garofalakis *et al.*, 2002], the authors proposed `SPIRIT` based on `GSP` for SPM with regular expressions. Later, [Trasarti *et al.*, 2008] introduces Sequence Mining Automata (`SMA`), a new approach based on a specialized kind of Petri Net. Two variants of `SMA` were proposed : `SMA-1P` (`SMA` one pass) and `SMA-FC` (`SMA` Full Check). `SMA-1P` processes by means of the `SMA` all sequences one by one, and enters all resulting valid patterns in a hash table for support counting, while `SMA-FC` allows frequency based pruning during the scan of the database. Finally, [Pei *et al.*, 2002] provides a survey for other constraints such as regular expressions, length and aggregates. But, all these proposals, though efficient, are ad hoc methods suffering from a lack of genericity. Adding new constraints often requires to develop new implementations.

C.3.2 CP Methods for SPM

Following the work of [Guns *et al.*, 2011] for itemset mining, several methods have been proposed to mine sequential patterns using CP.

Proposals. [Coquery *et al.*, 2012] have proposed a first SAT-based model for discovering a special class of patterns with wildcards [38] in a single sequence under different types of constraints (e.g. frequency, maximality, closedness). [Métivier *et al.*, 2013] have proposed a CSP model for SPM. Each sequence is encoded by an automaton capturing all subsequences that can occur in it. [Kemmar *et al.*, 2014] have proposed a CSP model for SPM with wildcards. They show how some constraints dealing with local patterns (e.g. frequency, size, gap, regular expressions) and constraints defining more complex patterns such as relevant subgroups [Novak *et al.*, 2009] and top-k patterns can be modeled using a CSP. [Négrevergne et Guns, 2015] have proposed two CP encodings for the SPM. The first one uses a global constraint to encode the subsequence relation (denoted `global-p.f`), while the second one encodes explicitly this relation using additional variables and constraints (denoted `decomposed-p.f`).

All these proposals use **reified constraints** to encode the database. For each sequence s of SDB, a reified constraint, stating whether (or not) the unknown pattern p is a subsequence of s, is imposed : $(S_s = 1) \Leftrightarrow (p \preceq s)$. A great consequence is that the encoding of the frequency measure is straightforward : $\texttt{freq}_{SDB}(p) = \sum_{s \in SDB} S_s$. But such an encoding has a major drawback since it requires $(m = \#SDB)$ reified constraints to encode the whole database. This constitutes a strong limitation of the size of the databases that could be managed.

38. A wildcard is a special symbol that matches any item of $\mathcal{I}$ including itself.

Most of these proposals encode **the subsequence relation** ($p \preceq s$) using variables $Pos_{s,j}$ ($s \in SDB$ and $1 \leq j \leq \ell$) to determine a position where p occurs in s. Such an encoding requires a large number of additional variables ($m \times \ell$) and makes the labeling computationally expensive. In order to address this drawback, [Négrevergne et Guns, 2015] have proposed a global constraint `exists-embedding` to encode the subsequence relation, and used projected frequency within an ad hoc specific branching strategy to keep only frequent items before branching over the variables of the pattern. But, this encoding still relies on reified constraints and requires to impose m `exists-embedding` global constraints.

So, we propose in the next section the PREFIX-PROJECTION global constraint that fully exploits the principle of projected databases to encode both the subsequence relation and the frequency constraint. PREFIX-PROJECTION does not require any reified constraints nor any extra variables to encode the subsequence relation. As a consequence, usual SPM constraints (see Section C.2.2) can be encoded in a straightforward way using directly the (global) constraints of the CP solver.

C.4 PREFIX-PROJECTION Global Constraint

C.4.1 A Concise Encoding

Let P be the unknown pattern of size ℓ we are looking for. The symbol $\square$ stands for an empty item and denotes the end of a sequence. The unknown pattern P is encoded with a sequence of ℓ variables $\langle P_1, P_2, \ldots, P_\ell \rangle$ s.t. $\forall i \in [1 \ldots \ell], D(P_i) = \mathcal{I} \cup \{\square\}$. There are two basic rules on the domains :

1. To avoid the empty sequence, the first item of P must be non empty, so ($\square \notin D_1$).
2. To allow patterns with less than ℓ items, we impose that $\forall i \in [1..(\ell-1)], (P_i = \square) \rightarrow (P_{i+1} = \square)$.

C.4.2 Definition and Consistency Checking

The global constraint PREFIX-PROJECTION ensures both subsequence relation and minimum frequency constraint.

Définition C.24 (PREFIX-PROJECTION GLOBAL CONSTRAINT)
Let $P = \langle P_1, P_2, \ldots, P_\ell \rangle$ be a pattern of size ℓ and $minfr$ a minimum support.

$$\text{PREFIX-PROJECTION}(P, SDB, minfr) = \{\langle d_1, ..., d_\ell \rangle \mid \forall\, i\, d_i \in D(P_i) : \texttt{freq}_{SDB}(\langle d_1, ..., d_\ell \rangle) \geq minfr\}$$

Proposition C.2 (CONSISTENCY CHECKING OF PREFIX-PROJECTION)
A PREFIX-PROJECTION $(P, SDB, minfr)$ constraint has a solution if and only if there exists an assignment $\sigma = \langle d_1, ..., d_\ell \rangle$ of variables of P s.t. $SDB|_\sigma$ has at least $minfr$ suffixes of σ : $\#SDB|_\sigma \geq minfr$.

Proof : This is a direct consequence of proposition C.1. We have straightforwardly $\texttt{freq}_{SDB}(\sigma) = \texttt{freq}_{SDB|_\sigma}(\langle\rangle) = \#SDB|_\sigma$. Thus, suffixes of $SDB|_\sigma$ are supports of σ in the constraint PREFIX-PROJECTION $(P, SDB, minfr)$, provided that $\#SDB|_\sigma \geq minfr$. $\square$

The following proposition characterizes values in the domain of unassigned (i.e. future) variable P_{i+1} that are consistent with the current assignment of variables $\langle P_1, ..., P_i \rangle$.

Proposition C.3 (Consistent values for Prefix-Projection $(P, SDB, minfr)$)
Let σ [39] $= \langle d_1, \ldots, d_i \rangle$ be a current assignment of variables $\langle P_1, \ldots, P_i \rangle$, P_{i+1} be a future variable. A value $d \in D(P_{i+1})$ appears in a solution for Prefix-Projection $(P, SDB, minfr)$ if and only if d is a frequent item in $SDB|_\sigma$:

$$\#\{(sid, \gamma) | (sid, \gamma) \in SDB|_\sigma \wedge \langle d \rangle \preceq \gamma\} \geq minfr$$

Proof : Suppose that value $d \in D(P_{i+1})$ occurs in $SDB|_\sigma$ more than $minfr$. From proposition C.1, we have $\texttt{freq}_{SDB}(concat(\sigma, \langle d \rangle)) = \texttt{freq}_{SDB|_\sigma}(\langle d \rangle)$. Hence, the assignment $\sigma \cup \langle d \rangle$ satisfies the constraint, so $d \in D(P_{i+1})$ participates in a solution. □

From proposition C.3 and according to the *anti-monotonicity property*, we can derive the following pruning rule :

Proposition C.4 (Filtering rules)
Let $\sigma = \langle d_1, \ldots, d_i \rangle$ be a current assignment of variables $\langle P_1, \ldots, P_i \rangle$. All values $d \in D(P_{i+1})$ that are locally not frequent in $SDB|_\sigma$ can be pruned from the domain of variable P_{i+1}. Moreover, these values d can also be pruned from the domains of variables P_j with $j \in [i+2, \ldots, \ell]$.

Proof : Let $\sigma = \langle d_1, \ldots, d_i \rangle$ be a current assignment of variables $\langle P_1, \ldots, P_i \rangle$. Let $d \in D(P_{i+1})$ s.t. $\sigma' = concat(\sigma, \langle d \rangle)$. Suppose that d is not frequent in $SDB|_\sigma$. According to proposition C.1, $\texttt{freq}_{SDB|_\sigma}(\langle d \rangle) = \texttt{freq}_{SDB}(\sigma') < minfr$, thus σ' is not frequent. So, d can be pruned from the domain of P_{i+1}.
Suppose that the assignment σ has been extended to $concat(\sigma, \alpha)$, where α corresponds to the assignment of variables P_j (with $j > i$). If $d \in D(P_{i+1})$ is not frequent, it is straightforward that $\texttt{freq}_{SDB|_\sigma}(concat(\alpha, \langle d \rangle)) \leq \texttt{freq}_{SDB|_\sigma}(\langle d \rangle) < minfr$. Thus, if d is not frequent in $SDB|_\sigma$, it will be also not frequent in $SDB|_{concat(\sigma,\alpha)}$. So, d can be pruned from the domains of P_j with $j \in [i+2, \ldots, \ell]$. □

Exemple C.12. Consider the sequence database of Table C.1 with $minfr = 2$. Let $P = \langle P_1, P_2, P_3 \rangle$ with $D(P_1) = \mathcal{I}$ and $D(P_2) = D(P_3) = \mathcal{I} \cup \{\square\}$. Suppose that $\sigma(P_1) = A$, Prefix-Projection$(P, SDB, minfr)$ will remove values A and D from $D(P_2)$ and $D(P_3)$, since the only locally frequent items in $SDB_1|_{<A>}$ are B and C.

Proposition C.4 guarantees that any value (i.e. item) $d \in D(P_{i+1})$ present but not frequent in $SDB|_\sigma$ does not need to be considered when extending σ, thus avoiding searching over it. Clearly, our global constraint encodes the anti-monotonicity of the frequency measure in a simple and elegant way, while CP methods for SPM have difficulties to handle this property. In [Négrevergne et Guns, 2015], this is achieved by using very specific propagators and branching strategies, making the integration quite complex (see [Négrevergne et Guns, 2015]).

C.4.3 Building the projected databases.

The key issue of our approach lies in the construction of the projected databases. When projecting a prefix, instead of storing the whole suffix as a projected subsequence, one can represent each suffix by a pair $(sid, start)$ where sid is the sequence identifier and $start$ is the

39. We indifferently denote σ by $\langle d_1, \ldots, d_i \rangle$ or by $\langle \sigma(P_1), \ldots, \sigma(P_i) \rangle$.

Algorithme 10 : *ProjectSDB* (SDB, $ProjSDB$, α)

```
   Data : SDB : initial database ; ProjSDB : projected sequences ; α : prefix
 1 begin
 2     SDB|α ← ∅ ;
 3     for each pair (sid, start) ∈ ProjSDB do
 4         s ← SDB[sid] ;
 5         pos_α ← 1 ; pos_s ← start ;
 6         while (pos_α ≤ |α| ∧ pos_s ≤ |s|) do
 7             if (α[pos_α] = s[pos_s]) then
 8                 pos_α ← pos_α + 1 ;
 9             pos_s ← pos_s + 1 ;
10         if (pos_α = |α| + 1) then
11             SDB|α ← SDB|α ∪ {(sid, pos_s)}
12     return SDB|α ;
13 end
```

starting position of the projected suffix in the sequence *sid*. For instance, let us consider the sequence database of Table C.1. $SDB|_{\langle A \rangle}$ consists of 3 suffix sequences : $\{(1, \langle BCBC \rangle), (2, \langle BC \rangle), (3, \langle B \rangle)\}$. By using the *pseudo-projection*, $SDB|_{\langle A \rangle}$ can be represented by the following three pairs : $\{(1,2), (2,3), (3,2)\}$. This is the principle of *pseudo-projection*, adopted in `PrefixSpan`, exploited during the filtering step of our PREFIX-PROJECTION global constraint. Algorithm 10 details this principle. It takes as input a set of projected sequences $ProjSDB$ and a prefix α. The algorithm processes all the pairs $(sid, start)$ of $ProjSDB$ one by one (line 3), and searches for the lowest location of α in the sequence s corresponding to the *sid* of that sequence in SDB (lines 7-9).

In the worst case, *ProjectSDB* processes all the items of all sequences. So, the time complexity is $O(\ell \times m)$, with $m = \#SDB$ and ℓ is the length of the longest sequence in SDB. The worst case space complexity of pseudo-projection is $O(m)$, since we need to store for each sequence only a pair $(sid, start)$, while for the standard projection the space complexity is $O(m \times \ell)$. Clearly, the pseudo-projection takes much less space than the standard projection.

C.4.4 Filtering

Ensuring DC on PREFIX-PROJECTION$(P, SDB, minfr)$ is equivalent to finding a sequential pattern of length $(\ell - 1)$ and then checking whether this pattern remains a frequent pattern when extended to any item d_ℓ in $D(P_\ell)$. Thus, finding such an assignment (i.e. support) is as much as difficult than the original problem of sequential pattern mining. [Yang, 2006] has proved that the problem of counting the number of maximal [40] frequent patterns in a database of sequences is #P-complete, thereby proving the NP-hardness of the problem of mining maximal frequent sequences. The difficulty is due to the exponential number of candidates that should be parsed to find the frequent patterns. Thus, finding, for every variable $P_i \in P$ and for every $d_i \in D(P_i)$, an assignment σ satisfying PREFIX-PROJECTION$(P, SDB, minfr)$ s.t. $\sigma(P_i) = d_i$ is of exponential nature.

So, the filtering of the PREFIX-PROJECTION constraint maintains a consistency lower than DC. This consistency is based on specific properties of the projected databases (see Propo-

40. A sequential pattern p is maximal if there is no sequential pattern q such that $p \preceq q$.

Algorithme 11 : *Filter-Prefix-Projection* (SDB, σ, i, P, $minfr$)

```
   Data : SDB : initial database ; σ : current prefix ⟨σ(P_1), ..., σ(P_i)⟩ ; minfr : the minimum support threshold ;
          PSDB : internal data structure of Prefix-Projection for storing pseudo-projected databases
 1 begin
 2     if (i ≥ 2 ∧ σ(P_i) = □) then
 3         for j ← i + 1 to ℓ do
 4             P_j ← □;
 5         return True;
 6     else
 7         PSDB_i ← ProjectSDB(SDB, PSDB_{i−1}, ⟨σ(P_i)⟩);
 8         if (||PSDB_i < minfr) then
 9             return False ;
10         else
11             FI ← getFreqItems(SDB, PSDB_i, minfr) ;
12             for j ← i + 1 to ℓ do
13                 foreach a ∈ D(P_j) s.t.(a ≠ □ ∧ a ∉ FI) do
14                     D(P_j) ← D(P_j) − {a};
15             return True;
16 end
17 Function getFreqItems (SDB, ProjSDB, minfr) ;
   Data : SDB : the initial database ; ProjSDB : pseudo-projected database ; minfr : the minimum support
          threshold ; ExistsItem, SupCount : internal data structures using a hash table for support counting over
          items ;
18 begin
19     SupCount[] ← {0, ..., 0} ;  F ← ∅ ;
20     for each pair (sid, start) ∈ ProjSDB do
21         ExistsItem[] ← {false, ..., false} ; s ← SDB[sid] ;
22         for i ← start to |s| do
23             a ← s[i] ;
24             if (¬ExistsItem[a]) then
25                 SupCount[a] ← SupCount[a] + 1 ; ExistsItem[a] ← true;
26                 if (SupCount[a] ≥ minfr) then
27                     F ← F ∪ {a};
28     return F;
29 end
```

sition C.3), and anti-monotonicity of the frequency constraint (see Proposition C.4), and resembles forward-checking regarding Proposition C.3. Prefix-Projection is considered as a global constraint, since all variables share the same internal data structures that awake and drive the filtering.

Algorithm 11 describes the pseudo-code of the filtering algorithm of the Prefix-Projection constraint. It is an incremental filtering algorithm that should be run when some i first variables are assigned according to the following lexicographic ordering $\langle P_1, P_2, \ldots, P_\ell \rangle$ of variables of P. It exploits internal data-structures enabling to enhance the filtering algorithm. More precisely, it uses an incremental data structure, denoted $\mathcal{PSDB}$, that stores the intermediate pseudo-projections of SDB, where $\mathcal{PSDB}_i$ ($i \in [0, \ldots, \ell]$) corresponds to the σ-projected database of the current partial assignment $\sigma = \langle \sigma(P_1), \ldots, \sigma(P_i) \rangle$ (also called prefix) of variables $\langle P_1, \ldots, P_i \rangle$, and $\mathcal{PSDB}_0 = \{(sid, 1) | (sid, s) \in SDB\}$ is the initial pseudo-projected database of SDB (case where $\sigma = \langle\rangle$). It also uses a hash table indexing the items $\mathcal{I}$ into integers ($1 \ldots |\mathcal{I}|$) for an efficient support counting over items (see function `getFreqItems`).

Algorithm 11 takes as input the current partial assignment $\sigma = \langle \sigma(P_1), \ldots, \sigma(P_i) \rangle$ of variables

$\langle P_1, \ldots, P_i \rangle$, the length i of σ (i.e. position of the last assigned variable in P) and the minimum support threshold $minfr$. It starts by checking if the last assigned variable P_i is instantiated to $\square$ (line 2). In this case, the end of sequence is reached (since value $\square$ can only appear at the end) and the sequence $\langle \sigma(P_1), \ldots, \sigma(P_i) \rangle$ constitutes a frequent pattern in SDB ; hence the algorithm sets the remaining $(\ell - i)$ unassigned variables to $\square$ and returns *true* (lines 3-5). Otherwise, the algorithm computes incrementally $\mathcal{PSDB}_i$ from $\mathcal{PSDB}_{i-1}$ by calling function *ProjectSDB* (see Algorithm 10). Then, it checks in line 8 whether the current assignment σ is a *legal* prefix for the constraint (see Proposition C.2). This is done by computing the size of $\mathcal{PSDB}_i$. If this size is less than $minfr$, we stop growing σ and we return *false*. Otherwise, the algorithm computes the set of locally frequent items $\mathcal{F}_{\mathcal{I}}$ in $\mathcal{PSDB}_i$ by calling function `getFreqItems` (line 11).

Function `getFreqItems` processes all the entries of the pseudo-projected database one by one, counts the number of first occurrences of items a (i.e. $SupCount[a]$) in each entry $(sid, start)$, and keeps only the frequent ones (lines 19-27). This is done by using $ExistsItem$ data structure. After the whole pseudo-projected database has been processed, the frequent items are returned (line 28), and Algorithm 11 updates the current domains of variables P_j with $j \geq (i+1)$ by pruning inconsistent values, thus avoiding searching over not frequent items (lines 12-14).

