Modélisation et optimisation des préférences en planification de personnel

Julien Menana, Sophie Demassey, Narendra Jussien
École des Mines de Nantes, LINA CNRS UMR 6241, F-44307 Nantes, France.
sophie.demassey@mines-nantes.fr

8 mars 2010

Résumé

Dans des travaux précédents, la contrainte globale `multicost-regular` a été appliquée à la résolution des problèmes de planification de personnel. Cette contrainte, à base d'automates, permet une modélisation systématique de l'ensemble des règles de séquencement et de comptage des activités que doit satisfaire l'horaire de chaque employé. Elle offre, par agrégation de ces règles, un filtrage efficace. Les problèmes de planification de personnel étant généralement sur-contraints, nous nous intéressons dans cet article à la modélisation des formes de préférences typiques de ces problèmes et à la minimisation des coûts de violation associés aux contraintes. Nous proposons deux approches originales de modélisation systématique de la relaxation de contraintes sur le nombre d'occurrences d'une valeur ou d'un motif dans une séquence définie par un automate.

1 Introduction

Planification de personnel. Les problèmes de planification de personnel considérés dans cette étude sont de type *staff rostering*, aussi appelés problèmes de conception d'horaires ou *shift scheduling*. Il s'agit de concevoir les horaires des employés d'une organisation de manière à couvrir la charge de travail prévue sur une période de temps fixée et discrétisée. La charge de travail est définie comme le nombre d'employés requis à chaque unité de temps pour la réalisation de chacune des activités de l'organisation. Un employé est affecté à au plus une activité par unité de temps. L'horaire d'un employé ou *roster* est la séquence des activités (et inactivités) auxquelles il est affecté consécutivement à chaque unité de temps. La conception d'un horaire est potentiellement soumise à des règles nombreuses et variées, individuelles, organisationnelles ou réglementaires. Ainsi, la solution d'un tel problème, appelée planning ou *timetable*, peut être encodée dans un modèle de satisfaction de contraintes par une matrice de variables d'affectation aux activités : les colonnes, figurant les unités de temps, et chaque

ligne, l'horaire d'un employé. Un exemple de planning est représenté de la sorte par la figure 1.

jour	1	2	3	4	5	6	7
employé 1	J	J	N	-	N	-	N
employé 2	N	-	J	J	J	J	J
employé 3	-	N	-	N	-	-	-
employé 4	J	J	J	-	-	N	N

FIGURE 1 – Exemple de planning hebdomadaire discrétisé en 7 jours pour 4 employés et 3 activités (J)our, (N)uit et (-) repos.

Les contraintes de base définissant ces solutions se partagent en deux groupes : les contraintes de couverture de charge portant sur chaque colonne et les contraintes d'horaire sur chaque ligne.

Relaxation des contraintes. Selon les instances du problème, ces différentes contraintes peuvent être imposées de manière dure ou souple. Une contrainte dure est impérativement satisfaite par toute solution, tandis qu'une contrainte souple peut être relâchée, dans une certaine mesure, induisant un coût supplémentaire pour toute solution violant partiellement la contrainte. En présence de contraintes souples, le problème de planification de personnel est un problème d'optimisation dont une solution est un planning satisfaisant toutes les contraintes dures et minimisant la somme des coûts de violation des contraintes souples. Un tel problème d'optimisation peut avantageusement être modélisé dans un schéma classique de programmation par contraintes basé sur une représentation des coûts par des variables. Suivant Petit et al. [11], cette approche des problèmes sur-contraints est en effet facile à mettre en œuvre dans tout système de résolution de contraintes, et offre surtout une grande flexibilité dans la manipulation et la combinaison des coûts. Par ailleurs, quand la relaxation concerne une contrainte globale, sa variante d'optimisation ou variante souple peut être considérée. Une telle variante consiste à intégrer une variable de coût – en l'occurrence, la variable de coût de violation associée à la relaxation – au sein du filtrage de la contrainte relâchée. Une variante d'optimisation offre ainsi généralement une propagation plus efficace, qu'un modèle décomposé, pour la minimisation des coûts de violation. Des variantes d'optimisation existent pour des contraintes globales usuelles telles que `alldifferent`, `global-cardinality`, `regular`. La contrainte `regular` [8] est fondamentale dans le contexte de la planification de personnel pour la modélisation et l'agrégation des contraintes d'horaire. Elle permet notamment d'interdire l'apparition de motifs prohibés (typiquement, une nuit suivie d'un jour travaillé) dans l'horaire (la séquence des activités) d'un employé. En revanche, les variantes souples existantes pour cette contrainte ne sont pas exactement utilisables dans ce contexte.