Proposition C.5 (TIME AND SPACE COMPLEXITIES OF THE FILTERING)
In the worst case, filtering with PREFIX-PROJECTION global constraint can be achieved in $O(m \times \ell + m \times d + \ell \times d)$. The worst case space complexity of PREFIX-PROJECTION is $O(m \times \ell)$.

Proof : Let ℓ be the length of the longest sequence in SDB, $m = |SDB|$, and $d = |\mathcal{I}|$. Computing the pseudo-projected database $\mathcal{PSDB}_i$ can be done in $O(m \times \ell)$: for each sequence (sid, s) of SDB, checking if σ occurs in s is $O(\ell)$ and there are m sequences. The total complexity of function *GetFreqItems* is $O(m \times (\ell + d))$. Lines (12-14) can be achieved in $O(\ell \times d)$. So, the whole complexity is $O(m \times \ell + m \times (\ell + d) + \ell \times d) = O(m \times \ell + m \times d + \ell \times d)$. The space complexity of the filtering algorithm lies in the storage of the $\mathcal{PSDB}$ internal data structure. In the worst case, we have to store ℓ pseudo-projected databases. Since each pseudo-projected database requires $O(m)$, the worst case space complexity is $O(m \times \ell)$. $\square$

C.4.5 Encoding of SPM Constraints

Usual SPM constraints (see Section C.2.2) can be reformulated in a straightforward way. Let P be the unknown pattern.

- *Minimum size constraint* : $size(P, \ell_{min}) \equiv \bigwedge_{i=1}^{i=\ell_{min}} (P_i \neq \square)$
- *Item constraint* : let V be a subset of items, l and u two integers s.t. $0 \leq l \leq u \leq \ell$. $item(P, V) \equiv \bigwedge_{t \in V} \texttt{Among}(P, \{t\}, l, u)$ enforces that items of V should occur at least l times and at most u times in P. To forbid items of V to occur in P, l and u must be set to 0.
- *Regular expression constraint* : let A_{reg} be the deterministic finite automaton encoding the regular expression *exp*. $reg(P, exp) \equiv \texttt{Regular}(P, A_{reg})$.

C.5 Experimental Evaluation

This section reports experiments on several real-life datasets from [Fournier-Viger *et al.*, 2014, Béchet *et al.*, 2012, Trasarti *et al.*, 2008] of large size having varied characteristics and representing different application domains (see Table C.2). Our objective is (1) to compare our approach

dataset	$\#SDB$	$\#\mathcal{I}$	avg ($\#s$)	$\texttt{Max}_{s \in SDB}$ ($\#s$)	type of data
Leviathen	5834	9025	33.81	100	book
Kosarak	69999	21144	7.97	796	web click stream
FIFA	20450	2990	34.74	100	web click stream
BIBLE	36369	13905	21.64	100	bible
Protein	103120	24	482	600	protein sequences
data-200K	200000	20	50	86	synthetic dataset
PubMed	17527	19931	29	198	bio-medical text

TABLE C.2 – Dataset Characteristics.

to existing CP methods as well as to state-of-the-art methods for SPM in terms of scalability which is a major issue of existing CP methods, (2) to show the flexibility of our approach allowing to handle different constraints simultaneously.

C.5.1 Experimental protocol

The implementation of our approach was carried out in the `Gecode` solver [41]. All experiments were conducted on a machine with a processor Intel X5670 and 24 GB of memory. A time limit of 1 hour has been used. For each dataset, we varied the $minfr$ threshold until the methods are not able to complete the extraction of all patterns within the time limit. ℓ was set to the length of the longest sequence of SDB. The implementation and the datasets used in our experiments are available online [42]. We compare our approach (indicated by `PP`) with :

1. two CP encodings [Négrevergne et Guns, 2015], the most efficient CP methods for SPM : `global-p.f` and `decomposed-p.f` ;
2. state-of-the-art methods for SPM : `PrefixSpan` and `cSpade` ;
3. `SMA` [Trasarti *et al.*, 2008] for SPM under regular expressions.

We used the author's `cSpade` implementation [43] for SPM, the publicly available implementations of `PrefixSpan` by Y. Tabei [44] and the `SMA` implementation [45] for SPM under regular expressions. The implementation [46] of the two CP encodings was carried out in the `Gecode` solver. All methods have been executed on the same machine.

C.5.2 Comparing with CP Methods for SPM

First we compare `PP` with the two CP encodings `global-p.f` and `decomposed-p.f` (see Section C.3.2). CPU times (in logscale for BIBLE, Kosarak and PubMed) of the three methods are shown on Fig. C.1. First, `decomposed-p.f` is the least performer method. On all the datasets, it fails to complete the extraction within the time limit for all values of $minfr$ we considered.

41. `http://www.gecode.org`
42. `https://sites.google.com/site/prefixprojection4cp/`
43. `http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/`
44. `https://code.google.com/p/prefixspan/`
45. `http://www-kdd.isti.cnr.it/SMA/`
46. `https://dtai.cs.kuleuven.be/CP4IM/cpsm/`

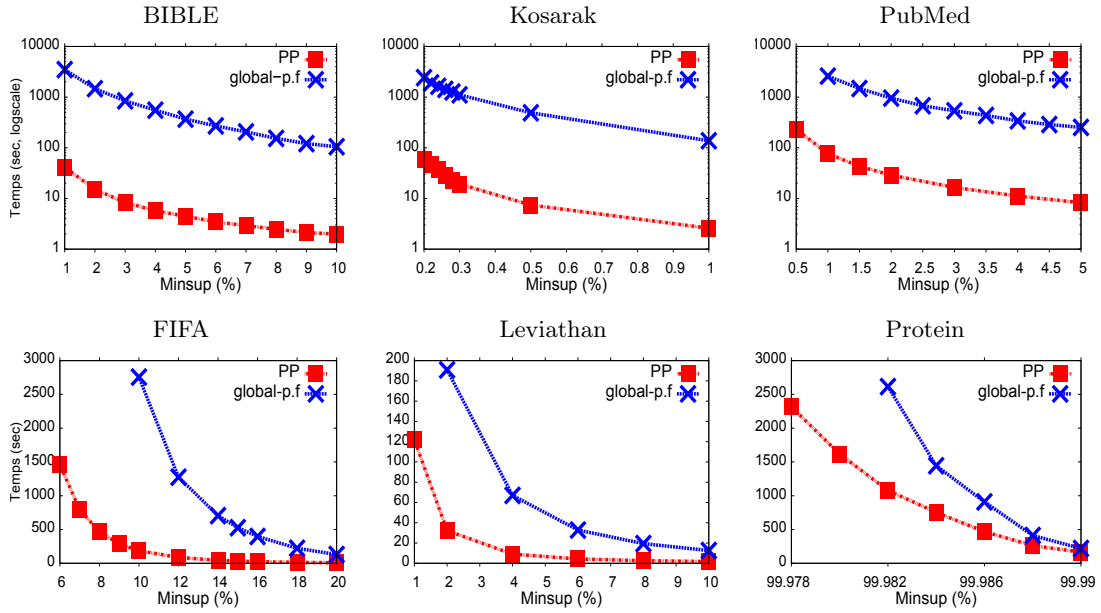


FIGURE C.1 – Comparing `PP` with `global-p.f` for SPM on real-life datasets : CPU times.

Second, `PP` largely dominates `global-p.f` on all the datasets : `PP` is more than an order of magnitude faster than `global-p.f`. The gains in terms of CPU times are greatly amplified for low values of $minfr$. On BIBLE (resp. PubMed), the speed-up is 84.4 (resp. 33.5) for $minfr$ equal to 1%. Another important observation that can be made is that, on most of the datasets (except BIBLE and Kosarak), `global-p.f` is not able to mine for patterns at very low frequency within the time limit. For example on FIFA, `PP` is able to complete the extraction for values of $minfr$ up to 6% in $1,457$ seconds, while `global-p.f` fails to complete the extraction for $minfr$ less than 10%.

To complement the results given by Fig. C.1, Table C.3 reports for different datasets and different values of $minfr$, the number of calls to the propagate routine of `Gecode` (column 5), and the number of nodes of the search tree (column 6). First, `PP` explores less nodes than `global-p.f`. But, the difference is not huge (gains of 45% and 33% on FIFA and BIBLE respectively). Second, our approach is very effective in terms of number of propagations. For `PP`, the number of propagations remains small (in thousands for small values of $minfr$) compared to `global-p.f` (in millions). This is due to the huge number of reified constraints used in `global-p.f` to encode the subsequence relation. On the contrary, our PREFIX-PROJECTION global constraint does not require any reified constraints nor any extra variables to encode the subsequence relation.

C.5.3 Comparing with ad hoc Methods for SPM

Our second experiment compares `PP` with state-of-the-art methods for SPM. Fig. C.2 shows the CPU times of the three methods. First, `cSpade` obtains the best performance on all datasets (except on Protein). However, `PP` exhibits a similar behavior as `cSpade`, but it is less faster (not counting the highest values of $minfr$). The behavior of `cSpade` on Protein is due to the vertical representation format that is not appropriated in the case of databases having large sequences

Dataset	$minfr$ (%)	#PATTERNS	CPU times (s)		#PROPAGATIONS		#NODES	
			PP	global-p.f	PP	global-p.f	PP	global-p.f
FIFA	20	938	**8.16**	129.54	**1884**	11649290	**1025**	1873
	18	1743	**13.39**	222.68	**3502**	19736442	**1922**	3486
	16	3578	**24.39**	396.11	**7181**	35942314	**3923**	7151
	14	7313	**44.08**	704	**14691**	65522076	**8042**	14616
	12	16323	**86.46**	1271.84	**32820**	126187396	**18108**	32604
	10	40642	**185.88**	2761.47	**81767**	266635050	**45452**	81181
BIBLE	10	174	**1.98**	105.01	**363**	4189140	**235**	348
	8	274	**2.47**	153.61	**575**	5637671	**362**	548
	6	508	**3.45**	270.49	**1065**	8592858	**669**	1016
	4	1185	**5.7**	552.62	**2482**	15379396	**1575**	2371
	2	5311	**15.05**	1470.45	**11104**	39797508	**7048**	10605
	1	23340	**41.4**	3494.27	**49057**	98676120	**31283**	46557
PubMed	5	2312	**8.26**	253.16	**4736**	15521327	**2833**	4619
	4	3625	**11.17**	340.24	**7413**	20643992	**4428**	7242
	3	6336	**16.51**	536.96	**12988**	29940327	**7757**	12643
	2	13998	**28.91**	955.54	**28680**	50353208	**17145**	27910
	1	53818	**77.01**	2581.51	**110133**	124197857	**65587**	107051
Protein	99.99	127	**165.31**	219.69	**264**	26731250	**172**	221
	99.988	216	**262.12**	411.83	**451**	44575117	**293**	390
	99.986	384	**467.96**	909.47	**805**	80859312	**514**	679
	99.984	631	**753.3**	1443.92	**1322**	132238827	**845**	1119
	99.982	964	**1078.73**	2615	**2014**	201616651	**1284**	1749
	99.98	2143	**2315.65**	–	**4485**	–	**2890**	–

TABLE C.3 – PP vs. global-p.f.

and small number of distinct items, thus degrading the performance of the mining process. Second, PP which also uses the concept of projected databases, clearly outperforms PrefixSpan on all datasets. This is due to our filtering algorithm combined together with incremental data structures to manage the projected databases. On FIFA, PrefixSpan is not able to complete the extraction for $minfr$ less than 12%, while our approach remains feasible until 6% within the time limit. On Protein, PrefixSpan fails to complete the extraction for all values of $minfr$ we considered. These results clearly demonstrate that our approach competes well with state-of-the-art methods for SPM on large datasets and achieves scalability while it is a major issue of existing CP approaches.

C.5.4 SPM under size and item constraints

Our third experiment aims at assessing the interest of pushing simultaneously different types of constraints. We impose on the PubMed dataset usual constraints such as *the minimum frequency* and the *minimum size* constraints and other useful constraints expressing some linguistic knowledge such as *the item constraint*. The goal is to retain sequential patterns which convey linguistic regularities (e.g., gene - rare disease relationships) [Béchet *et al.*, 2012]. *The size constraint* allows to remove patterns that are too small w.r.t. the number of items (number of words) to be relevant patterns. We tested this constraint with ℓ_{min} set to 3. *The item constraint* imposes that the extracted patterns must contain the item GENE and the item DISEASE. As no ad

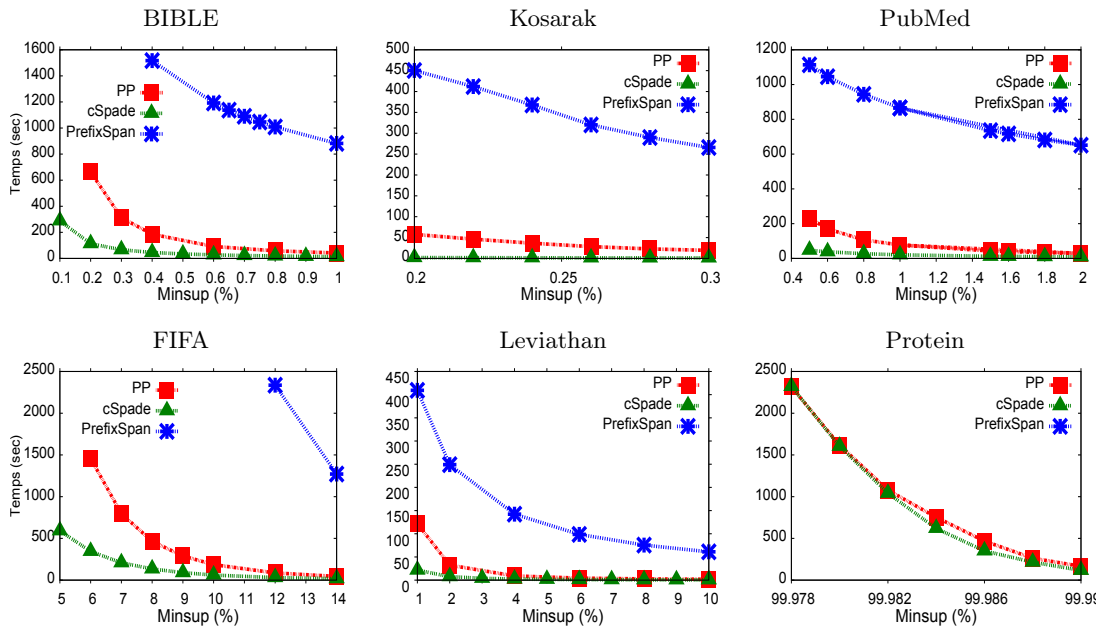
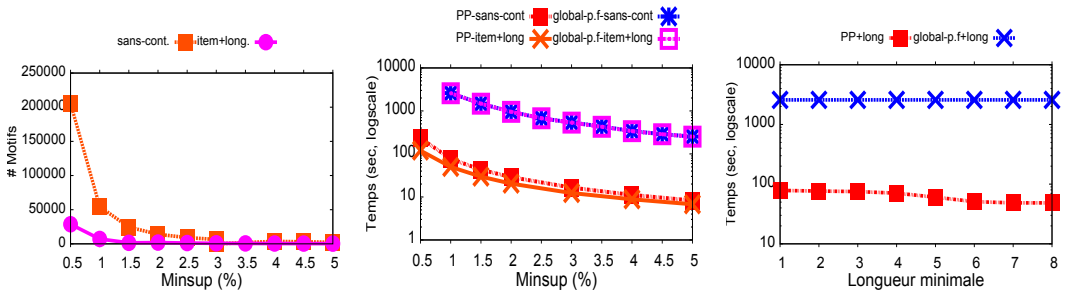


FIGURE C.2 – Comparing PREFIX-PROJECTION with state-of-the-art algorithms for SPM.

Dataset	$minfr$ (%)	#PATTERNS	CPU times (s)		#PROPAGATIONS		#NODES	
			PP	global-p.f	PP	global-p.f	PP	global-p.f
PubMed	5	279	**6.76**	252.36	**7878**	12234292	**2285**	4619
	4	445	**8.81**	339.09	**12091**	16475953	**3618**	7242
	3	799	**12.35**	535.32	**20268**	24380096	**6271**	12643
	2	1837	**20.41**	953.32	**43088**	42055022	**13888**	27910
	1	7187	**49.98**	2574.42	**157899**	107978568	**52508**	107051

TABLE C.4 – PP vs. global-p.f under minimum size and item constraints.

hoc method exists for this combination of constraints, we only compare PP with global-p.f. Fig. C.3 shows the CPU times and the number of sequential patterns extracted with and without constraints. First, pushing simultaneously the two constraints enables to reduce significantly the number of patterns. Moreover, the CPU times for PP decrease slightly whereas for global-p.f (with and without constraints), they are almost the same. This is probably due to the weak communication between the m exists-embedding reified global constraints and the two constraints. This reduces significantly the quality of the whole filtering. Second (see Table C.4), when considering the two constraints, PP clearly dominates global-p.f (speed-up value up to 51.5). Moreover, the number of propagations performed by PP remains very small as compared to global-p.f. Fig. C.3c compares the two methods under the minimum size constraint for different values of ℓ_{min}, with $minfr$ fixed to 1%. Once again, PP is always the most performer method (speed-up value up to 53.1). These results also confirm what we observed previously, namely the weak communication between reified global constraints and constraints imposed on patterns (i.e., size and item constraints).



(a) # of patterns (b) CPU times (logscale) (c) Minimum size constraint

FIGURE C.3 – Comparing PP with global-p.f under minimum size and item constraints on PubMed.

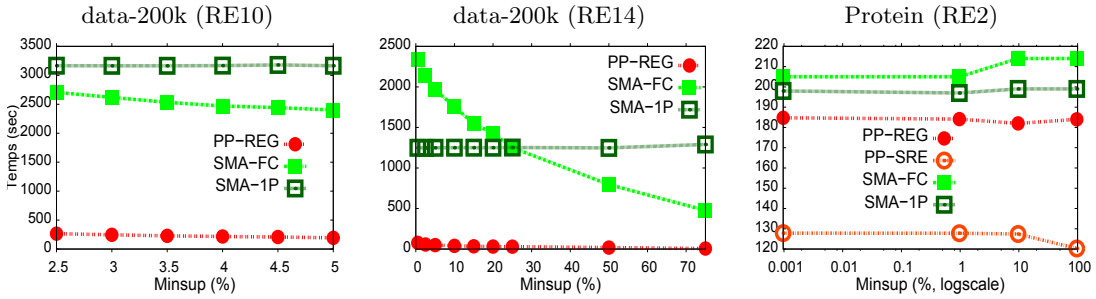


FIGURE C.4 – Comparing PREFIX-PROJECTION with SMA for SPM under RE constraint.

C.5.5 SPM under regular constraints

Our last experiment compares PP-REG against two variants of SMA : SMA-1P (SMA one pass) and SMA-FC (SMA Full Check). Two datasets are considered from [Trasarti *et al.*, 2008] : one synthetic dataset (data-200k), and one real-life dataset (Protein). For data-200k, we used two RE : RE10 $\equiv A^*B(B|C)D^*EF^*(G|H)I^*$ and RE14 $\equiv A^*(Q|BS^*(B|C))D^*E(I|S)^*(F|H)G^*R$. For Protein, we used RE2 $\equiv (S|T)\ .\ (R|K)$ (where . represents any symbol). Fig. C.4 reports CPU-times comparison. On the synthetic dataset, our approach is very effective. For RE14, our method is more than an order of magnitude faster than SMA. On Protein, the gap between the 3 methods shrinks, but our method remains effective. For the particular case of RE2, the Regular constraint can be substituted by restricting the domain of the first and third variables to $\{S, T\}$ and $\{R, K\}$ respectively (denoted as PP-SRE), thus improving performances.

C.6 Conclusion

We have proposed the global constraint PREFIX-PROJECTION for sequential pattern mining. PREFIX-PROJECTION uses a concise encoding and provides an efficient filtering based on specific properties of the projected databases, and anti-monotonicity of the frequency constraint. When this global constraint is integrated into a CP solver, it enables to handle several constraints simultaneously. Some of them like size, item membership and regular expression are considered in this paper. Another point of strength, is that, contrary to existing CP approaches for SPM, our global constraint does not require any reified constraints nor any extra variables to encode

the subsequence relation. Finally, although PREFIX-PROJECTION is well suited for constraints on sequences, it would require to be adapted to handle constraints on subsequence relations like gap.

Experiments performed on several real-life datasets show that our approach clearly outperforms existing CP approaches and competes well with ad hoc methods on large datasets and achieves scalability while it is a major issue of CP approaches. As future work, we intend to handle constraints on set of sequential patterns such as closedness, relevant subgroup and skypattern constraints.

Annexe D

Global Constraint for mining Sequential Patterns with GAP constraint

Article [Kemmar *et al.*, 2016] : paru dans les actes du congrès CPAIOR en 2016

Auteurs : Amina Kemmar, Samir Loudni, Yahia Lebbah, Patrice Boizumault, Thierry Charnois

Abstract

Sequential pattern mining (SPM) under gap constraint is a challenging task. Many efficient specialized methods have been developed but they are all suffering from a lack of genericity. The Constraint Programming (CP) approaches are not so effective because of the size of their encodings. In [Kemmar *et al.*, 2015], we have proposed the global constraint Prefix-Projection for SPM which remedies to this drawback. However, this global constraint cannot be directly extended to support gap constraint. In this paper, we propose the global constraint GAP-SEQ enabling to handle SPM with or without gap constraint. GAP-SEQ relies on the principle of right pattern extensions. Experiments show that our approach clearly outperforms both CP approaches and the state-of-the-art `cSpade` method on large datasets.

D.1 Introduction

Mining sequential patterns (SPM) is an important task in data mining. There are many useful applications, including discovering changes in customer behaviors, detecting intrusion from web logs and finding relevant genes from DNA sequences. In recent years many studies have focused on SPM with gap constraints [Zaki, 2000, Zhang *et al.*, 2007]. Limited gaps allow a mining process to bear a certain degree of flexibility among correlated pattern elements in the original sequences. For example, [Ji *et al.*, 2005] analyses purchase behaviors to reflect products usually bought by customers at regular time intervals according to time gaps. In computational biology, the gap constraint helps discover periodic patterns with significant biological and medical values [Wu *et al.*, 2013].

Mining sequential patterns under gap constraint (GSPM) is a challenging task, since the *apriori property* does not hold for this problem : *a subsequence of a frequent sequence is not necessarily frequent*. Several specialized approaches have been proposed [Ji *et al.*, 2005, Li *et al.*, 2012,

Zhang *et al.*, 2007] but they have a lack of genericity to handle simultaneously various types of constraints. Recently, a few proposals [Kemmar *et al.*, 2014, Métivier *et al.*, 2013, Négrevergne et Guns, 2015] have investigated relationships between GSPM and constraint programming (CP) in order to provide a declarative approach, while exploiting efficient and generic solving methods. But, due to the size of the proposed encodings, these CP methods are not as efficient as specialized ones. More recently, we have proposed the global constraint Prefix-Projection for SPM which remedies this drawback [Kemmar *et al.*, 2015]. However, this global constraint cannot be directly extended to support gap constraint.

In this paper, we introduce the global constraint GAP-SEQ enabling to handle SPM with or without gap constraint. GAP-SEQ relies on the principle of right pattern extension and its filtering exploits the prefix anti-monotonicity property of the gap constraint to provide an efficient pruning of the search space. GAP-SEQ enables to handle simultaneously different types of constraints and its encoding does not require any reified constraints nor any extra variables. Finally, experiments show that our approach clearly outperforms CP approaches as well as specialized methods for GSPM and achieves scalability while it is a major issue for CP approaches.

The paper is organized as follows. Section D.2 introduces the prefix anti-monotonicity of the gap constraint as well as right pattern extensions that will enable an efficient filtering. Section D.3 provides a critical review of specialized methods and CP approaches for sequential pattern mining under gap constraint. Section D.4 presents the global constraint GAP-SEQ. Section D.5 reports experiments we performed. Finally, we conclude and draw some perspectives.

D.2 Preliminaries

First, we provide the basic definitions for GSPM. Then, we show that the anti-monotonicity property of frequency of SPM does not hold for GSPM. Finally, we introduce right pattern extensions that will enable an efficient filtering for GSPM.

D.2.1 Definitions

Let $\mathcal{I}$ be a finite set of distinct *items*. The language of sequences corresponds to $\mathcal{L}_{\mathcal{I}} = \mathcal{I}^n$ where $n \in \mathbb{N}^*$.

Définition D.25 (sequence, sequence database)
A sequence s over $\mathcal{L}_{\mathcal{I}}$ is an ordered list $\langle s_1 \dots s_n \rangle$, where s_i, $1 \leq i \leq n$, is an item. n is called the length of the sequence s. A sequence database SDB is a set of tuples (sid, s), where sid is a sequence identifier and s a sequence denoted by $SDB[sid]$.

We now define the subsequence relation $\preceq^{[M,N]}$ under $gap[M,N]$ constraint which restricts the allowed distance between items of subsequences in sequences.

Définition D.26 (subsequence relation $\preceq^{[M,N]}$ under $gap[M,N]$)
$\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ is a subsequence of $s = \langle s_1 \dots s_n \rangle$, under $gap[M,N]$, denoted by $(\alpha \preceq^{[M,N]} s)$, if $m \leq n$ and, for all $1 \leq i \leq m$, there exist integers $1 \leq j_1 \leq \dots \leq j_m \leq n$, such that $\alpha_i = s_{j_i}$, and $\forall k \in \{1, ..., m-1\}, M \leq j_{k+1} - j_k - 1 \leq N$. In this context, the pair $(s, [j_1, j_m])$ denotes an **occurrence** of α in s, where j_1 and j_m represent the positions of the first and last items of α in s. We say that α is contained in s or s is a super-sequence of α under $gap[M,N]$. We also say that α is a $gap[M,N]$ **constrained pattern** in s.

sid	Sequence
1	$\langle ABCDB \rangle$
2	$\langle ACCBACB \rangle$
3	$\langle ADCBEEC \rangle$
4	$\langle AACC \rangle$

TABLE D.1 – A sequence database example SDB_1.

— Let $AllOcc(\alpha, s) = \{[j_1, j_m] \,|\, (s, [j_1, j_m])$ is an occurrence of α in $s\}$ be the set of all the occurrences of some sequence α under $gap[M, N]$ in s.

— Let $AllOcc(\alpha, SDB) = \{(sid, AllOcc(\alpha, SDB[sid])) \,|\, (sid, SDB[sid]) \in SDB\}$ be the set of all the occurrences of some sequence α under $gap[M, N]$ in SDB.

— Let $gap[M, \infty]$ and $gap[0, N]$ the **minimum and the maximum gap** constraints respectively. The relation $\preceq$ stands for $\preceq^{[0,\infty]}$ where the gap constraint is inactive.

For example, the sequence $\langle BABC \rangle$ is a super-sequence of $\langle AC \rangle$ under $gap[0, 2] : \langle AC \rangle \preceq^{[0,2]} \langle BABC \rangle$.

Définition D.27 (PREFIX, POSTFIX)
Let $\beta = \langle \beta_1 \dots \beta_n \rangle$ be a sequence. The sequence $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ where $m \leq n$ is called the prefix of β iff $\forall i \in [1..m], \alpha_i = \beta_i$. The sequence $\gamma = \langle \beta_{m+1} \dots \beta_n \rangle$ is called the postfix of s w.r.t. α. With the standard concatenation operator "$concat$", we have $\beta = concat(\alpha, \gamma)$.

The cover of a sequence α in SDB is the set of all tuples in SDB in which α is contained. The support of a sequence α in SDB is the cardinal of its cover.

Définition D.28 (COVER AND SUPPORT)
Let α be a sequence. $\texttt{cover}_{SDB}^{[M,N]}(\alpha) = \{(sid, s) \in SDB \,|\, \alpha \preceq^{[M,N]} s\}$ and $\texttt{freq}_{SDB}^{[M,N]}(\alpha) = \#\texttt{cover}_{SDB}^{[M,N]}(\alpha)$.

Définition D.29 ($gap[M, N]$ CONSTRAINED SEQUENTIAL PATTERN MINING (GSPM))
Given a sequence database SDB, a minimum support threshold $minfr$ and a gap constraint $gap[M, N]$. The problem of $gap[M, N]$ constrained sequential pattern mining is to find all subsequences α such that $\texttt{freq}_{SDB}^{[M,N]}(\alpha) \geq minfr$.

Exemple D.13. Table D.1 represents a sequence database of four sequences where the set of items is $\mathcal{I} = \{A, B, C, D, E\}$. Let the sequence $\alpha = \langle AC \rangle$. The occurrences under $gap[0, 1]$ of α in $SDB_1[2]$ is given by $AllOcc(\alpha, SDB_1[2]) = \{[1, 2], [1, 3], [5, 6]\}$. We have $cover_{SDB_1}^{[0,1]}(\alpha) = \{(1, s_1), (2, s_2), (3, s_3), (4, s_4)\}$. If we consider $minfr = 2$, α is a $gap[0, 1]$ constrained sequential pattern because $sup_{SDB_1}^{[0,1]}(\alpha) \geq 2$.

D.2.2 Prefix anti-monotonicity of $gap[M, N]$

Most SPM algorithms rely on the *anti-monotonicity property of frequency* to reduce the search space : all the subsequences of a frequent sequence are frequent as well (or, equivalently, if a subsequence is infrequent, then no super-sequence of it can be frequent). However, this property does not hold for the gap constraint, and more precisely for the maximum gap constraint. A simple

illustration from our running example suffices to show that sequence $\langle AB \rangle$ is not a sequential pattern under $gap[0,1]$ (for $minfr = 3$) whereas sequence $\langle ACB \rangle$ is a $gap[0,1]$ constrained sequential pattern. As a consequence, one needs to use other techniques for pruning the search space. The following proposition shows how the *prefix anti-monotonicity property* introduced in [Pei *et al.*, 2002] can be exploited to ensure the prefix anti-monotonicity of the gap constraint.

Définition D.30 (PREFIX ANTI-MONOTONE PROPERTY [PEI *et al.*, 2002])
A constraint c is called prefix anti-monotone if for every sequence α satisfying c, every prefix of α also satisfies the constraint.

Proposition D.6 ($gap[M,N]$)
$gap[M,N]$ is prefix anti-monotone.

Proof : Let $\alpha = \langle \alpha_1 \dots \alpha_m \rangle$ and $s = \langle s_1 \dots s_n \rangle$ be two sequences s.t. $\alpha \preceq^{[M,N]} s$ and $m \leq n$. By definition, there exist integers $1 \leq j_1 \leq \dots \leq j_m \leq n$, such that $\alpha_i = s_{j_i}$, and $\forall k \in \{1, ..., m-1\}, M \leq j_{k+1} - j_k - 1 \leq N$. As a consequence, the property also holds for every prefix of α. □

Hence, if a sequence α does not satisfy $gap[M,N]$, then all sequences that have α as prefix will not satisfy this constraint. Sect. D.4.2 shows how this property can be exploited to provide an efficient filtering.

D.2.3 Right pattern extensions

Right pattern extensions of some pattern p gives all the possible subsequences which can be appended at right of p to form a $gap[M,N]$ constrained pattern. According to proposition D.6, the set of all items locally frequent within the right pattern extensions of p in SDB can be used to extend p. In the following, we introduce an operator allowing to compute all the right pattern extensions of a pattern w.r.t. $gap[M,N]$.

Définition D.31 (RIGHT PATTERN EXTENSIONS)
Given some sequence (sid, s) and a pattern p s.t. $p \preceq^{[M,N]} s$. The **right pattern extensions** of p in s, denoted by $Ext_R^{[M,N]}(p,s)$, is the collection of legal subsequences of s located at the right of p and satisfying $gap[M,N]$. To define $Ext_R^{[M,N]}(p,s)$, we need to define $BE^{[M,N]}(p,s)$ **basic right extensions** :

$$BE^{[M,N]}(p,s) = \bigcup_{[j_1,j_m] \in AllOcc(p,s)} \{(j_m, \texttt{SubSeq}(s, j_m + M + 1, min(j_m + N + 1, \#s)))\}$$

$$\text{where } \texttt{SubSeq}(s, i_1, i_2) = \begin{cases} \langle s[i_1], ..., s[i_2] \rangle & \text{if } i_1 \leq i_2 \leq \#s \\ \langle\rangle & \text{otherwise} \end{cases}$$

Right pattern extensions $Ext_R^{[M,N]}(p,s)$ is defined as follows :

$$Ext_R^{[M,N]}(p,s) = \begin{cases} \{Sb \mid (j'_m, Sb) \in BE^{[M,N]}(p,s) \wedge & \text{if } N \geq \#s \\ \quad j'_m = \min_{(j_m,Sb) \in BE^{[M,N]}(p,s)} \{j_m\}\} & \\ \bigcup_{(j_m,Sb) \in BE^{[M,N]}(p,s)} \{Sb\} & \text{otherwise} \end{cases} \quad \text{(D.1)}$$

Formula (D.1) states exactly the set of all possible extensions of pattern p within s. In case where ($N \geq \#s$), since that any extension from $BE^{[M,N]}(p,s)$ always reaches the end of the sequence s, thus all possible extensions can be aggregated within one unique extension going from the lowest starting position $j'_m = \min_{(j_m,Sb)\in BE^{[M,N]}(p,s)}\{j_m\}$. We point out that these cases ($N \geq \#s$) cover the special case of no gap $gap[0,\infty]$.

The right pattern extensions of p in SDB is the collection of all its right pattern extensions in all sequences of SDB :

$$Ext_R^{[M,N]}(p, SDB) = \{(sid, Ext_R^{[M,N]}(p,s))|(sid,s) \in SDB \wedge p \preceq^{[M,N]} s\} \tag{D.2}$$

Exemple D.14. Let $p_1 = \langle AC\rangle$ be a pattern and the gap constraint be $gap[0,1]$. We have $AllOcc(p_1, SDB_1[2]) = \{[1,2),[1,3],[5,6]\}$. The right pattern extensions of p_1 in $SDB_1[2]$ is equal to $Ext_R^{[0,1]}(p_1, SDB_1[2]) = \{\langle CB\rangle, \langle BA\rangle, \langle B\rangle\}$. The right pattern extensions of p_1 in SDB_1 is given by $Ext_R^{[0,1]}(p_1, SDB_1) = \{(1,\{\langle DB\rangle\}),(2,\{\langle CB\rangle, \langle BA\rangle, \langle B\rangle\}),(3,\{\langle BE\rangle\}),(4,\{\langle C\rangle\})\}$.

Let the gap constraint be $gap[0,\infty]$. To compute $Ext_R^{[0,\infty]}(p_1, SDB_1[2])$, only the first occurrence of p_1 in $SDB_1[2]$ need to be considered (i.e. $[1,2)$) (cf. Definition D.31). $Ext_R^{[0,\infty]}(p_1, SDB_1[2])$ $= \{\langle CBACB\rangle\})$. The right pattern extensions of p_1 in SDB_1 is equal to $Ext_R^{[0,\infty]}(p_1, SDB_1)$ $= \{(1,\{\langle DB\rangle\}),\ (2,\{\langle CBACB\rangle\}),\ (3,\{\langle BEEC\rangle\}),\ (4,\{\langle C\rangle\})\}$.

We define $supExt_{SDB}^{[M,N]}(\alpha, p)$ as the support of α within the right pattern extensions :

$$\begin{aligned} supExt_{SDB}^{[M,N]}(\alpha,p) = \quad & \#\{(sid,s) \in SDB \,|\, \exists (sid,E) \in Ext_R^{[M,N]}(p, SDB), \\ & \exists\, s' \in E, \langle\alpha\rangle \preceq s')\}. \end{aligned} \tag{D.3}$$

Let $\mathcal{RF}_{SDB}^{[M,N]}(p)$ be the set of locally frequent items within the right extensions :

$$\begin{aligned} \mathcal{RF}_{SDB}^{[M,N]}(p) = \quad & \{v \in \mathcal{I} \,|\, \#\{sid \,|\, \exists (sid,E) \in Ext_R^{[M,N]}(p, SDB), \\ & \exists \alpha \in E, \langle v\rangle \preceq \alpha)\} \geq minfr\}. \end{aligned} \tag{D.4}$$

Given a $gap[M,N]$ constrained pattern p in SDB, according to proposition D.6, items in $\mathcal{RF}_{SDB}^{[M,N]}(p)$ can be used to extend p. Proposition D.7 establishes the support count of a sequence γ w.r.t. its right pattern extensions.