Dans cette étude, nous considérons une nouvelle forme de relaxation de `regular` bornant le nombre d'occurrences d'un motif dans une séquence. En associant une variable-compteur d'occurrences du motif, nous montrons que

cette relaxation peut directement et systématiquement être modélisée au moyen d'une contrainte `cost-regular` [3] ou agrégée aux autres contraintes dures et souples d'horaire au sein d'une unique contrainte `multicost-regular` [6] (Section 3). Nous dérivons enfin une variante souple de `multicost-regular` en intégrant les variables de coûts de violation associées aux variables compteurs (Section 4). Le filtrage correspondant est une simple adaptation du filtrage de `multicost-regular` permettant une meilleure back-propagation, directement depuis les bornes des coûts. Incidemment, cette relaxation n'impose aucune restriction sur la nature de la fonction de pénalité liant un compteur à son coût de violation. Suite à nos précédents travaux sur le traitement systématique en satisfaction des problèmes de planification de personnel [6], nous visons ici le traitement systématique en optimisation. Nous débutons donc cette étude par examiner les formes de relaxations usuellement rencontrées dans ces problèmes (Section 2).

2 Modèles des préférences

Dans cette section, nous examinons les contraintes de base, de couverture et d'horaire, définissant les problèmes de planification de personnel. Pour chaque type de contraintes, nous discutons des formes de relaxation de ces contraintes et rapportons des modélisations possibles au moyen de contraintes globales génériques, couplées à des variables de coût de violation. Les variables de décisions sur lesquelles sont définis ces modèles sont les variables d'affectation x_{et} des employés e et unités de temps t aux activités a.

Nous n'abordons pas ici le cas des contraintes couplantes portant simultanément sur plusieurs employés ou plusieurs colonnes. Les contraintes couplantes souples de répartition des charges et d'équité entre employés sont notamment abordées dans [9].

Contraintes d'affectation. Les contraintes d'affectation sont des contraintes unaires traduisant des affectations interdites ($x_{et} \neq a$) ou obligatoires ($x_{et} = a$) d'un employé e à un temps t à une activité a. Ces contraintes sont classiquement relâchées en préférences, spécifiées par une pénalité $f_{et}(a) \in \mathbb{R}_+$ associée à chaque affectation $x_{et} = a$: plus la pénalité est faible, plus il est préférable d'effectuer l'affectation à la valeur. Le coût de violation associé à une affectation est défini par $z_{et} = f_{et}(x_{et})$. L'objectif du problème relâché consiste alors à minimiser la somme des coûts d'affectation :

$$z = \sum_e \sum_t z_{et}, \text{ avec } z_{et} = f_{et}(x_{et}) \ \forall (e,t).$$

Contraintes de couverture. Les contraintes de couverture des charges de travail sont généralement définies par des bornes minimales L_{at} et maximales U_{at} sur le nombre d'employés affectés à chaque activité a, à chaque unité de temps t. Les variantes communes sont les contraintes de couverture portant sur un groupe

d'activités ou restreintes à des sous-ensembles d'employés qui, typiquement, partagent une même compétence. Les contraintes `among` et `global-cardinality` (ou `gcc`) sont directement applicables à leur modélisation : `among` compte le nombre d'occurrences d'une valeur ou d'un groupe de valeurs, tandis que `gcc` porte simultanément sur toutes les valeurs. Soit $y_{at} \in [L_{at}, U_{at}]$ la variable compteur associée à une activité a et une période t, la contrainte $\texttt{gcc}((\{x_{et}\}_e, \{y_{at}\}_a)$ maintient la relation $y_{at} = |\{x_{et} = a\}_e|$ pour tout activité a. Une relaxation de cette contrainte consiste à spécifier des bornes préférentielles $l_{at} \geq L_{at}$ et $u_{at} \leq U_{at}$ et une mesure de pénalité $f_{at} : [L_{at}, U_{at}] \to \mathbb{R}_+$ vérifiant $f_{at}(y) = 0$ si $l_{at} \leq y \leq u_{at}$. Cette fonction permet d'exprimer différemment les coûts de violation en fonction de sa nature, constante, linéaire, ou même non continue ou non monotone. À noter que la fonction de pénalité associée à des bornes préférentielles est souvent de nature quadratique :

$$f_{at}(y) = \max(0, \underline{c}. \max(0, l_{at} - y)^2, \overline{c}. \max(0, y - u_{at})^2).$$