Proposition D.7 (Support count)
For any sequence γ in SDB with prefix α and postfix β s.t. $\gamma = concat(\alpha,\beta)$, $sup_{SDB}^{[M,N]}(\gamma) = supExt_{SDB}^{[M,N]}(\beta,\alpha)$.

This proposition ensures that only the sequences in SDB grown from α need to be considered for the support count of a sequence γ. From proposition D.7, we can derive the following proposition to establish a condition to check when a pattern is a $gap[M,N]$ constrained sequential pattern.

Proposition D.8 ($gap[M,N]$ constrained sequential pattern)
Let SDB be a sequence database and a minimum support threshold $minfr$. A pattern p is a

$gap[M,N]$ constrained sequential pattern in SDB if and only if the following condition holds : $\#Ext_R^{[M,N]}(p, SDB) \geq minfr$

Exemple D.15. Let $minfr$ be 2 and the gap constraint be $gap[0,1]$. From Example D.14, we have $\#Ext_R^{[0,1]}(p_1, SDB_1) = 4 \geq minfr$. Thus, $p_1 = \langle AC \rangle$ is a $gap[0,1]$ constrained sequential pattern. The locally frequent items within the right pattern extensions $Ext_R^{[0,1]}(p_1, SDB_1)$ of p_1 are B and C with supports of 3 and 2 respectively. According to proposition D.7, p_1 can be extended to two $gap[0,1]$ constrained sequential patterns $\langle ACB \rangle$ and $\langle ACC \rangle$.

D.3 Related works

Specialized methods for GSPM. The SPM was first proposed in [Agrawal et Srikant, 1995]. Since then, many efficient specialized approaches have been proposed [Pei *et al.*, 2001, Zaki, 2001]. There are also several methods focusing on gap constraints. Zaki [Zaki, 2000] first proposed `cSpade`, a depth-first search based on a vertical database format, incorporating constraints on gap (min_gap and max_gap) and time windows (max_span). Other constraints on length, items and classes for classification datasets are also mentioned in the paper but they are not supported in the author's `cSpade` implementation. Ji *et al.* [Ji *et al.*, 2005] and Li *et al.* [Li et Wang, 2008] studied the problem of mining frequent patterns with gap constraints. In [Ji *et al.*, 2005], a minimal distinguishing subsequence that occurs frequently in the positive sequences and infrequently in the negative sequences is proposed, where the maximum gap constraint is defined. In [Li et Wang, 2008], closed frequent patterns with gap constraints are mined. All these proposals, though efficient, lack genericity to handle simultaneously various types of constraints. Finally, Pei *et al.* [Pei *et al.*, 2002] have proposed an algorithm based on prefix-growth which handles constraints that are prefix anti-monotone. These classes of constraints are stated a posteriori and are only used for -testing- solutions (without any pruning). For the particular case of the gap constraint, when a current prefix satisfies a constraint, no pruning is achieved and all possible "right-parts" have to be tested.

CP Methods for GSPM. There are few methods for SPM with gap constraints using CP. [Métivier *et al.*, 2013] have proposed to model a sequence using an automaton capturing all subsequences that can occur in it. The gap constraint is encoded by removing from the automaton all transitions that do not respect the gap constraint. [Kemmar *et al.*, 2014] have proposed a CSP model for SPM with explicit wildcards. The gap constraints is enforced using the regular global constraint. [Négrevergne et Guns, 2015] have proposed two CP encodings for the SPM. The first one uses a global constraint to encode the subsequence relation (denoted `global-p.f`), while the second one (denoted `decomposed-p.f`) encodes explicitly this relation using additional variables and constraints in order to support constraints like gap. However, all these proposals usually lead to constraint networks of huge size. Space complexity is clearly identified as the main bottleneck behind the competitiveness of these declarative approaches. In [Kemmar *et al.*, 2015], we have proposed the global constraint Prefix-Projection for sequential pattern mining which remedies to this drawback. However, this constraint cannot be directly extended to handle gap constraints. This requires changing the way the subsequence relation is encoded.

The next section introduces the global constraint GAP-SEQ enabling to handle SPM with or without gap constraints. GAP-SEQ relies on the prefix anti-monotonicity of the gap constraint

and on the right pattern extensions to provide an efficient filtering. This global constraint does not require any reified constraints nor any extra variables to encode the subsequence relation.

D.4 GAP-SEQ global constraint

This section is devoted to the GAP-SEQ global constraint. Section D.4.1 defines the GAP-SEQ global constraint and presents the CSP modeling. Section D.4.2 shows how the filtering can take advantage of the prefix anti-monotonicity property of the $gap[M,N]$ constraint (see Proposition D.11) and of the right pattern extensions (see Proposition D.10) to remove inconsistent values from the domain of a future variable. Section D.4.3 details the filtering algorithm and Section D.4.4 provides its temporal and spatial complexities.

D.4.1 CSP modeling for GSPM

A *Constraint Satisfaction Problem* (CSP) consists of a set X of n variables, a domain $\mathcal{D}$ mapping each variable $X_i \in X$ to a finite set of values $D(X_i)$, and a set of constraints $\mathcal{C}$. An assignment σ is a mapping from variables in X to values in their domains. A constraint $c \in \mathcal{C}$ is a subset of the cartesian product of the domains of the variables that occur in c. The goal is to find an assignment such that all constraints are satisfied.

(a) Variables and domains. Let P be the unknown pattern of size ℓ we are looking for. The symbol $\square$ ($\square \notin \mathcal{I}$) stands for an empty item and denotes the end of a sequence. We encode the unknown pattern P of maximum length ℓ with a sequence of ℓ variables $\langle P_1, P_2, \ldots, P_\ell \rangle$. Each variable P_j represents the item in the jth position of the sequence. The size ℓ of P is determined by the length of the longest sequence of SDB. The domains of variables are defined as follows : (i) $D(P_1) = \mathcal{I}$ to avoid the empty sequence, and (ii) $\forall i \in \{2 \ldots \ell\}, D(P_i) = \mathcal{I} \cup \{\square\}$. To allow patterns with less than ℓ items, we impose that $\forall i \in \{2..(\ell-1)\}, (P_i = \square) \rightarrow (P_{i+1} = \square)$.

(b) Definition of GAP-SEQ. The global constraint GAP-SEQ encodes both subsequence relation $\preceq^{[M,N]}$ under gap constraint $gap[M,N]$ and minimum frequency constraint directly on the data.

Définition D.32 (GAP-SEQ global constraint)
Let $P = \langle P_1, P_2, \ldots, P_\ell \rangle$ be a pattern of size ℓ and $gap[M,N]$ be the gap constraint.

$$\text{GAP-SEQ}(P, SDB, minfr, M, N) = \{\langle d_1, ..., d_\ell \rangle \mid \forall\, i\, d_i \in D(P_i) : \texttt{freq}_{SDB}^{[M,N]}(\langle d_1, ..., d_\ell \rangle) \geq minfr\}$$

Proposition D.9 (GAP-SEQ consistency)
GAP-SEQ$(P, SDB, minfr, M, N)$ has a solution iff there exists an assignment $\sigma = \langle d_1, ..., d_\ell \rangle$ of variables of P s.t. $\#Ext_R^{[M,N]}(\sigma, SDB) \geq minfr$.

Proof : This is a direct consequence of proposition D.8. $\square$

(c) Other SPM constraints can be directly modeled as follows :
- *Minimum Size* constraint restricts the number of items of a pattern to be at least ℓ_{min} : $minSize(P, \ell_{min}) \equiv \bigwedge_{i=1}^{i=\ell_{min}} (P_i \neq \square)$
- *Maximum Size* constraint restricts the number of items of a pattern to be at most ℓ_{max} : $maxSize(P, \ell_{max}) \equiv \bigwedge_{i=\ell_{max}+1}^{i=\ell} (P_i = \square)$

- *Membership* constraint states that a subset of items V must belong (or not) to the extracted patterns. $item(P, V) \equiv \bigwedge_{t \in V} \texttt{Among}(P, \{t\}, l, u)$ enforces that items of V should occur at least l times and at most u times in P. To forbid items of V to occur in P, l and u must be set to 0.

D.4.2 Principles of filtering

(a) Maintaining a local consistency. SPM is a challenging task due to the exponential number of candidates that should be parsed to find the frequent patterns. For instance, we have $O(n^k)$ potential candidate patterns of length at most k in a sequence of length n. With gap constraints, the problem is even much harder since the complexity of checking for subsequences taking a gap constraint into account is higher than the complexity of the standard subsequence relation. Furthermore, the NP-hardness of mining maximal frequent sequences was established in [Yang, 2006] by proving the #P-completeness of the problem of counting the number of maximal frequent sequences. Hence, ensuring *Domain Consistency* (DC) for GAP-SEQ i.e., finding, for every variable P_j, a value $d_j \in D(P_j)$, satisfying the constraint is NP-hard.

So, the filtering of GAP-SEQ constraint maintains a consistency lower than DC. This consistency is based on specific properties of the $gap[M, N]$ constraint and resembles forward-checking (regarding Proposition D.10). GAP-SEQ is considered as a global constraint, since all variables share the same internal data structures that awake and drive the filtering. The prefix anti-monotonicity property of the $gap[M, N]$ constraint (see Proposition D.11) and of the right pattern extensions (see Proposition D.10) will enable to remove inconsistent values from the domain of a future variable.

(b) Detecting inconsistent values. Let $\mathcal{RF}_{SDB}^{[M,N]}(\sigma)$ be the set of locally frequent items within the right pattern extensions (see (D.4) in Sect. D.2.3). The following proposition characterizes values, of a future (unassigned) variable P_{j+1}, that are consistent with the current assignment of variables $\langle P_1, \ldots, P_j \rangle$.

Proposition D.10 (Consistent values)
Let [47] $\sigma = \langle d_1, \ldots, d_j \rangle$ be a current assignment of variables $\langle P_1, \ldots, P_j \rangle$, P_{j+1} be a future variable. A value $d \in D(P_{j+1})$ occurs in a solution for the global constraint GAP-SEQ$(P, SDB, minfr, M, N)$ iff $d \in \mathcal{RF}_{SDB}^{[M,N]}(\sigma)$.

Proof : Assume that $\sigma = \langle d_1, \ldots, d_j \rangle$ is $gap[M, N]$ constrained sequential pattern in SDB. Suppose that value $d \in D(P_{j+1})$ appears in $\mathcal{RF}_{SDB}^{[M,N]}(\sigma)$. As the local support of d within the right extensions (see (D.3)) is equal to $supExt_{SDB}^{[M,N]}(\langle d \rangle, \sigma)$, from proposition D.7 we have $\texttt{freq}_{SDB}^{[M,N]}(concat(\sigma, \langle d \rangle)) = supExt_{SDB}^{[M,N]}(\langle d \rangle, \sigma)$. Hence, we can get a new assignment $\sigma \cup \langle d \rangle$ that satisfies the constraint. Therefore, $d \in D(P_{j+1})$ participates in a solution. □

From proposition D.10 and according to the prefix anti-monotonicity property of the gap constraint, we can derive the following pruning rule :

Proposition D.11 (GAP-SEQ Filtering rules)
Let $\sigma = \langle d_1, \ldots, d_j \rangle$ be a current assignment of variables $\langle P_1, \ldots, P_j \rangle$. All values $d \in D(P_{j+1})$ that are not in $\mathcal{RF}_{SDB}^{[M,N]}(\sigma)$ can be removed from the domain of variable P_{j+1}.

47. We indifferently denote σ by $\langle d_1, \ldots, d_j \rangle$ or by $\langle \sigma(P_1), \ldots, \sigma(P_j) \rangle$.

Algorithme 12 : *FILTER-GAP-SEQ* (SDB, σ, j, P, $minfr$, M, N)

Data : SDB : initial database ; σ : current assignment $\langle \sigma(P_1), \ldots, \sigma(P_j) \rangle$; $minfr$: the minimum support threshold ; $\mathcal{ALLOCC}$: internal data structure for storing occurrences of patterns in SDB ; Ext_R : internal data structure for storing right pattern extensions of σ in SDB.

```
1  begin
2      Ext_R ← getRightExt(SDB, ALLOCC_{j-1}, σ, M, N) ;
3      if (#Ext_R < minfr) then
4          return False ;
5      if (j ≥ 2 ∧ σ(P_j) = □) then
6          for k ← j + 1 to ℓ do
7              P_k ← □ ;
8      else
9          RF ← getFreqItems (SDB, Ext_R, minfr) ;
10         foreach a ∈ D(P_{j+1}) s.t.(a ≠ □ ∧ a ∉ RF) do
11             D(P_{j+1}) ← D(P_{j+1}) − {a} ;
12     return True ;
13 end
```

Exemple D.16. Consider the running example of Table D.1, let $minfr$ be 2 and the gap constraint be $gap[1,2]$. Let $P = \langle P_1, P_2, P_3, P_4 \rangle$ with $D(P_1) = \mathcal{I}$ and $D(P_2) = D(P_3) = D(P_4) = \mathcal{I} \cup \{\square\}$. Suppose that $\sigma(P_1) = A$. We have $Ext_R^{[1,2]}(\langle A \rangle, SDB_1) = \{(1, \{\langle CD \rangle\}), (2, \{\langle CB \rangle, \langle B \rangle\}), (3, \{\langle CB \rangle\}), (4, \{\langle CC \rangle, \langle C \rangle\})\}$. As B and C are the only locally frequent items in $Ext_R^{[1,2]}(\langle A \rangle, SDB_1)$, GAP-SEQ will remove values A, D and E from $D(P_2)$.

D.4.3 Filtering algorithm

Algorithm 12 describes the pseudo-code of GAP-SEQ filtering algorithm. It takes as input : the index j of the last assigned variable in P, the current partial assignment $\sigma = \langle \sigma(P_1), \ldots, \sigma(P_j) \rangle$, the minimum support threshold $minfr$, the minimum and the maximum gaps. The internal data-structure $\mathcal{ALLOCC}$ stores all the intermediate occurrences of patterns in SDB, where $\mathcal{ALLOCC}_j = AllOcc(\sigma, SDB)$, for $j \in \{1 \ldots \ell\}$. If $\sigma = \langle\rangle$, then $\mathcal{ALLOCC}_0 = \{(sid, [1, \#s]) \mid (sid, s) \in SDB\}$.

Algorithm 12 starts by computing the right pattern extensions Ext_R of σ in SDB by calling function *getRightExt* (see Algorithm 13). Then, it checks whether the current assignment σ satisfies the constraint (line 3). If not, we stop growing σ and we return *False*. Otherwise, the algorithm checks if the last assigned variable P_j is instantiated to $\square$ (line 5). If so, the end of the sequence is reached (since value $\square$ can only appear at the end) and the sequence $\langle \sigma(P_1), \ldots, \sigma(P_j) \rangle$ is a $gap[M,N]$ constrained sequential pattern in SDB ; hence, the algorithm sets the remaining $(\ell - j)$ unassigned variables to $\square$ and returns *True* (lines 6-7). If $(P_j \neq \square)$, the set of locally frequent items, within the right pattern extensions Ext_R of σ in SDB, is computed by calling function *getFreqItems* (line 9) and the domain of variable P_{j+1} is updated accordingly (lines 10-11).

Algorithm 13 gives the pseudo-code of the function *getRightExt* . First, if σ is empty (i.e. $\#\sigma = 0$), all the sequences of SDB are considered as valid right pattern extensions ; the whole SDB should be returned. Otherwise, the function *getAllOcc* is called to compute the occurrences of σ in SDB (line 4). Then, the algorithm processes all the entries of $\mathcal{ALLOCC}_j$, one by one (line 6), and, for each pair $(sid, OccSet)$, scans the occurrences of σ in the sequence sid (line 8).

Algorithme 13 : *getRightExt* (SDB, $\mathcal{ALLOCC}_{j-1}$, σ, M, N)

Data : SDB : initial database ; $\mathcal{ALLOCC}_{j-1}$: occurrences of the partial assignment $\langle\sigma(P_1),\ldots,\sigma(P_{j-1})\rangle$ in SDB ; σ : the current partial assignment $\langle\sigma(P_1),\ldots,\sigma(P_j)\rangle$; $OccSet$: the positions of the first and last items of $\langle\sigma(P_1),\ldots,\sigma(P_{j-1})\rangle$ in $SDB[sid]$; Sb : the positions of the first and last items of the right pattern extensions of σ in $SDB[sid]$.

```
1  begin
2      if (σ = ⟨⟩) then
3          return {(sid, (1, #s))|(sid, s) ∈ SDB} ;
4      ALLOCC_j ← getAllOcc (SDB, ALLOCC_{j−1}, σ, M, N) ;
5      Ext_R ← ∅ ;
6      foreach pair (sid, OccSet) ∈ ALLOCC_j do
7          s ← SDB[sid] ; Sb ← ∅ ;
8          foreach pair [j_1, j_m] ∈ OccSet do
9              j'_1 ← j_m + M + 1 ; j'_m ← min(j_m + N + 1, #s) ;
10             if (j'_1 ≤ j'_m) then
11                 Sb ← Sb ∪ {(j'_1, j'_m)} ;
12         Ext_R ← Ext_R ∪ {(sid, Sb)} ;
13     return Ext_R ;
14 end
15 Function getAllOcc (SDB, ALLOCC_{j−1}, σ, M, N) ;
16 begin
17     ALLOCC_j ← ∅ ; inf ← 0 ; sup ← 0;
18     foreach pair (sid, OccSet) ∈ ALLOCC_{j−1} do
19         s ← SDB[sid] ; newOccSet ← ∅ ; redundant ← false ; i ← 1 ;
20         while (i ≤ #OccSet ∧¬redundant) do
21             [j_1, j_m] ← OccSet[i] ; i ← i + 1;
22             if (#σ = 1) then
23                 inf ← 1 ; sup ← #s ;
24             else
25                 inf ← j_m + M + 1 ; sup ← min(j_m + N + 1, #s) ;
26             k ← inf ;
27             while ((k ≤ sup) ∧ (¬redundant)) do
28                 if (s[k] = σ(P_j)) then
29                     if (#σ = 1) then
30                         newOccSet ← newOccSet ∪ {[k, k]} ;
31                     else
32                         newOccSet ← newOccSet ∪ {[j_1, k]} ;
33                     if (((sup = #s) ∧ (#σ > 1)) ∨ (N ≥ #s)) then
34                         redundant ← true ;
35                 k ← k + 1 ;
36         if (newOccSet ≠ ∅) then
37             ALLOCC_j ← ALLOCC_j ∪ (sid, newOccSet) ;
38     return ALLOCC_j ;
39 end
```

For each occurrence $[j_1, j_m] \in OccSet$, the algorithm computes its right pattern extensions, i.e. the part of the sequence sid which is in the range $[j_m + M + 1, min(j_m + N + 1, \#s)]$ (line 9). If the new range is valid, it is added to the set Sb (line 11). After processing the whole entries in $OccSet$, the right pattern extensions of σ in the sequence sid are built and then added to the set Ext_R (line 12). The process ends when all entries of $\mathcal{ALLOCC}_j$ have been considered. The right pattern extensions of σ in SDB are then returned (line 13).