La relaxation d'une contrainte de couverture peut facilement – et ce, quelque soit la nature de la mesure de pénalité – être modélisée par une contrainte externe à `gcc`. Pour chaque activité a et chaque période t, une variable de coût $z_{at} \in \mathbb{R}_+$ est ajoutée et liée à la variable compteur y_{at} par une relation binaire $z_{at} = f_{at}(y_{at})$.

Quand la mesure de pénalité est linéaire ($f_{at}(y) = \max(0, l_{at} - y, y - u_{at})$), une alternative consiste à modéliser la relaxation au moyen de la variante souple de `global-cardinality` présentée par van Hoeve et al. [14] :

$$\texttt{soft\_val\_gcc}((\{x_{et}\}_e, \{l_{at}\}_a, \{u_{at}\}_a, z_t).$$

Cette contrainte lie les variables de décision à la variable de coût z_t en maintenant la contrainte couplante :

$$z_t = \sum_a f_{at}(|\{x_{et} = a\}_e|).$$

Dans le contexte du *staff rostering*, les contraintes de couverture s'appliquent indépendamment à chaque colonne de la matrice d'affectation. Une vision simultanée de l'ensemble des colonnes est préférable dans le contexte de problèmes d'ordonnancement cumulatif, où une activité est affectée à un employé sur plusieurs unités de temps consécutives. Petit et Poder [10] ont ainsi proposé une variante souple de `cumulative` avec des pénalités linéaires de sur-charge à chaque instant ($f_{at}(y) = \max(0, y - u_{at})$) et une contrainte couplante de somme des pénalités.

Contraintes d'horaires : occurrences de valeurs. Orthogonales aux contraintes de couverture, les contraintes d'horaires portent sur chaque ligne – chaque employé – indépendamment. La contrainte d'horaires la plus courante impose des bornes l_{ae} et u_{ae} sur le nombre d'affectations d'un employé e à une activité a, ou à un groupe d'activités. Comme pour les contraintes de couverture,

elle se modélise naturellement par `among` ou au sein de `gcc`, ou par une variante souple, en cas de relaxation de ces bornes. Le nombre d'occurences d'une activité peut indifféremment être compté sur la période totale de planification, sur un sous-intervalle de temps (au moins l_R repos par semaine, par exemple), ou sur un intervalle glissant (au plus u_N nuits travaillées tous les k jours consécutifs, par exemple). Ce dernier cas peut avantageusement être modélisé au moyen d'une contrainte `sequence` (ou `among_seq`) au lieu de la conjonction non-disjointe de `among`. La variante souple de cette contrainte a été étudiée dans [5] pour une mesure de pénalité linéaire en l'écart aux bornes ($f_{ae}(y) = \max(0, l_{ae} - y, y - u_{ae})$) : `soft_sequence`$((\{x_{et}\}_t, l_{ae}, u_{ae}, k_e, a, z_{ae})$ maintient la relation $z_{ae} \geq \sum_t f_{ae}(|\{x_{e(t+i)} = a\}_{i=0,\ldots,k-1}|)$.

Contraintes d'horaires : motifs interdits. D'autres contraintes d'horaires des plus hétérogènes sont présentes dans tout problème de planification de personnel, par exemple : un repos est requis après une nuit travaillée ; au plus trois journées consécutives sont travaillées ; les jours de week-end sont tous deux libres ou tous deux travaillés ; etc.

Toutes ces règles s'apparentent à interdire l'apparition d'un motif donné dans la séquence d'activités formant l'horaire d'un employé. Un horaire valide s'apparente alors à un mot de longueur fixe (la durée du planning) dans un langage rationnel, sur l'alphabet des activités, défini comme l'ensemble des mots ne contenant aucun des motifs interdits. Un tel langage est représentable sous la forme d'un automate constructible, par union et complément, à partir des expressions régulières des motifs. La contrainte `regular` [8] a initialement été développée dans ce cadre. Étant donné un automate fini déterministe Π_e modélisant le langage $\mathcal{L}(\Pi_e)$ des horaires valides pour l'employé e, `regular`$(\{x_{et}\}_t, \Pi_e)$ assure que la séquence $\{x_{et}\}_t$ des variables d'affectations de l'employé e forme un mot du langage $\mathcal{L}(\Pi_e)$. Cette contrainte globale est d'autant plus efficace qu'elle permet de traiter simultanément l'ensemble des motifs interdits spécifiés pour un même employé. À noter que les contraintes glissantes d'occurrences peuvent également être traduites en motifs interdits et intégrées à cette contrainte [13, 2].