Function *getAllOcc* computes incrementally $\mathcal{ALLOCC}_j$ from $\mathcal{ALLOCC}_{j-1}$. More precisely, lines (22-23) and (29-30) are considered when the first variable P_1 is instantiated (i.e. $\#\sigma = 1$),

and consequently all of its initial occurrences should be found and stored in $\mathcal{ALLOCC}_1$ through the initialization step (lines 29-30). After that, $\mathcal{ALLOCC}_j (j > 1)$ is incrementally computed from $\mathcal{ALLOCC}_{j-1}$ through line (32).

Exemple D.17. Consider the running example of Table D.1, let the gap constraint be $gap[0, 4]$, and $\sigma = \langle ACB \rangle$. The occurrences of $\langle A \rangle$ in SDB_1 are stored in $\mathcal{ALLOCC}_1 = \{(1, \{[1, 1]\}), (2, \{[1, 1], [5, 5]\}), (3, \{[1, 1]\}), (4, \{[1, 1], [2, 2]\})\}$. From $\mathcal{ALLOCC}_1$, we get the occurrences of $\langle AC \rangle : \mathcal{ALLOCC}_2 = \{(1, \{[1, 3]\}), (2, \{[1, 2], [1, 3], [\mathbf{1}, \mathbf{6}], [\mathbf{5}, \mathbf{6}]\}), (3, \{[1, 3]\}), (4, \{[\mathbf{1}, \mathbf{3}], [\mathbf{2}, \mathbf{3}], [\mathbf{1}, \mathbf{4}], [\mathbf{2}, \mathbf{4}],\})\}$. But, as $BE^{[M,N]}(p, s)$ is only based on the final position j_m of each occurrence (see Definition D.31), the occurrences with the same final position j_m (in bold in our example) are considered only once. Thus, $\mathcal{ALLOCC}_2 = \{(1, \{[1, 3]\}), (2, \{[1, 2], [1, 3], [1, 6]\}), (3, \{[1, 3]\}), (4, \{[1, 3], [1, 4]\})\}$.

We avoid computing occurrences leading to redundant right pattern extensions thanks to the conditions $((sup = \#s) \wedge (\#\sigma > 1))$ in line (33). Moreover, when computing the right pattern extensions, instead of storing the part of subsequence $\langle s[j'_1], \ldots, s[j'_m] \rangle$, one can only store the positions of its first and last items (j'_1, j'_m) in the sequence *sid*. Finally, the filtering algorithm handles as efficiently the case *without gap constraints*. For each pair $(sid, OccSet)$, only the first occurrence $[j_1, j_m]$ in $OccSet$ is determined thanks to the condition $(N \geq \#s)$ in line (33).

D.4.4 Temporal and spatial complexities of the filtering algorithm

Let m=$|SDB|$, d=$|\mathcal{I}|$, and ℓ be the length of the longest sequence in SDB. Computing $\mathcal{ALLOCC}_j$ from $\mathcal{ALLOCC}_{j-1}$ (see function *GetAllOcc* of Algorithm 13) can be achieved in $O(m \times \ell^2)$. The function *getRightExt* (see Algorithm 13) processes all the occurrences of σ in each sequence of the SDB. The number of occurrences may exceed ℓ. However, as occurrences $(s, [j_1, j_m])$ with the same final position j_m are considered only once (see operator BE in Definition D.31), there may exist at most ℓ of such occurrences in each sequence of the SDB in the worst case. So, the time complexity of function *getRightExt* is $O(m \times \ell^2 + m \times \ell)$ i.e. $O(m \times \ell^2)$.

Proposition D.12 (Complexities)
In the worst case, (i) filtering can be achieved in $O(m \times \ell^2 + d)$ and (ii) the space complexity is $O(m \times \ell^2)$.

Proof : (i) The complexity of function *getRightExt* is $O(m \times \ell^2)$. The total complexity of function *GetFreqItems* is $O(m \times \ell)$. Lines (10-11) can be achieved in $O(d)$. So, the whole complexity is $O(m \times \ell^2 + m \times \ell + d)$, i.e. $O(m \times \ell^2 + d)$.
(ii) The space complexity of the filtering algorithm lies in the storage of the $\mathcal{ALLOCC}$ internal data structure. The occurrences $\mathcal{ALLOCC}_j$ of each assignment σ in SDB, with the length of σ varying from 1 to ℓ, have to be stored. Since it may exist at most ℓ occurrences of σ in each sequence *sid*, storing any $\mathcal{ALLOCC}_j$ costs in the worst case $O(m \times \ell)$. Since we can have ℓ prefixes, the worst space complexity of storing all the occurrences $\mathcal{ALLOCC}_j (j = 1..\ell)$, is $O(m \times \ell^2)$. □

D.5 Experiments

This section reports experiments on several real-life datasets [Fournier-Viger *et al.*, 2014, Béchet *et al.*, 2012] of large size having varied characteristics and representing different application domains (see Tab. D.2). First, we compare our approach with CP methods and with the

dataset	$\#SDB$	$\#\mathcal{I}$	avg ($\#s$)	$\texttt{Max}_{s \in SDB}$ ($\#s$)	type of data
Leviathan	5834	9025	33.81	100	book
PubMed	17527	19931	29	198	bio-medical text
FIFA	20450	2990	34.74	100	web click stream
BIBLE	36369	13905	21.64	100	bible
Kosarak	69999	21144	7.97	796	web click stream
Protein	103120	24	482	600	protein sequences

TABLE D.2 – Dataset Characteristics.

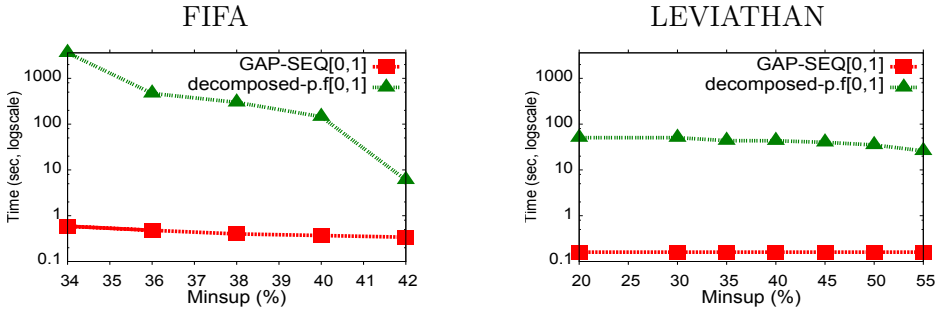


FIGURE D.1 – Comparing `GAP-SEQ` with `decomposed-p.f` for GSPM : CPU times.

state-of-the-art specialized method `cSpade` in terms of scalability. Second, we show the flexibility of our approach for handling different types of constraints simultaneously.

D.5.1 Experimental protocol

Our approach was carried out using the `gecode` solver [48]. All experiments were conducted on a processor Intel X5670 with 24 GB of memory. A time limit of 1 hour has been set. If an approach is not able to complete the extraction within the time limit, it will be reported as $(-)$. ℓ was set to the length of the longest sequence of SDB. We compare our approach (indicated by `GAP-SEQ`) with :

1. `decomposed-p.f` [49], the most efficient CP methods for GSPM,
2. `cSpade` [50], the state-of-the-art specialized method for GSPM,
3. the PREFIX-PROJECTION global constraint for SPM.

D.5.2 GSPM : `GAP-SEQ` vs the most efficient CP method

We compare CPU times for `GAP-SEQ` and `decomposed-p.f`. In the experiments, we used the gap constraint $gap[0, 1]$ and various values of $minsup$. Fig. D.1 shows the results for the two datasets FIFA and LEVIATHAN (results are similar for other datasets and not reported due to page limitation). `GAP-SEQ` clearly outperforms `decomposed-p.f` on the two datasets even for high values of $minsup$: `GAP-SEQ` is more than an order of magnitude faster than `decomposed-p.f`. For low values of $minsup$, `decomposed-p.f` fails to complete the extraction within the time limit.

48. `http://www.gecode.org`
49. `https://dtai.cs.kuleuven.be/CP4IM/cpsm/`
50. `http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/`

Dataset	*minsup* (%)	#PATTERNS	CPU times (s)		#PROPAGATIONS		#NODES	
			`GAP-SEQ`	`decomposed-p.f`	`GAP-SEQ`	`decomposed-p.f`	`GAP-SEQ`	`decomposed-p.f`
FIFA	42	1	0.34	6.06	2	0	1	2
	40	5	0.37	144.95	10	778010	6	11
	38	10	0.4	298.68	20	2957965	11	21
	36	17	0.48	469.3	34	9029578	18	35
	34	35	0.59	–	70	–	36	–

TABLE D.3 – `GAP-SEQ` vs. `decomposed-p.f` on FIFA dataset.

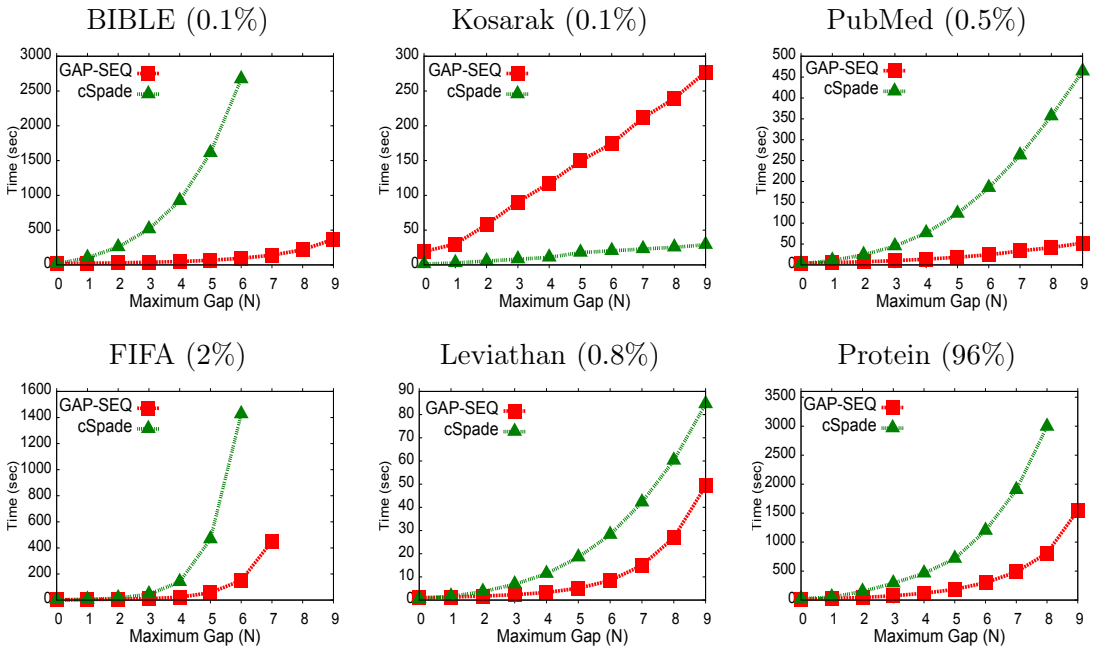


FIGURE D.2 – Varying the value of parameter N in the gap constraint ($M = 0$) : CPU times.

Tab. D.3 reports for the FIFA dataset and different values of *minsup*, the number of calls to the propagate function of `gecode` (col. 5) and the number of nodes of the search tree (col. 6). `GAP-SEQ` is very effective in terms of number of propagations. For `GAP-SEQ`, the number of propagations remains very small compared to `decomposed-p.f` (millions). This is due to the huge number of reified constraints used by `decomposed-p.f` to encode the subsequence relation. Regarding CPU times, `GAP-SEQ` requires less than 1s. to complete the extraction, while `decomposed-p.f` needs much more time to end the extraction (speed-up value up to 938).

D.5.3 GSPM : `GAP-SEQ` vs the state-of-the-art specialized method

Second experiments compare `GAP-SEQ` with `cSpade`. We first fixed *minsup* to the smallest possible value w.r.t. the dataset used, and varied the maximum gap N from 0 to 9. The minimum gap M was set to 0. Fig. D.2 reports the CPU times of both methods. First, `GAP-SEQ` clearly dominates `cSpade` on all the datasets. The gains in terms of CPU times are greatly amplified as the value of N increases. On FIFA, the speed-up is 9.5 for N=6. On BIBLE, `GAP-SEQ` is able to complete the extraction for values of N up to 9 in 433 seconds, while `cSpade` failed to complete

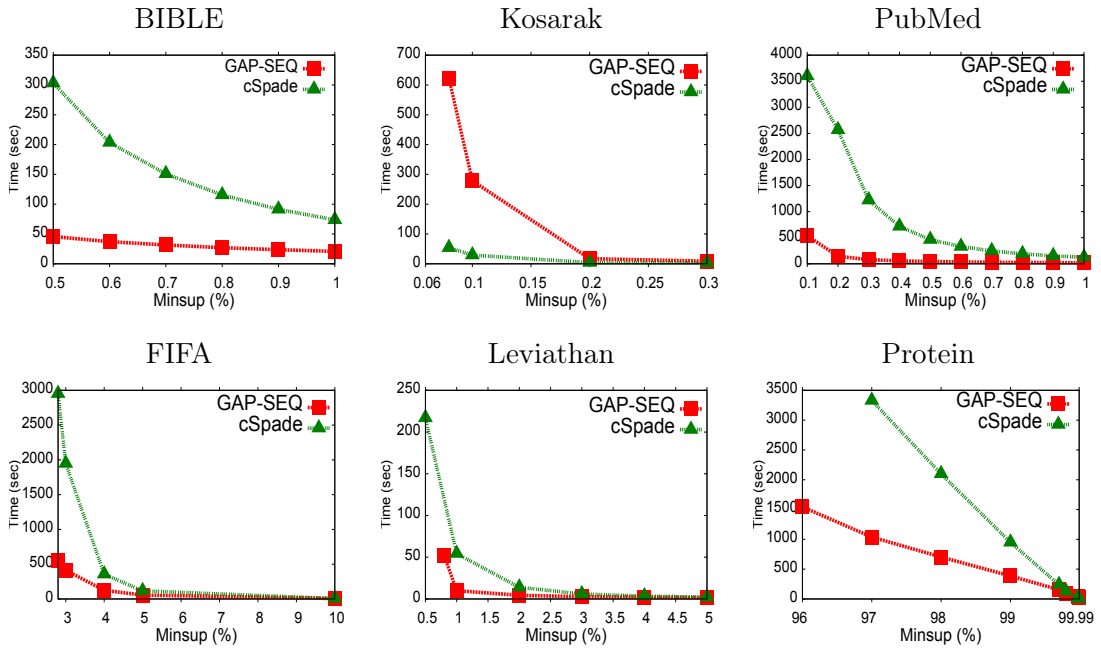


FIGURE D.3 – Varying the value of *minsup* with the gap constraint $gap[0, 9]$: CPU times.

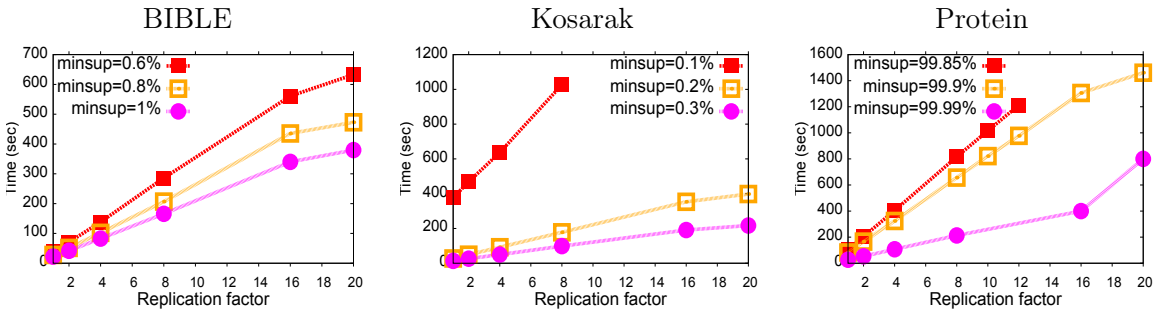


FIGURE D.4 – Scalability of `GAP-SEQ` global constraint on BIBLE, Kosarak and Protein.

the extraction for N greater than 6. The only exception is for the Kosarak dataset, where `cSpade` is efficient. For this dataset (which is the largest one both in terms of number of sequences and items), the size of the domains is important as compared to the other datasets. So, filtering takes much more time. This probably explains the behavior of `GAP-SEQ` on this dataset.

We also conducted experiments to evaluate how sensitive `GAP-SEQ` and `cSpade` are to *minsup*. We used the $gap[0, 9]$ constraint, while *minsup* varied until the two methods were not able to complete the extraction within the time limit. Results are depicted in Fig. D.3. Once again, `GAP-SEQ` obtains the best performance on all datasets (except for Kosarak). When the minimum support decreases, CPU times for `GAP-SEQ` increase reasonably while for `cSpade` they increase dramatically. On PubMed, with *minsup* set to 0.1%, `cSpade` finished the extraction after $3,500$ seconds, while `GAP-SEQ` only used 500 seconds (speed-up value 7). These results clearly demonstrate that our approach is very effective as compared to `cSpade` on large datasets.

minsup	#PATTERNS		CPU times (s)		#PROPAGATIONS		#NODES	
	gap	gap+size+item	gap	gap+size+item	gap	gap+size+item	gap	gap+size+item
1 %	14032	1805	19.34	16.83	28862	47042	17580	16584
0.5 %	48990	6659	43.46	34.6	100736	163205	61149	58625
0.4 %	72228	10132	55.66	43.47	148597	240337	90477	87206
0.3 %	119965	17383	79.88	59.28	246934	398626	151280	146601
0.2 %	259760	39140	143.91	100.09	534816	861599	329185	321304
0.1 %	963053	153411	539.57	379.04	1986464	3186519	1236340	1219193

TABLE D.4 – `GAP-SEQ` under size and membership constraints on the PUBMED dataset.