L'imbrication des différentes règles portant sur un même employé est généralement telle qu'un filtrage efficace n'est envisageable qu'en les agrégeant pour les faire interagir au sein d'une même contrainte globale. La contrainte `global-sequencing` [12], agrégeant `global-cardinality` (compteurs globaux) à `sequence` (compteurs périodiques), a ainsi été développée dans le contexte du car-sequencing. Plus récemment, la contrainte `multicost-regular` [6] a permis d'agréger `global-cardinality` (compteurs de valeurs) à `regular` (motifs interdits).

Des relaxations de la contrainte `regular` existent pour différentes mesures de pénalités. La variante d'optimisation `cost-regular` [3] permet par exemple d'intégrer les coûts d'affectation : `cost-regular`$(\{x_{et}\}_t, \Pi_e, \{f_{eta}\}_{e,t,a}, z_e)$ assure que la séquence $\{x_{et}\}_{et}$ appartient au langage $\mathcal{L}(\Pi_e)$ et maintient les bornes de la somme des coûts d'affectation $z_e = \sum_t f_{et}(x_{et})$. Dans [14], deux variantes souples de `regular` sont également proposées. Cependant, les mesures de péna-

lité associées ne sont pas satisfaisantes dans le contexte de la planification de personnel, où les règles expriment des motifs interdits. Dans la section suivante, nous présentons une mesure de pénalité plus appropriée à ce contexte, en terme de nombre d'occurrences d'un motif dans une séquence, et montrons comment cette mesure peut directement être modélisée au moyen de `cost-regular` ou de `multicost-regular`.

3 Relaxation de motifs

Relaxations de `regular`. Deux variantes souples de `regular` ont été proposées par Van Hoeve et al. [14] : La première, `soft_hamming_regular`$((\{x_t\}_t, \Pi, z)$ assure que le coût z est égal au nombre minimal de substitutions de valeurs nécessaires dans la séquence $\{x_t\}_t$ pour que celle-ci soit acceptée par l'automate Π. La seconde, `soft_edit_regular`$((\{x_t\}_t, \Pi, z)$ assure que le coût z est égal au nombre minimal d'ajouts, de substitutions et de suppressions de valeurs nécessaires dans la séquence $\{x_t\}_t$ pour que celle-ci soit acceptée par l'automate Π. Ces deux mesures ne sont pas toujours appropriées à la planification de personnel. Dans ce contexte, la contrainte `regular` est employée pour modéliser des règles spécifiées le plus souvent, ou traductibles, en termes de motifs interdits. Une relaxation naturelle des règles de motifs interdits consiste à autoriser un nombre limité d'apparitions du motif. Les deux mesures ci-dessus ne permettent pas de compter le nombre d'occurrences d'un motif. Si on considère l'exemple [14] d'un alphabet à deux activités $\{a, b\}$ et deux règles : la longueur de toute sous-séquence maximale de a (resp. de b) est égale à 2. Le mot *abbaabbaab* viole deux de ces règles car il contient une séquence initiale (resp. finale) de a (resp. de b) de longueur différente de 2. Si la mesure edit retourne bien 2 dans ce cas, la mesure de hamming retourne 5. Comme second exemple, on considère le mot *aaaabb* et une règle interdisant le motif *aab*. Ce motif est seulement présent 1 fois dans la séquence, mais les mesures edit et hamming retournent 2, car toute substitution, ajout ou suppression d'une unique valeur dans la séquence laisse le motif présent.