D.5.4 GSPM : evaluating the scalability of `GAP-SEQ`

We used three datasets and replicated them from 1 to 20 times. The gap constraint was set to $gap[0, 9]$, and *minsup* to three different values. Fig. D.4 reports the CPU times according to the replication factor (i.e. dataset sizes). CPU times increase (almost) linearly as the number of sequences. This indicates that `GAP-SEQ` achieves scalability while it is a major issue for CP approaches. The behavior of `GAP-SEQ` on Protein is quite different for low values of *minsup*. Indeed, for large sequences (such as in Protein), the size of $\mathcal{ALLOCC}$ may be very large and thus checking the gap constraint becomes costly (see Sect. D.4.4).

D.5.5 GSPM : handling various additional constraints

To illustrate the flexibility of our approach, we selected the PubMed dataset and stated additional constraints such as minimum frequency, minimum size, and other useful constraints expressing some linguistic knowledge as membership. The goal is to extract sequential patterns which convey linguistic regularities (e.g., gene - rare disease relationships) [Béchet *et al.*, 2012]. The size constraint allows to forbid patterns that are too small w.r.t. the number of items (number of words) to be relevant patterns ; we set ℓ_{min} to 3. The membership constraint enables to filter out sequential patterns that do not contain some selected items. For example, we state that extracted patterns must contain at least the two items GENE and DISEASE. We used the $gap[0, 9]$ constraint, which is the best setting found in [Béchet *et al.*, 2012]. As no specialized method exists for this combination of constraints, we thus compare `GAP-SEQ` with and without additional constraints.

Table D.4 reports, for each value of *minsup*, the number of patterns extracted and the associated CPU times, the number of propagations and the number of nodes in the search tree. Additional constraints obviously restrict the number of extracted patterns. As the problem is more constrained, the size of the developed search tree is smaller. Even if the number of propagations is higher, the resulting CPU times are smaller. To conclude, thanks to the `GAP-SEQ` global constraint and its encoding, additional constraints like size, membership and regular expressions constraints can be easily stated.

D.5.6 Evaluating the ability of GAP-SEQ to efficiently handle SPM

In order to simulate the absence of gap constraints, we used the ineffective $gap[0, \ell]$ constraint (recall that ℓ is the size of the longest sequence of SDB). We compared `GAP-SEQ`$[0, \ell]$ with PREFIX-PROJECTION and two configurations of `cSpade` for SPM : `cSpade` without gap constraint

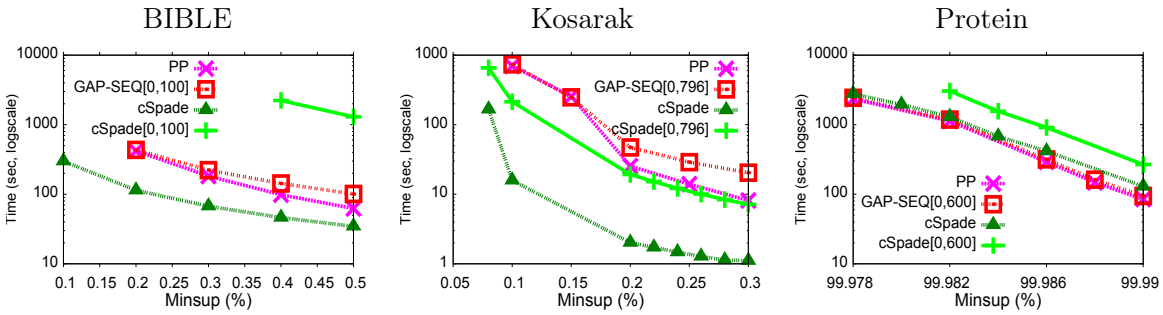


FIGURE D.5 – Comparing GAP-SEQ with PREFIX-PROJECTION and `cSpade` for SPM on BIBLE, Kosarak and Protein.

and `cSpade` with M and N set respectively to 0 and ℓ, denoted by `cSpade`$[0, \ell]$. Let us note that all the above methods will extract the same set of sequential patterns.

Fig. D.5 reports the CPU times for the four methods. First, `cSpade` obtains the best performance (except on Protein). These results confirm those observed in [Kemmar *et al.*, 2015]. Second, `GAP-SEQ`$[0, \ell]$ and PREFIX-PROJECTION exhibit similar behavior, even if `GAP-SEQ`$[0, \ell]$ is slightly less faster. So, even if `GAP-SEQ` handles both cases (with and without gap), it remains very competitive for SPM. Third, `GAP-SEQ`$[0, \ell]$ clearly outperforms `cSpade`$[0, \ell]$ (except on Kosarak). This is probably due to the huge number of unnecessary joining operations performed by `cSpade`$[0, \ell]$. To conclude, all the performed experiments demonstrate the ability of GAP-SEQ to efficiently handle SPM.

D.6 Conclusion

In this paper, we have introduced the global constraint GAP-SEQ enabling to handle SPM with or without gap constraints. The filtering algorithm benefits from the principle of right pattern extensions and prefix anti-monotonicity property of the gap constraint. GAP-SEQ enables to handle several types of constraints simultaneously and does not require any reified constraints nor any extra variables to encode the subsequence relation. Experiments performed on several real-life datasets (i) show that our approach clearly outperforms existing CP approaches as well as specialized methods for GSPM on large datasets, and (ii) demonstrate the ability of GAP-SEQ to efficiently handle SPM.

This work opens several issues for future researches. We plan to handle constraints on set of sequential patterns such as closedness, relevant subgroup and skypattern constraints.

Annexe E

Rayonnement et responsabilités

Organisation de conférences

— **Co-président du comité d'organisation** de JFPC 2010 (sixièmes Journées Francophones de Programmation par Contraintes) à Caen. Les JFPC sont le principal congrès des communautés francophones travaillant sur les CSP, les problèmes SAT et la programmation logique avec contraintes (CLP).

— **Membre du comité d'organisation** de CP-AI-OR'02 (Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems) au Croisic, France.

Distinctions

— Mehdi Maamar a obtenu le prix du meilleur article jeune chercheur à COSI'2015.

— La thèse de Abdelkader Ouali a été sélectionnée pour un financement en deuxième année par une bourse d'excellence scientifique Eiffel.

— Willy Ugarte a obtenu le Carl Smith award[51] du meilleur article jeune chercheur à DS'12.

— Mathieu Fontaine a obtenu le premier prix au Pascal Inference Challenge (PIC'11) sur l'inférence dans les réseaux bayésiens dans la catégorie CPU-1h. Invavtion à la conférence internationale UAI'2012.

Invitations

— Université d'Oran 1 (Algérie). Invité par Yahia Lebbah (Pr) pour un séjour *de 4 jours* en Juin 2008, Février 2012 et Mai 2014. Animation de groupes de travail sur la relaxation de contraintes globales, la fouille/PPC, séminaire sur l'extraction de motifs sous contraintes souples.

51. `http://eric.univ-lyon2.fr/ds-2012/awards.html`

— Spring Workshop on Mining and Learning[52] (SML 2014). Invité par Siegfried Nijssen (KU Leuven) pour présenter un poster sur "Soft Constraints for Pattern Set Mining".

Comités

— **Vice Président du Comité de Programme de COSI'2015** : Colloque sur l'Optimisation et les Systèmes d'Information, 1-3 juin 2015, Oran, Algérie.

— **Membre du Comité de Programme** : JFPC 2004, 2010 et 2011, HM'2014 (9th International Workshop on Hybrid Metaheuristics, Hamburg, Germany), COSI'2016, CNIA 2016 (Conférence Nationale en Intelligence Artificielle, RFIA'16).

— **Responsable d'une session** à ICTAI'14 - Special Track on SAT and CSP technologies.

— **Relecteur pour des revues internationales** : Int. J. of Comp. Appl. (2006), JMMA (2009), J. of Scheduling (2011), EJOR (2012), Int. J. of Bio-Inspi. Comp. (2014), Artificial Intelligence (2014), Constraints (2016).

— **Relecteur additionnel pour des conférences nationales et internationales** :
  — CP'2004, CP'2011, CP'2016 : International Conference on Principles and Practice of Constraint Programming,
  — SAC'2010, 25th Annual ACM Symposium on Applied Computing, track Constraint Solving and Programming ;
  — DS'2012, Discovery Science - 15th International Conference ;
  — ECAI'2014, 21st European Conference on Artificial Intelligence ;
  — ICTAI'2014, 26th International Conference on Tools with Artificial Intelligence ;
  — COSI'2014, Colloque sur l'Optimisation et les Systèmes d'Information ;
  — EGC'2015, 15ème conférence internationale sur l'extraction et la gestion des connaissances ;
  — IDA'2015, Fourteenth International Symposium on Intelligent Data Analysis ;
  — ICDM'2015, IEEE International Conference on Data Mining.
  — IJCAI'16, 25th International Joint Conference on Artificial Intelligence.
  — PKDD'2004, PKDD'2016 : European Conference on Machine Learning and Principles and Practice of Knowledge Discovery.

— **Rapporteur** pour le Prix de thèse AFIA 2013.

— **Expertise** du programme ANR 2013 JCJ SIMI 2.

— **Jury de thèse** : Membre du jury de thèse de Thi Hong Hiep, "Cohérences fortes pour les réseaux de fonctions de coûts pondérés", soutenue à l'INRA Toulouse le 15 Janvier 2015.

Diffusion logicielle

— Les méthodes développées par Mathieu Fontaine et Abdelkader Ouali sont disponibles et diffusées à la communauté scientifique via la plate-forme ToulBar2.

52. `https://dtai.cs.kuleuven.be/sml/`

— Le prototype du papier CP'15 (disponible à l'url : `https://sites.google.com/site/cp4spm/`) est déjà utilisé par plusieurs chercheurs : B. Negrevergne (Post-Doc, INSA Rennes), T. Guns (chercheur, KU Leuven), John Aoga (Phd, UCLouvain), Pierre Schaus (Ass. Prof., UCLouvain).

Responsabilités collectives

— **2006 – 2008** : membre titulaire de la commission de spécialistes, section 27, Université de Caen Basse-Normandie.

— **2005 – 2007** : membre suppléant de la commission de spécialistes, section 27, Université de Picardie Jules Vernes d'Amiens.

— **Depuis juin 2008** : membre élu du CA de l'AFPC : Association Française pour la Programmation par Contraintes (site : `www.afpc-asso.org`). Depuis juin 2010, **je suis trésorier** de l'association. Cette association regroupe plusieurs chercheurs Francophones et industriels. Elle organise une manifestation annuelle, les JFPC, qui regroupe près de 100 participants.

— **2011 – 2014** : responsable scientifique pour le GREYC du projet ANR **FICOLOFO** (*Filtrage par Cohérence Locales Fortes pour les réseaux de fonctions de coûts et autres modèles graphiques*).

— **2015** : vice-président du CoS pour le poste de MCF CNU 27 à l'UCBN.

— **Sept. 2016** (pour 3 ans) : Chef de département Informatique à l'IUT de Caen.

Responsabilités d'enseignement

— **Responsable des stages** au département R&T, IUT de Caen (2004 – 2007 & 2008 – 2009) (tâches : mise en relation des étudiants avec les entreprises, validation des sujets de stages, planning des soutenances).

— **Responsable des projets** au département R&T, IUT de Caen (**1A** : 2004 – 2005 & 2008 – 2009 ; **2A** : 2003 – 2005 & 2007 – 2008) (tâches : recensement des projets, répartition des étudiants sur les projets, affectation des tuteurs, planning des soutenances).

— **Directeur des études 2A** au département R&T, IUT de Caen (2007 – 2008) (suivi de 24 étudiants : diffusion d'information et conseil, gestion des absences, organisation du jury de fin de semestre, coordination de réunions pédagogiques entre enseignants et étudiants).

— **Responsable des EDT** au département R&T, IUT de Caen (Janv. 2010 – Juin 2010).

— **Responsable des stages** au département Informatique, IUT de Caen (depuis 2011) (contacts avec les entreprises pour définir et valider les sujets de stages, diffusion des offres et conseil auprès des étudiants, accompagnement des étudiants pour les stages à l'étranger coordination des encadrements de stages, planning des soutenances).

— **Responsable de la matière Algorithmique - Programmation** (depuis 2011) des deux années du DUT informatique (160 étudiants) : gestion et coordination des CM, des 6 groupes de TD et 12 groupes de TP ; coordination et recrutement de la douzaine d'intervenants (soit plus de 900 heures équivalent TD à gérer).

Encadrement de stages M2 à vocation recherche

— **Comparaison expérimentale des performances des consistances d'arcs appliquées à PFC-MRDAC pour la résolution des VCSP**, Sofiane Sahraoui, stage de recherche du DEA IMAG, GREYC, université de Caen, encadrement principal (100%), avril 2004 à septembre 2004.

— **Étude du routage après placement dans les circuits imprimés**, A. Hafid, stage de recherche du master AMI (Algorithmes et Modèles de l'Information), GREYC, université de Caen, co-encadrement (50%) avec Patrice Boizumault, avril 2005 à septembre 2005. Stage effectué dans le cadre du projet SaveCost.

— **Relaxation de contraintes globales : l'exemple de `AllDifferent`**, Jean-Philippe Métivier, stage de recherche du master AMI, GREYC, université de Caen, co-encadrement (50%) avec Patrice Boizumault, avril 2006 à septembre 2006.

— **Apports de la décomposition arborescente dans les méthodes de recherche locale à voisinages étendus (VNS)**, Mathieu Fontaine, stage de recherche du master AMI, GREYC, université de Caen, encadrement principal (100%), avril 2009 à septembre 2009.

— **Optimisation combinatoire, portfolios pour le problème SAT**, Ibrahim Abdoulahi, stage de recherche du master AMI, INRIA Saclay, co-encadrement avec Youssef Hamadi (Microsoft) et Michèle Sebag (INRIA Saclay), avril 2010 à septembre 2010.

— **Relaxation de contraintes pour l'extraction de motifs**, Willy Ugarte, stage de recherche du master AMI, GREYC, université de Caen, co-encadrement (50%) avec Patrice Boizumault, avril 2011 à septembre 2011.

— **Approche déclarative de découverte de motifs séquentiels fondée sur la programmation par contraintes**, Xin Huang, stage de recherche du master DECIM (décision et optimisation), GREYC, université de Caen, co-encadrement (50%) avec Patrice Boizumault, mars 2013 à août 2013.

— **Exploitation parallèle de la décomposition arborescente pour guider VNS**, Samir Hamitouche, stage de recherche du master IMALANG, GREYC, université de Caen, encadrement principal (100%), mars 2013 à août 2013.

— **Conception d'une contrainte globale de liaison aux données pour la fouille d'itemsets**, Yann Dauxais, stage de recherche du master DECIM, GREYC, université de Caen, co-encadrement (25%) avec Arnaud Lallouet (25%) et Patrice Boizumault (50%), mars 2014 à août 2014.

— **10) Stratégies parallèles pour DGVNS**, Abdelkader Ouali, stage de recherche du master DECIM, GREYC, université de Caen, encadrement principal (100%), mars 2014 à juillet 2014.

— **Navigation et compression dans un cube de skypatterns**, Maxime Houssin, stage de recherche du master DECIM, GREYC, université de Caen, co-encadrement (50%) avec Bruno Crémilleux, mars 2015 à août 2015.

Bibliographie

[Agrawal *et al.*, 1993] Agrawal, R., Imielinski, T. et Swami, A. N. (1993). Mining association rules between sets of items in large databases. *In* Buneman, P. et Jajodia, S., éditeurs : *SIGMOD Conference*, pages 207–216. ACM Press. 51

[Agrawal et Srikant, 1995] Agrawal, R. et Srikant, R. (1995). Mining sequential patterns. *In* Yu, P. S. et Chen, A. L. P., éditeurs : *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 3–14. IEEE Computer Society. 51, 56, 57, 107, 126

[Allouche *et al.*, 2015] Allouche, D., Bessiere, C., Boizumault, P., de Givry, S., Gutierrez, P., Lee, J. H. M., Leung, K. L., Loudni, S., Métivier, J., Schiex, T. et Wu, Y. (2015). Tractability and decompositions of global cost functions. *CoRR*, abs/1502.02414. 41

[Allouche *et al.*, 2012a] Allouche, D., Bessiere, C., Boizumault, P., De Givry, S., Gutierrez, P., Loudni, S., Métivier, J.-P. et Schiex, T. (2012a). Filtering Decomposable Global Cost Functions. *In AAAI'12 : Conference on Artificial Intelligence*, page 7, Toronto, Ontario, Canada. 19, 46, 47

[Allouche *et al.*, 2012b] Allouche, D., Bessière, C., Boizumault, P., De Givry, S., Gutierrez, P., Loudni, S., Métivier, J.-P. et Schiex, T. (2012b). Filtrage de fonctions de coût globales d écomposables. *In 8-èmes Journées Francophones de Programmation par Contraintes (JFPC 2012)*, Toulouse, France. 46

[auf'm Hofe, 2000] auf'm Hofe, H. M. (2000). Solving rostering tasks as constraint optimization. *In* Burke, E. K. et Erben, W., éditeurs : *PATAT*, volume 2079 de *Lecture Notes in Computer Science*, pages 191–212. Springer. 45

[Ayres *et al.*, 2002] Ayres, J., Flannick, J., Gehrke, J. et Yiu, T. (2002). Sequential pattern mining using a bitmap representation. *In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 429–435. ACM. 58, 109

[Béchet *et al.*, 2012] Béchet, N., Cellier, P., Charnois, T. et Crémilleux, B. (2012). Sequential pattern mining to discover relations between genes and rare diseases. *In CBMS*. 114, 117, 131, 135

[Beeri *et al.*, 1983] Beeri, C., Fagin, R., Maier, D. et M.Yannakakis (1983). On the desirability of acyclic database schemes. *Journal of the ACM*, 30:479–513. 47

[Beldiceanu et Contejean, 1994a] Beldiceanu, N. et Contejean, E. (1994a). Introducing global constraints in CHIP. *Journal of Mathematical and Computer Modelling*, 20(12):97–123. 46

[Beldiceanu et Contejean, 1994b] Beldiceanu, N. et Contejean, E. (1994b). Introducing global constraints in CHIP. *Journal of Mathematical and Computer Modelling*, 20(12):97–123. 108

[Bensana *et al.*, 1999] Bensana, E., Lemaître, M. et Verfaillie, G. (1999). Earth observation satellite management. *Constraints*, 4(3):293–299. 30