Plus récemment, Beldiceanu et. al [1] ont proposé d'ajouter une variable binaire en début de la séquence indiquant si un motif est présent ou non dans la séquence. Soit $\mathcal{L}$ le langage formé de l'ensemble des mots contenant le motif, et $\overline{\mathcal{L}}$ son complément, la séquence $\{x_t\}_t$ précédée de la variable binaire est contrainte alors d'être acceptée par le langage $(0\mathcal{L})|(1\overline{\mathcal{L}})$. Si n est le nombre d'états de l'automate décrivant $\mathcal{L}$ et Σ l'alphabet, construire l'automate complémentaire décrivant $\overline{\mathcal{L}}$ s'effectue en temps $O(n|\Sigma|)$. Intégrée dans `regular`, cette modélisation possède deux limitations. D'une part, elle ne s'applique que dans le cas d'une mesure de pénalité binaire : la règle de motif interdit est violée ou non. D'autre part, la variable binaire ajoutée à la séquence de variables étant propre à la règle considérée, toutes ces règles ne peuvent plus alors être agrégées au sein d'une unique contrainte `regular`.

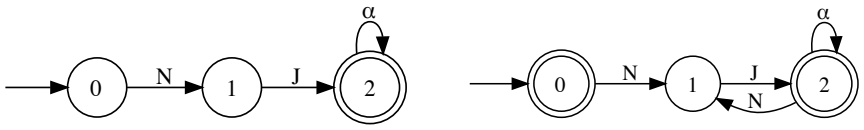
Compter un motif. Afin de compter le nombre d'occurrences d'un motif dans une séquence solution, et à l'instar du nombre d'occurrences d'une valeur, nous proposons de dériver de l'expression régulière du motif, à la fois un automate avec coût et une variable compteur. Ces derniers peuvent alors être traités par une contrainte `cost-regular`. Cette contrainte assure que la valeur du compteur est égale à la somme des coûts des transitions empruntées par la solution dans l'automate. L'automate dérivé accepte tous les mots du langage défini par $\Sigma*$. La difficulté de la construction consiste à définir les coûts des transitions de cet automate de manière à ce que le coût d'une solution corresponde au nombre d'occurrences du motif dans la solution. Le motif doit notamment être reconnu de manière glissante : le motif aa est rencontré par exemple 5 fois dans la séquence $aaaaaa$.

Métivier et al. [7] ont mis en œuvre cette technique de manière ad-hoc en construisant manuellement l'automate et les coûts pour des motifs particuliers. Une manière systématique de construire l'automate consiste à adapter la transformation d'une règle de séquence en une règle de séquence glissante, décrite initialement par van Hoeve et al. [13] pour l'encodage de `sequence` en `regular`, et généralisée dans [2] pour l'encodage de `slide`. Il est simple alors d'identifier les transitions à pondérer afin de compter le motif de manière glissante. L'automate produit par cette transformation comporte $O(k|\Sigma|^k)$ transitions et $O(|\Sigma|^k)$ états pour un motif de taille k. En effet, par cette construction, tous les mots de taille $k-1$ forment des chemins deux à deux sommets-disjoints dans l'automate.

Comme la complexité de l'algorithme de `regular` dépend du nombre de transitions dans l'automate, nous proposons une méthode de construction alternative produisant un automate minimal. Cette méthode simple est facilement implémentable à partir des opérations classiques des automates. Soit Σ l'alphabet, α un symbole n'appartenant pas à Σ, et $\mathcal{M}$ une expression rationnelle modélisant le motif à compter. À l'aide des opérations de concaténation et de répétition nous construisons l'automate $\mathcal{M}'$ correspondant à l'expression suivante :

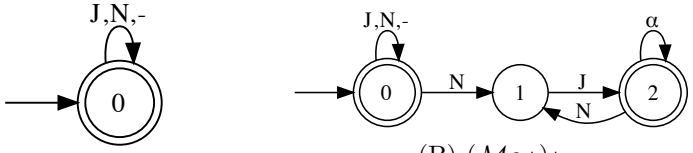
$$\mathcal{M}' \;=\; (\Sigma * (\mathcal{M}\alpha*)*)* \tag{1}$$

La concaténation de deux automates se fait en temps linéaire par rapport au nombre de transitions dans les deux automates. L'opération de répétition est linéaire en le nombre d'états terminaux fois la taille de l'alphabet. L'automate $\mathcal{M}'$ accepte tous les mots du langage formé par $\Sigma*$, ainsi que les mots constitués du motif à reconnaitre suivi de zéro ou plusieurs occurrences du symbole α. Il est nécessaire de déterminiser $\mathcal{M}'$ afin d'identifier les arcs que l'on pénalisera. Notons que l'automate $\Sigma*$ ne possède qu'un seul état, de ce fait, sa concaténation avec un autre automate ne nécessite pas d'ε-transition, réduisant la complexité de la déterminisation dans le pire des cas de $O(n^3 2^n)$ à $O(2^n)$ avec n le nombre d'états. Par construction $\mathcal{M}'$ est l'automate fini non déterministe (NFA) permettant de reconnaître l'ensemble des mots décrits par $\mathcal{M}\alpha*$ dans un texte. Si l'on forme un arbre à partir de ces mots, alors la déterminisation produira un automate fini déterministe (DFA) possédant le même nombre d'états