[Bessière, 2006] Bessière, C. (2006). *Constraint Propagation*, chapitre 3, pages 29–83. Foundations of Artificial Intelligence. Elseveir. 15

[Bessiere *et al.*, 2007] Bessiere, C., Hebrard, E., Hnich, B. et Walsh, T. (2007). The complexity of reasoning with global constraints. *Constraints*, 12(2):239–259. 42

[Bessière et Régin, 1997] Bessière, C. et Régin, J. (1997). Arc consistency for general constraint networks : Preliminary results. *In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pages 398–404. Morgan Kaufmann. 15

[Bessière et Régin, 1996] Bessière, C. et Régin, J.-C. (1996). MAC and combined heuristics : Two reasons to forsake FC (and CBJ ?) on hard problems. *In* Freuder, E. C., éditeur : *CP*, volume 1118 de *Lecture Notes in Computer Science*, pages 61–75. Springer. 13

[Böhm *et al.*, 2005] Böhm, K., Jensen, C. S., Haas, L. M., Kersten, M. L., Larson, P.-Å. et Ooi, B. C., éditeurs (2005). *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*. ACM. 149, 152

[Börzsönyi *et al.*, 2001] Börzsönyi, S., Kossmann, D. et Stocker, K. (2001). The skyline operator. *In* Georgakopoulos, D. et Buchmann, A., éditeurs : *ICDE*, pages 421–430. IEEE Computer Society. 65, 91, 99

[Boulicaut *et al.*, 2000] Boulicaut, J.-F., Bykowski, A. et Rigotti, C. (2000). Approximation of frequency queries by means of free-sets. *In* Zighed, D. A., Komorowski, H. J. et Zytkow, J. M., éditeurs : *PKDD*, volume 1910 de *Lecture Notes in Computer Science*, pages 75–85. Springer. 54

[Boutilier, 2009] Boutilier, C., éditeur (2009). *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*. 145, 150

[Cabon *et al.*, 1999] Cabon, B., De Givry, S., Lobjois, L., Schiex, T. et Warners, J. (1999). Radio link frequency assignment. *Constraints*, 4(1):79–89. 29

[Cambazard et Jussien, 2006] Cambazard, H. et Jussien, N. (2006). Identifying and exploiting problem structures using explanation-based constraint programming. *Constraints*, 11(4):295–313. 13

[Cooper, 2003] Cooper, M. C. (2003). Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets Syst.*, 134(3):311–342. 16, 17

[Cooper *et al.*, 2008] Cooper, M. C., de Givry, S., Sánchez, M., Schiex, T. et Zytnicki, M. (2008). Virtual arc consistency for weighted CSP. *In AAAI*, pages 253–258. 17

[Coquery *et al.*, 2012] Coquery, E., Jabbour, S., Saïs, L. et Salhi, Y. (2012). A sat-based approach for discovering frequent, closed and maximal patterns in a sequence. *In ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pages 258–263. 105, 109

[Dao *et al.*, 2015] Dao, T.-B.-H., Lesaint, W. et Vrain, C. (2015). Clustering conceptuel et relationnel en programmation par contraintes. *In JFPC 2015*, Bordeaux, France. 75

[de Givry, 2011] de Givry, S. (2011). *Rapport d'habilitation à diriger des recherches.* Thèse de doctorat, INRA UBIA Toulouse. Mémoire d'HDR. 17

[de Givry *et al.*, 2005] de Givry, S., Heras, F., Zytnicki, M. et Larrosa, J. (2005). Existential arc consistency : Getting closer to full arc consistency in weighted CSPs. *In* Kaelbling, L. P. et Saffiotti, A., éditeurs : *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 84–89. Professional Book Center. 17

[de Givry *et al.*, 2003] de Givry, S., Larrosa, J., Meseguer, P. et Schiex, T. (2003). Solving max-sat as weighted CSP. *In* Rossi, F., éditeur : *CP*, volume 2833 de *Lecture Notes in Computer Science*, pages 363–376. Springer. 26

[De Raedt *et al.*, 2008] De Raedt, L., Guns, T. et Nijssen, S. (2008). Constraint programming for itemset mining. *In* Li, Y., Liu, B. et Sarawagi, S., éditeurs : *KDD*, pages 204–212. ACM. 55, 73, 80

[Debruyne et Bessière, 2001] Debruyne, R. et Bessière, C. (2001). Domain filtering consistencies. *J. Artif. Intell. Res. (JAIR)*, 14:205–230. 13

[Demassey *et al.*, 2006] Demassey, S., Pesant, G. et Rousseau, L.-M. (2006). A cost-regular based hybrid column generation approach. *Constraints*, 11(4):315–333. 45

[Dong et Pei, 2007] Dong, G. et Pei, J. (2007). *Sequence Data Mining*, volume 33 de *Advances in Database Systems*. Kluwer. 57

[Fargier et Lang, 1993] Fargier, H. et Lang, J. (1993). Uncertainty in constraint satisfaction problems : a probalistic approach. *In* Clarke, M., Kruse, R. et Moral, S., éditeurs : *ECSQARU*, volume 747 de *Lecture Notes in Computer Science*, pages 97–104. Springer. 11

[Fontaine, 2013] Fontaine, M. (2013). *Apports de la décomposition arborescente pour les méthodes de type VNS.* Thèse de doctorat, Université de Caen Basse-Normandie, France. Rapporteurs : C. Solnon, G. Verfaillie. Examinateurs : P. Jégou, L. Sais, B. Neveu. Directeur : P. Boizumault. Co-encadrant : S. Loudni. 6, 30

[Fontaine *et al.*, 2011] Fontaine, M., Loudni, S. et Boizumault, P. (2011). Guiding VNS with Tree Decomposition. *In Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 505 – 512, boca raton, Floride, United States. IEEE. 32

[Fontaine *et al.*, 2012] Fontaine, M., Loudni, S. et Boizumault, P. (2012). Exploitation de la décomposition arborescente pour guider la recherche VNS. *In 8-èmes Journées Francophones de Programmation par Contraintes (JFPC 2012)*, toulouse, France. 32

[Fontaine *et al.*, 2013a] Fontaine, M., Loudni, S. et Boizumault, P. (2013a). Exploiting Tree Decomposition for Guiding Neighborhoods Exploration for VNS. *RAIRO-Operations Research*, 47(2):pp91–123. 32

[Fontaine *et al.*, 2013b] Fontaine, M., Loudni, S. et Boizumault, P. (2013b). Raffinement de la décomposition arborescente par fusion de clusters pour guider DGVNS. *In 9-èmes Journées Francophones de Programmation par Contraintes (JFPC'13)*, pages 133–142, aix-en-provence, France. 34, 35

[Ford et Fulkerson, 1956] Ford, L. R. et Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404. 43

[Fournier-Viger *et al.*, 2014] Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C. et Tseng, V. (2014). SPMF : An Open-Source Data Mining Library in Java. *J. of Machine Learning Resea.*, 15:3389–3393. 114, 131

[Freuder et Wallace, 1992] Freuder, E. C. et Wallace, R. J. (1992). Partial constraint satisfaction. *Artif. Intell.*, 58(1-3):21–70. 11

[Frost et Dechter, 1995] Frost, D. et Dechter, R. (1995). Look-ahead value ordering for constraint satisfaction problems. *In IJCAI (1)*, pages 572–578. 13

[Fu *et al.*, 2000] Fu, A. W.-C., w. Kwong, R. W. et Tang, J. (2000). Mining n-most interesting itemsets. *In* Ras, Z. W. et Ohsuga, S., éditeurs : *ISMIS*, volume 1932 de *Lecture Notes in Computer Science*, pages 59–67. Springer. 56

[Garofalakis *et al.*, 2002] Garofalakis, M. N., Rastogi, R. et Shim, K. (2002). Mining sequential patterns with regular expression constraints. *IEEE Trans. Knowl. Data Eng.*, 14(3):530–552. 58, 107, 109

[Geng et Hamilton, 2006] Geng, L. et Hamilton, H. J. (2006). Interestingness measures for data mining : A survey. *ACM Comput. Surv.*, 38(3). 53

[Glover, 1986] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & OR*, 13(5):533–549. 20

[Glover et Laguna, 1997] Glover, F. et Laguna, F. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA. 34

[Golomb et Baumert, 1965] Golomb, S. W. et Baumert, L. D. (1965). Backtracking programming. *Journal of the Association for computing Machinery*, 12:516–524. 13

[Gouda et Zaki, 2005] Gouda, K. et Zaki, M. J. (2005). Genmax : An efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.*, 11(3):223–242. 54

[Guns *et al.*, 2011] Guns, T., Nijssen, S. et De Raedt, L. (2011). Itemset mining : A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983. 98, 109

[Guns *et al.*, 2013] Guns, T., Nijssen, S. et De Raedt, L. (2013). k-pattern set mining under constraints. *IEEE Trans. Knowl. Data Eng.*, 25(2):402–418. 75, 76, 81

[Han *et al.*, 2002] Han, J., Wang, J., Lu, Y. et Tzvetkov, P. (2002). Mining top-k frequent closed patterns without minimum support. *In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 211–218. IEEE Computer Society. 56

[Hand, 2002] Hand, D. J. (2002). Pattern detection and discovery. *In* Hand, D. J., Adams, N. M. et Bolton, R. J., éditeurs : *Pattern Detection and Discovery*, volume 2447 de *Lecture Notes in Computer Science*, pages 1–12. Springer. 53

[Hansen *et al.*, 2009] Hansen, K., Mika, S., Schroeter, T., Sutter, A., ter Laak, A., Steger-Hartmann, T., Heinrich, N. et Müller, K.-R. (2009). Benchmark data set for in silico prediction of ames mutagenicity. *Journal of Chemical Information and Modeling*, 49(9):2077–2081. 70, 100

[Hansen *et al.*, 2001] Hansen, P., Mladenović, N. et Perez-Britos, D. (2001). Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350. 22

[Hao *et al.*, 1999] Hao, J.-K., Galinier, P. et Habib, M. (1999). Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'Intelligence Artificielle*, 13:1–39. 20

[Harvey et Ginsberg, 1995] Harvey, W. D. et Ginsberg, M. L. (1995). Limited discrepancy search. *In IJCAI (1)*, pages 607–615. 14

[Hentenryck *et al.*, 1992] Hentenryck, P. V., Simonis, H. et Dincbas, M. (1992). Constraint satisfaction using constraint logic programming. *Artif. Intell.*, 58(1-3):113–159. 43

[Hirschhorn et Daly, 2005] Hirschhorn, J. et Daly, M. (2005). Genome-wide association studies for common diseases and complex traits. *Nature Reviews Genetics*, 6(2):95–108. 30

[Hoeve *et al.*, 2006] Hoeve, W. J. V., Pesant, G. et Rousseau, L.-M. (2006). On global warming : Flow-based soft global constraints. *J. Heuristics*, 12(4-5):347–373. 42

[Jégou *et al.*, 2006] Jégou, P., Ndiaye, S. et Terrioux, C. (2006). Strategies and Heuristics for Exploiting Tree-decompositions of Constraint Networks. *In Inference methods based on graphical structures of knowledge (WIGSK'06)*, pages 13–18. 36

[Ji *et al.*, 2005] Ji, X., Bailey, J. et Dong, G. (2005). Mining minimal distinguishing subsequence patterns with gap constraints. *In Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 194–201. IEEE Computer Society. 121, 122, 126

[Jones et Harrold, 2005] Jones, J. A. et Harrold, M. J. (2005). Empirical evaluation of the tarantula automatic fault-localization technique. *In 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), November 7-11, 2005, Long Beach, CA, USA*, pages 273–282. 74

[Kemmar *et al.*, 2015] Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P. et Charnois, T. (2015). PREFIX-PROJECTION Global Constraint for Sequential Pattern Mining. *In Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, volume 9255 de *Lecture Notes in Computer Science*, pages 226–243. Springer. 72, 105, 121, 122, 126, 136

[Kemmar *et al.*, 2016] Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P. et Charnois, T. (2016). A global constraint for mining sequential patterns with GAP constraint. *In* Quimper, C., éditeur : *Integration of AI and OR Techniques in Constraint Programming - 13th International Conference, CPAIOR 2016, Banff, AB, Canada, May 29 - June 1, 2016, Proceedings*, volume 9676 de *Lecture Notes in Computer Science*, pages 198–215. Springer. 72, 121

[Kemmar *et al.*, 2014] Kemmar, A., Ugarte, W., Loudni, S., Charnois, T., Lebbah, Y., Boizumault, P. et Crémilleux, B. (2014). Mining Relevant Sequence Patterns with CP-Based Framework. *In 26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, November 10-12, 2014*, pages 552–559. IEEE Computer Society. 71, 105, 109, 122, 126

[Khiari *et al.*, 2010] Khiari, M., Boizumault, P. et Crémilleux, B. (2010). Constraint programming for mining n-ary patterns. *In* Cohen, D., éditeur : *CP*, volume 6308 de *Lecture Notes in Computer Science*, pages 552–567. Springer. 81, 100

[Kirkpatrick *et al.*, 1983] Kirkpatrick, S., Jr., D. G. et Vecchi, M. P. (1983). Optimization by simmulated annealing. *Science*, 220(4598):671–680. 20

[Kitching et Bacchus, 2009] Kitching, M. et Bacchus, F. (2009). Exploiting decomposition on constraint problems with high tree-width. *In* [Boutilier, 2009], pages 525–531. 35

[Larrosa, 2002] Larrosa, J. (2002). Node and arc consistency in weighted CSP. *In* Dechter, R. et Sutton, R. S., éditeurs : *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada.*, pages 48–53. AAAI Press / The MIT Press. 11, 16, 18

[Larrosa et Schiex, 2003] Larrosa, J. et Schiex, T. (2003). In the quest of the best form of local consistency for weighted CSP. *In* Gottlob, G. et Walsh, T., éditeurs : *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 239–244. Morgan Kaufmann. 16, 17, 19, 26

[Lazaar *et al.*, 2016] Lazaar, N., Lebbah, Y., Loudni, S., Maamar, M., Lemiere, V., Bessière, C. et Boizumault, P. (2016). A Global Constraint for Closed Itemset Mining. *In Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, 5-9 Sptember, 2016, Proceedings*, pages 1–16. Springer. 80

[Lee et Leung, 2012] Lee, J. H. et Leung, K. L. (2012). Consistency techniques for flow-based projection-safe global cost functions in weighted constraint satisfaction. *J. Artif. Intell. Res. (JAIR)*, 43:257–292. 46

[Lee et Leung, 2009] Lee, J. H.-M. et Leung, K. L. (2009). Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. *In Proceedings of the 21st international jont conference on Artifical intelligence*, pages 559–565. Morgan Kaufmann Publishers Inc. 19

[Levasseur, 2008] Levasseur, N. (2008). *Heuristiques de recherche pour la résolution des WCSP*. Thèse de doctorat, Université de Caen Basse-Normandie, France. Rapporteurs : J.-K. Hao, T. Schiex. Examinateurs : A. Lallouet, B. Neveu, C. Solnon. Directeur : P. Boizumault. Co-encadrant : S. Loudni. 6, 13

[Levasseur *et al.*, 2007] Levasseur, N., Boizumault, P. et Loudni, S. (2007). A Value Ordering Heuristic for Weighted CSP. *In 19-th IEEE International Conference on Tools with Artificial Intelligence, (ICTAI'07)*, pages 259–262, Patras, Greece. 25

[Levasseur *et al.*, 2008a] Levasseur, N., Boizumault, P. et Loudni, S. (2008a). Boosting VNS with Neighborhood Heuristics for Solving Constraint Optimization Problems. *In 5-th International Workshop on Hybrid Metaheuristics (HM'08)*, volume 5296 de *LNCS*, pages 131–145, Malaga, Spain. Springer-Verlag. 28

[Levasseur *et al.*, 2008b] Levasseur, N., Boizumault, P. et Loudni, S. (2008b). Heuristique de choix de valeur dirigée par h-quality dans les WCSP. *In 4-èmes Journées Francophones de Programmation par Contraintes (JFPC'08)*, pages 257–266, Nantes. 25, 26

[Levasseur *et al.*, 2008c] Levasseur, N., Boizumault, P. et Loudni, S. (2008c). Heuristiques de choix de voisinage pour les recherches à voisinages variables dans les WCSP. *In 4-èmes Journées Francophones de Programmation par Contraintes (JFPC'08)*, pages 247–256, Nantes. 29

[Li et Wang, 2008] Li, C. et Wang, J. (2008). Efficiently mining closed subsequences with gap constraints. *In Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, pages 313–322. SIAM. 126

[Li *et al.*, 2012] Li, C., Yang, Q., Wang, J. et Li, M. (2012). Efficient mining of gap-constrained subsequences and its various applications. *ACM Trans. Knowl. Discov. Data*, 6(1):2 :1–2 :39. 109, 122

[Loudni et Boizumault, 2003] Loudni, S. et Boizumault, P. (2003). Solving Constraint Optimization Problems in *Anytime* Context. *In 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 251–256, Acapulco, Mexique. 1, 22

[Loudni et Boizumault, 2008] Loudni, S. et Boizumault, P. (2008). Combining VNS with constraint programming for solving anytime optimization problems. *EJOR*, 191:705–735. 22

[Loudni *et al.*, 2006] Loudni, S., Boizumault, P. et David, P. (2006). On-line Resources Allocation for ATM Networks with Rerouting. *Journal of Computers & Operations Research*, 33(11):2891–2917. 1

[Loudni *et al.*, 2010] Loudni, S., Boizumault, P. et Levasseur, N. (2010). Advanced Generic Neighorhoood Heuristics for VNS. *Journal of Engineering Applications of Artificial Intelligence (EAAI)*, 23(5):Pages 736–764. 28, 29

[Loudni *et al.*, 2012] Loudni, S., Fontaine, M. et Boizumault, P. (2012). Exploiting Separators for Guiding VNS. *Electronic Notes in Discrete Mathematics (39)*, pages 265–272. 37

[Loudni *et al.*, 2013a] Loudni, S., Fontaine, M. et Boizumault, P. (2013a). DGVNS guidée par les séparateurs. *In 9-èmes Journées Francophones de Programmation par Contraintes (JFPC'13)*, pages 205–214, aix-en-provence, France. 37

[Loudni *et al.*, 2013b] Loudni, S., Fontaine, M. et Boizumault, P. (2013b). Intensification/-Diversification in Decomposition Guided VNS. *In 8th International Workshop on Hybrid MetaHeuristics (HM'13)*, volume 7919, pages pp22–36, Napoli, Italy. Springer Berlin / Heidelberg. 32, 33

[Maamar *et al.*, 2016] Maamar, M., Laazar, N., Loudni, S. et Lebbah, Y. (2016). F-CPminer : Une approche pour la localisation de fautes basée sur l'extraction de motifs ensemblistes sous contraintes. *In In* 12-èmes Journées Francophones de Programmation par Contraintes (JFPC'16), Montpelier, Juin 2016, pages 1–10. 73

[Maamar *et al.*, 2015] Maamar, M., Lazaar, N., Loudni, S. et Lebbah, Y. (2015). Fault Localization Using Itemset Mining under Constraints. *Automated Software Engineering*, pages 1–28. 73

[Mackworth, 1977] Mackworth, A. K. (1977). Consistency in networks of relations. *Artif. Intell.*, 8(1):99–118. 15

[Mannila et Toivonen, 1997] Mannila, H. et Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258. 92

[Métivier, 2010] Métivier, J.-P. (2010). *Relaxation de contraintes globales.* Thèse de doctorat, Université de Caen Basse-Normandie, France. Rapporteurs : C. Bessière, N. Jussien. Examinateurs : E. Grandjean, S. de Givry, A. Lallouet, T. Petit. Directeur : P. Boizumault. Co-encadrant : S. Loudni. 6

[Metivier *et al.*, 2012] Metivier, J.-P., Boizumault, P., Crémilleux, B., Khiari, M. et Loudni, S. (2012). Constrained Clustering using SAT. *In 11th Int. Symposium on Intelligent Data Analysis (IDA 2012)*, pages 207–218, Helsinki, Finland. 75

[Métivier *et al.*, 2009a] Métivier, J.-P., Boizumault, P. et Loudni, S. (2009a). Softening `Gcc` and `Regular` with Preferences. *In 24th annual ACM Symposium on Applied Computing (SAC'09)*, pages 1392–1396, University of Hawaii at Manoa, USA. 42

[Métivier *et al.*, 2009b] Métivier, J.-P., Boizumault, P. et Loudni, S. (2009b). Solving Nurse Rostering Problems Using Soft Global Constraints. *In 15th International Conference on Prin-*

ciples and Practice of Constraint Programming (CP'2009), volume 5732 de *LNCS*, pages 73–87, Lisbon, Portugal. Springer-Verlag. 45

[Métivier *et al.*, 2013] Métivier, J.-P., Loudni, S. et Charnois, T. (2013). A Constraint Programming Approach for Mining Sequential Patterns in a Sequence Database. *In Int. Workshop Languages for Data Mining and Machine Learningco-located with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2013)*, pages p.50–64, Prague, Czech Republic. 70

[Métivier *et al.*, 2013] Métivier, J.-P., Loudni, S. et Charnois, T. (2013). A constraint programming approach for mining sequential patterns in a sequence database. *In ECML/PKDD Workshop on Languages for Data Mining and Machine Learning*. 105, 109, 122, 126

[Metivier *et al.*, 2014] Metivier, J.-P., Loudni, S. et Charnois, T. (2014). Une approche PPC pour la fouille de données séquentielles. *In 14èmes Journées Francophones Extraction et Gestion des Connaissances, EGC 2014*, volume RNTI-E-26, pages pp.395–400, rennes, France. 70

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artif. Intell.*, 18(2):203–226. 53

[Mladenovic et Hansen, 1997] Mladenovic, N. et Hansen, P. (1997). Variable neighborhood search. *Computers And Operations Research*, 24:1097–1100. 21

[Négrevergne et Guns, 2015] Négrevergne, B. et Guns, T. (2015). Constraint-based sequence mining using constraint programming. *In* Michel, L., éditeur : *Integration of AI and OR Techniques in Constraint Programming - 12th International Conference, CPAIOR 2015, Barcelona, Spain, May 18-22, 2015, Proceedings*, volume 9075 de *Lecture Notes in Computer Science*, pages 288–305. Springer. 71, 105, 109, 110, 111, 115, 122, 126

[Nessa *et al.*, 2008] Nessa, S., Abedin, M., Wong, W. E., Khan, L. et Qi, Y. (2008). Software fault localization using n-gram analysis. *In Wireless Algorithms, Systems, and Applications*, pages 548–559. Springer. 74

[Neveu *et al.*, 2004] Neveu, B., Trombettoni, G. et Glover, F. (2004). ID walk : A candidate list strategy with a simple diversification device. *In* [Wallace, 2004], pages 423–437. 34

[Novak *et al.*, 2009] Novak, P. K., Lavrac, N. et Webb, G. I. (2009). Supervised descriptive rule discovery : A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403. 53, 109

[Ouali *et al.*, 2016a] Ouali, A., Loudni, S., Lebbah, Y., Boizumault, P., Zimmermann, A. et Loukil, L. (2016a). Clustering conceptuel en PLNE. *In In* 12-èmes Journées Francophones de Programmation par Contraintes (JFPC'16), Montpelier, Juin 2016, pages 1–10. 74

[Ouali *et al.*, 2016b] Ouali, A., Loudni, S., Lebbah, Y., Boizumault, P., Zimmermann, A. et Loukil, L. (2016b). Efficiently Finding Conceptual Clustering Models with Integer Linear Programming. *In Proceedings of the Twenty-Five International Joint Conference on Artificial Intelligence (IJCAI 2016), New York, USA, July 2016*, pages 1–8. AAAI Press. 74, 76

[Ouali *et al.*, 2014] Ouali, A., Loudni, S., Loukil, L., Boizumault, P. et Lebbah, Y. (2014). Cooperative Parallel Decomposition Guided VNS for Solving Weighted CSP. *In 9th Int. Workshop on Hybrid MetaHeuristics (HM 14)*, volume 8457, pages pp 100–114, hamburg, Germany. Springer International Publishing. 37, 38

[Ouali *et al.*, 2015] Ouali, A., Loudni, S., Loukil, L., Boizumault, P. et Lebbah, Y. (2015). Replicated Parallel Strategies for Decomposition Guided VNS. *Electronic Notes in Discrete Mathematics*, 47:93–100. 37, 38

[Pasquier *et al.*, 1999] Pasquier, N., Bastide, Y., Taouil, R. et Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. *In* Beeri, C. et Buneman, P., éditeurs : *ICDT*, volume 1540 de *Lecture Notes in Computer Science*, pages 398–416. Springer. 54

[Pei *et al.*, 2007] Pei, J., Fu, A. W.-C., Lin, X. et Wang, H. (2007). Computing compressed multidimensional skyline cubes efficiently. *In* Chirkova, R., Dogac, A., Özsu, M. T. et Sellis, T. K., éditeurs : *ICDE*, pages 96–105. IEEE. 100

[Pei *et al.*, 2001] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. et Hsu, M. (2001). Prefixspan : Mining sequential patterns by prefix-projected growth. *In* Georgakopoulos, D. et Buchmann, A., éditeurs : *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 215–224. IEEE Computer Society. 58, 59, 106, 108, 109, 126

[Pei *et al.*, 2002] Pei, J., Han, J. et Wang, W. (2002). Mining sequential patterns with constraints in large databases. *In Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 4-9, 2002*, pages 18–25. ACM. 105, 107, 109, 124, 126

[Pei *et al.*, 2005] Pei, J., Jin, W., Ester, M. et Tao, Y. (2005). Catching the best views of skyline : A semantic approach based on decisive subspaces. *In* [Böhm *et al.*, 2005], pages 253–264. 100

[Pei *et al.*, 2006] Pei, J., Yuan, Y., Lin, X., Jin, W., Ester, M., Liu, Q., Wang, W., Tao, Y., Yu, J. X. et Zhang, Q. (2006). Towards multidimensional subspace skyline analysis. *ACM Trans. Database Syst.*, 31(4):1335–1381. 100

[Pensa *et al.*, 2005] Pensa, R. G., Robardet, C. et Boulicaut, J. (2005). A bi-clustering framework for categorical data. *In PKDD 2005*, pages 643–650. 75, 76

[Perkowitz et Etzioni, 1999] Perkowitz, M. et Etzioni, O. (1999). Adaptive web sites : Conceptual cluster mining. *In IJCAI 99*, pages 264–269. 75, 76

[Perron *et al.*, 2004] Perron, L., Shaw, P. et Furnon, V. (2004). Propagation guided large neighborhood search. *In* [Wallace, 2004], pages 468–481. 28

[Pesant, 2004a] Pesant, G. (2004a). A regular language membership constraint for finite sequences of variables. *In* [Wallace, 2004], pages 482–495. 44

[Pesant, 2004b] Pesant, G. (2004b). A regular language membership constraint for finite sequences of variables. *In* Wallace, M., éditeur : *CP'04*, volume 2239 de *LNCS*, pages 482–495. Springer. 108

[Petit, 2002] Petit, T. (2002). *Modélisation et Algorithmes de Résolution de Problèmes Sur-Contraints*. Thèse de doctorat, LIRMM - Université de Montpellier II. 63

[Petit *et al.*, 2001] Petit, T., Régin, J.-C. et Bessière, C. (2001). Specific filtering algorithms for over-constrained problems. *In* Walsh, T., éditeur : *CP*, volume 2239 de *Lecture Notes in Computer Science*, pages 451–463. Springer. 41, 42

[Qu et He, 2008] Qu, R. et He, F. (2008). A hybrid constraint programming approach for nurse rostering problems. *In The Twenty-eighth SGAI International Conference on Artificial Intelligence*, pages 211–224. 45

[Raïssi *et al.*, 2010] Raïssi, C., Pei, J. et Kister, T. (2010). Computing closed skycubes. *PVLDB*, 3(1):838–847. 91, 100

[Refalo, 2004] Refalo, P. (2004). Impact-based search strategies for constraint programming. *In* [Wallace, 2004], pages 557–571. 13

[Régin, 1996] Régin, J.-C. (1996). Generalized arc consistency for global cardinality constraint. *In AAAI/IAAI, Vol. 1*, pages 209–215. 42, 43

[Robertson et Seymour, 1986] Robertson, N. et Seymour, P. (1986). Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322. 30

[Rosenfeld *et al.*, 1976] Rosenfeld, A., Hummel, R. A. et Zucker, S. W. (1976). Scene labeling by relaxation operations. *Systems, Man and Cybernetics, IEEE Transactions on*, 6(6):420–433. 11

[Ruttkay, 1994] Ruttkay, Z. (1994). Fuzzy constraint satisfaction. *In In Proc. 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268. 11

[Sabin et Freuder, 1994] Sabin, D. et Freuder, E. (1994). Contradicting conventional wisdom in constraint satisfaction. *In* Borning, A., éditeur : *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP-94)*, volume 874 de *Lecture Notes in Computer Science*. Springer. 13

[Sánchez *et al.*, 2009] Sánchez, M., Allouche, D., de Givry, S. et Schiex, T. (2009). Russian doll search with tree decomposition. *In* [Boutilier, 2009], pages 603–608. 30, 36

[Sanchez *et al.*, 2008] Sanchez, M., De Givry, S. et Schiex, T. (2008). Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints*, 13(1-2):130–154. 19

[Schiex, 1992] Schiex, T. (1992). Possibilistic constraint satisfaction problems or "How to handle soft constraints ?". *In* Dubois, D. et Wellman, M. P., éditeurs : *UAI*, pages 268–275. Morgan Kaufmann. 11

[Schiex, 2000] Schiex, T. (2000). Arc consistency for soft constraints. *In* Dechter, R., éditeur : *CP*, volume 1894 de *Lecture Notes in Computer Science*, pages 411–424. Springer. 16, 18

[Schiex *et al.*, 1995] Schiex, T., Fargier, H. et Verfaillie, G. (1995). Valued constraint satisfaction problems : Hard and easy problems. *In IJCAI (1)*, pages 631–639. 11, 15

[Shapiro et Haralick, 1981] Shapiro, L. et Haralick, R. (1981). Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:504–519. 11

[Soulet *et al.*, 2011] Soulet, A., Raïssi, C., Plantevit, M. et Crémilleux, B. (2011). Mining dominant patterns in the sky. *In* Cook, D. J., Pei, J., Wang, W., Zaïane, O. R. et Wu, X., éditeurs : *ICDM*, pages 655–664. IEEE. 65, 66, 67, 91, 93, 100

[Srikant et Agrawal, 1996] Srikant, R. et Agrawal, R. (1996). Mining sequential patterns : Generalizations and performance improvements. *In* Apers, P. M. G., Bouzeghoub, M. et Gardarin, G., éditeurs : *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, volume 1057 de *Lecture Notes in Computer Science*, pages 3–17. Springer. 58, 109

[Tarjan, 1972] Tarjan, R. E. (1972). Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160. 43

[Tarjan et Yannakakis, 1984] Tarjan, R. E. et Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579. 32

[Trasarti *et al.*, 2008] Trasarti, R., Bonchi, F. et Goethals, B. (2008). Sequence mining automata : A new technique for mining frequent sequences under regular expressions. *In Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 1061–1066. IEEE Computer Society. 109, 114, 115, 119

[Ugarte, 2014] Ugarte, W. (2014). *Extraction de motifs sous contraintes souples.* Thèse de doctorat, Université de Caen Basse-Normandie, France. Rapporteurs : J.-F. Boulicaut, L. Sais. Examinateurs : C. Vrain, C. Raissi. Directeurs : P. Boizumault, B. Crémilleux. Co-encadrant : S. Loudni. 6, 56

[Ugarte *et al.*, 2015a] Ugarte, W., Boizumault, P., Crémilleux, B., Lepailleur, A., Loudni, S., Plantevit, M., Raïssi, C. et Soulet, A. (2015a). Skypattern Mining : From Pattern Condensed Representations to Dynamic Constraint Satisfaction Problems. *Artificial Intelligence*, pages 1–22. 65

[Ugarte *et al.*, 2012a] Ugarte, W., Boizumault, P., Loudni, S. et Crémilleux, B. (2012a). Soft Threshold Constraints for Pattern Mining. *In 15th Int. Conf. on Discovery Science (DS 2012)*, pages 313–327, Lyon, France. Carl Smith Award Winner DS 2102, Springer. 61

[Ugarte *et al.*, 2014a] Ugarte, W., Boizumault, P., Loudni, S. et Crémilleux, B. (2014a). Computing Skypattern Cubes Using Relaxation. *In 26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, November 10-12, 2014*, pages 859–866. IEEE Computer Society. 69, 70

[Ugarte *et al.*, 2014b] Ugarte, W., Boizumault, P., Loudni, S. et Crémilleux, B. (2014b). Mining Skypattern Cubes. *In 21st Europ. Conf. on Artificial Intelligence, (ECAI'14)*, pages p1–6, Prague, Czech Republic. 69, 70, 91

[Ugarte *et al.*, 2013] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2013). Découverte des soft-skypatterns avec une approche PPC. *In Actes des 13ème Journées Francophones Extraction et Gestion de Connaissances (EGC 2013)*, pages 217–228, toulouse, France. 65

[Ugarte *et al.*, 2014c] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2014c). Mining (Soft-) Skypatterns using Dynamic CSP. *In 11th Int. Conf. on Integration of Artificial Intelligence (AI) and Operations Research (OR) techniques in Constraint Programming (CP-AI-OR'14)*, volume 8451, pages 71–87, Cork, Irlande. 65, 69, 91, 98, 99, 100

[Ugarte *et al.*, 2015b] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2015b). Mining (Soft-) Skypatterns using Constraint Programming. *In Advances in Knowledge Discovery and Management AKDM 5 (Post-EGC'13 Selected Papers)*, Studies in Computational Intelligence, pages 105–137. Springer. 65

[Ugarte *et al.*, 2015c] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2015c). Soft Constraints for Pattern Mining. *Journal of Intelligent Information Systems (JIIS)*, 44(2):193–221. 61

[Ugarte *et al.*, 2012b] Ugarte, W., Crémilleux, B., Loudni, S. et Boizumault, P. (2012b). Extraction de motifs sous contraintes souples de seuil. *In 8-èmes Journées Francophones de Programmation par Contraintes (JFPC 2012)*, Toulouse, France. 61

[Uno *et al.*, 2004] Uno, T., Asai, T., Uchida, Y. et Arimura, H. (2004). An efficient algorithm for enumerating closed patterns in transaction databases. *In* Suzuki, E. et Arikawa, S., éditeurs : *Discovery Science*, volume 3245 de *Lecture Notes in Computer Science*, pages 16–31. Springer. 75

[Verfaillie et Jussien, 2005] Verfaillie, G. et Jussien, N. (2005). Constraint solving in uncertain and dynamic environments : A survey. *Constraints*, 10(3):253–281. 98

[Wallace, 2004] Wallace, M., éditeur (2004). *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 de *Lecture Notes in Computer Science*. Springer. 148, 149, 150

[Wu *et al.*, 2013] Wu, X., Zhu, X., He, Y. et Arslan, A. N. (2013). PMBC : pattern mining from biological sequences with wildcard constraints. *Comp. in Bio. and Med.*, 43(5):481–492. 121

[Yan *et al.*, 2003] Yan, X., Han, J. et Afshar, R. (2003). Clospan : Mining closed sequential patterns in large databases. *In* Barbará, D. et Kamath, C., éditeurs : *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*, pages 166–177. SIAM. 109

[Yang, 2006] Yang, G. (2006). Computational aspects of mining maximal frequent patterns. *Theor. Comput. Sci.*, 362(1-3):63–85. 112, 128

[Yannakakis, 1981] Yannakakis, M. (1981). Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2:77–86. 32

[Yuan *et al.*, 2005] Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J. X. et Zhang, Q. (2005). Efficient computation of the skyline cube. *In* [Böhm *et al.*, 2005], pages 241–252. 100

[Zaki, 2000] Zaki, M. J. (2000). Sequence mining in categorical domains : Incorporating constraints. *In Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 6-11, 2000*, pages 422–429. ACM. 109, 121, 126

[Zaki, 2001] Zaki, M. J. (2001). SPADE : an efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60. 58, 109, 126

[Zhang *et al.*, 2007] Zhang, M., Kao, B., Cheung, D. W. et Yip, K. Y. (2007). Mining periodic patterns with gap requirement from sequences. *TKDD*, 1(2). 121, 122

[Zimmermann, 2009] Zimmermann, A. (2009). *Mining Sets of Patterns.* Thèse de doctorat, University Freiburg, Germany. 56