(A) $\mathcal{M}\alpha*$ (B) $(\mathcal{M}\alpha*)*$

FIGURE 2 – Automates reconnaissant $\mathcal{M}\alpha*$ (A) et $(\mathcal{M}\alpha*)*$ (B)



(B) $(\mathcal{M}\alpha*)*$

(A) $\mathcal{M}\alpha*$

FIGURE 3 – Automates reconnaissant $\Sigma*$ (A) et $\Sigma*(\mathcal{M}\alpha*)*$ (B)

que de noeuds dans cet arbre [4]. En pratique, cela signifie que pour un motif formant par exemple deux mots de taille k, l'automate construit possèdera au maximum $n = 2k + 1$ états, et par conséquent $n|\Sigma|$ transitions. Il est encore possible par la suite de minimiser cet automate sans perte d'information grâce aux transitions α. Par construction, tout état de $\mathcal{M}'$ possédant une transition sortante labélisée α n'est accessible qu'après avoir lu le motif $\mathcal{M}$. Pour chacun de ces états, on associe un coût de 1 à toute transition entrante. On associe un coût de 0 à toute autre transition dans l'automate. Finalement, l'automate résultat $\mathcal{M}_{min}$ est obtenu après suppression de toutes les transitions labélisées α ainsi que des états accessibles uniquement par de telles transitions.

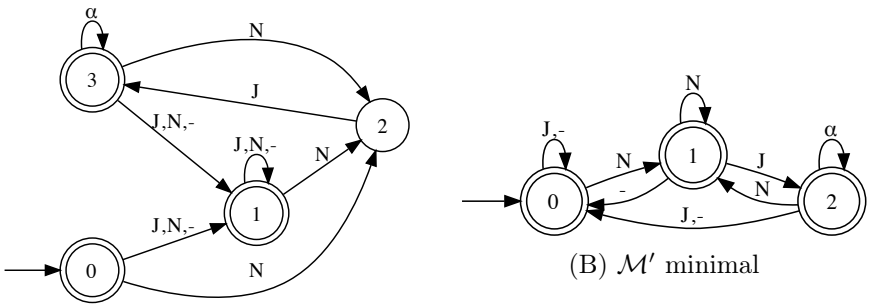
Par exemple, considérons le problème de conception d'horaires permettant à un employé de travailler de jour (J), de nuit (N) ou de ne pas travailler $(-)$. Nous allons construire l'automate $\mathcal{M}_{min}$ permettant de compter le nombre de fois où la règle interdisant de travailler de jour après une nuit (N) est violée.

Nous allons calculer $\mathcal{M}_{min}$ en suivant la Formule 1. Nous construisons d'abord le motif $\mathcal{M}\alpha*$ ainsi que sa version avec répétition (Figure 2).

Puis $\Sigma*$ et sa concaténation avec $(\mathcal{M}\alpha*)*$ calculé précédemment (Figure 3).

La répétition de l'automate Figure 3(B) forme $\mathcal{M}'$ (Figure 4(A)), que l'on peut minimiser de manière à obtenir le $\mathcal{M}'$ déterministe minimal (Figure 4(B)).

Enfin pour construire $\mathcal{M}'$ (Figure 5), les transitions labélisées α sont retirées et un coût de 1 est ajouté à chaque arc entrant dans l'état 2.



(A) $\mathcal{M}'$ (B) $\mathcal{M}'$ minimal

FIGURE 4 – $\mathcal{M}'$ (A) et $\mathcal{M}'$ minimal (B)

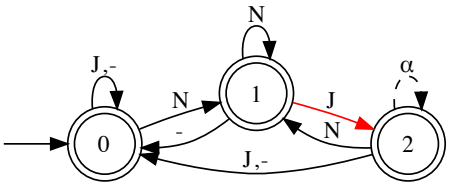


FIGURE 5 – $\mathcal{M}_{min}$

Agrégation dans `multicost-regular`. Une contrainte `cost-regular` appliquée à l'automate avec coûts $\mathcal{M}_{min}$ et à une nouvelle variable de compteur $y_{\mathcal{M}}$ permet de modéliser la relaxation escomptée : $y_{\mathcal{M}}$ compte le nombre d'occurrences du motif $\mathcal{M}$ dans la séquence. Le coût de violation $z_{\mathcal{M}}$ associé à toute mesure de pénalité $f_{\mathcal{M}}$ peut être modélisé par une relation binaire $f_{\mathcal{M}}(y_{\mathcal{M}}) = z_{\mathcal{M}}$ externe à `cost-regular`. De cette manière, chaque règle souple de motifs est traitée indépendamment dans sa propre contrainte globale, ce qui peut mener à un défaut de filtrage.

Une modélisation alternative, analogue à celle présentée dans [6], consiste à agréger l'ensemble des règles d'occurrences de valeurs et de motifs, dures et souples, au sein d'une même contrainte `multicost-regular`. Cette contrainte permet en effet de traiter plusieurs types de coûts indépendants et plusieurs variables compteurs associées sur un même automate, ce dernier représentant l'ensemble des mots autorisés (contraintes dures) pour une séquence.

Afin que les coûts des transitions de l'automate $\mathcal{M}_{min}$ soient conservés dans l'automate global à construire, il est cependant nécessaire de définir l'intersection d'automates avec coûts (Algorithme 1). En effet, on doit pouvoir retrouver dans l'automate intersection final, les transitions de coût égal à 1 dans chacun des automates individuels initiaux $\mathcal{M}_{min}$. Ces transitions sont dites *marquées*.

De manière similaire à la construction de l'automate $\mathcal{M}_{min}$, le marquage d'une transition consiste à remplacer son étiquette par un symbole n'appartenant pas à l'alphabet initial. Ainsi les opérations d'intersection et de minimisa-

tion peuvent être effectuées sans perdre d'information sur les transitions. Chaque nouvelle marque est identifiée de manière unique par l'identifiant de l'automate d'origine (ici, le numéro de compteur du motif correspondant), l'étiquette de la transition et le coût (ici, 1) de la transition. Une fois toutes les intersections réalisées et l'automate résultat minimisé, toutes les transitions portant un symbole hors alphabet sont *démarquées* : le symbole de la transition est remise à l'étiquette d'origine, et le coût associé au compteur correspondant.

Entrées: un DFA $\mathcal{A}$, un DFA marqué $\mathcal{M}_{min}$
Sorties: un DFA marqué $\mathcal{A} \cap \mathcal{M}_{min}$
Soit $\mathcal{A}_{mod}$ un nouveau DFA;
pour tous les *états* $q \in \mathcal{A}$ **faire**
  pour tous les *états* $q' \in \mathcal{M}_{min}$ **faire**
    ajouter l'état (q, q') à $\mathcal{A}_{mod}$
  fin
fin
pour tous les *états* $(q, q') \in \mathcal{A}_{mod}$ **faire**
  pour tous les *symboles* σ *dans* Σ **faire**
    $r \leftarrow \delta_{\mathcal{A}}(q, \sigma)$;
    $r' \leftarrow \delta_{\mathcal{M}_{min}}(q', \sigma)$;
    si *les états* r *et* r' *existent* **alors**
      $\delta_{\mathcal{A}_{mod}}((q, q'), \sigma) \leftarrow (r, r')$;
      si *l'arc* (q', r') *est marqué dans* $\mathcal{M}_{min}$ **alors**
        marquer l'arc $((q, q'), (r, r'))$ dans $\mathcal{A}_{mod}$;
      fin
    fin
  fin
fin

Algorithm 1: Intersection d'automates avec transitions marquées.

4 `multicost-regular` avec compteurs souples

Dans la section précédente, nous avons présenté un moyen de définir un compteur pour chaque motif dont on souhaite relâcher la règle d'interdiction en contrainte sur les bornes du nombre d'occurrences. Nous montrons également comment agréger ce type de règles avec l'ensemble des règles dures ou souples d'occurrences de valeur ou de motif, portant sur une même séquence de variables $\{x_t\}_t$ et au sein d'une unique contrainte `multicost-regular`. À chaque compteur d'occurrences y_r de valeur ou de motif peut alors être associé un coût de violation z_r des bornes préférentielles du compteur.

Modélisation des coûts de violation. Une première méthode consiste à donner des bornes plus larges aux variables compteurs y_r tout en pénalisant l'utilisation des valeurs au-delà des bornes préférentielles. Cette pénalisation est

modélisée à l'aide d'une variable de coût z_r et d'une table de paires autorisées définissant en extension une fonction f_r de violation. Cette fonction définit la relation suivante :

$$\forall r, \quad z_r = f_r(y_r)$$

Par exemple, considérons la règle indiquant que le motif NN doit apparaître entre 1 et 2 fois. Si cette règle est obligatoire, la définition d'un compteur $y_r \in [1, 2]$ ainsi que l'intégration de l'automate définissant ce motif dans la `multicost-regular` par la méthode décrite Section 3 sont suffisantes. Supposons maintenant que le nombre d'apparitions de ce motif puisse être violé et que le coût de violation pour chaque déviation augmente de manière quadratique ($f_r(y_r) = c_r.y_r^2$) avec, par exemple, un coût de 10 pour une déviation aux bornes de 1 ($1^2 * 10$), de 40 pour une déviation de 2 ($2^2 * 10$), etc. Il suffit alors d'étendre les bornes de y_r de 0 au nombre maximum de fois que le motif peut apparaître jusqu'à l'horizon du planning. Une table de relation entre z_r et y_r est alors calculée à partir de la fonction de violation. Pour un horizon de planning de 6 jours, le motif NN peut apparaître entre 0 et 5 fois, la Table 1 présente la liste des paires autorisées entre z_r et y_r.

y_r	0	1	2	3	4	5
z_r	10	0	0	10	40	90

TABLE 1 – Paires autorisées entre une variable compteur y_r et son coût de violation z_r

Avec cette modélisation, nous pouvons définir des fonctions de coûts de violation non linéaires et associer un coût de violation z à chaque `multicost-regular` (i.e. à chaque employé) comprenant un ensemble S de compteurs souples, défini par $z = \sum_{r \in S} z_r$.

Filtrage du coût de violation global. L'efficacité de la contrainte `multicost-regular` repose sur l'agrégation des différentes dimensions de coûts et compteurs au sein d'un algorithme de filtrage basé sur la relaxation lagrangienne [6]. La modélisation des coûts de violation proposée ci-dessus, extérieure à la contrainte, ne permet pas d'utiliser la structure de graphe sous-jacente à `multicost-regular` pour calculer de bonnes bornes pour la variable coût global z. Nous proposons donc d'adapter la `multicost-regular` à ce cas : `soft-multicost-regular` intègre la variable z dans la contrainte afin de calculer de meilleures bornes. Rappellons que le filtrage de `multicost-regular` s'effectue sur le graphe acyclique $\Pi_n = G(Q, E)$, qui est l'automate Π acceptant uniquement les mots de longueur n.

soft-multicost-regular. Soit
- $\{x_t\}_t$ une séquence de variables ;
- $\{y_r\}_r$ un ensemble de variables, représentant des compteurs ;

– $\{z_r\}_r$ un ensemble de variables, représentant les coûts de violation ou d'affectation des compteurs $\{y_r\}_r$;
– $\{f_r : \mathbb{N} \to \mathbb{Z}\}_r$ un ensemble de mesures de pénalité ;
– z la variable représentant le coût total de la contrainte ;
– Π un automate fini déterministe ;
– $\{c_{er}\}_{er}$ le coût d'utilisation de l'arc $e \in \Pi_n$ pour un compteur r.

`soft-multicost-regular` $(\{x_t\}_t, \{y_r\}_r, \{z_r\}_r, \{f_r\}_r, z, \Pi, \{c_{er}\}_{er})$ est satisfaite si et seulement si

$$\{x_t\}_t \in \mathcal{L}(\Pi) \tag{2}$$

$$z = \sum_r z_r \tag{3}$$

$$z_r = f_r(y_r) \qquad \forall r \tag{4}$$

$$y_r = \sum_{(e)\in\Pi_n} c_{er}\delta_e^x \qquad \forall r, \tag{5}$$

où $(\delta_e^x)_{e\in E} \in \{0,1\}^E$ dénote le vecteur d'incidence associé au chemin $\{x_t\}_t$ dans Π_n.

Notons Γ l'ensemble des chemins dans Π_n. La cohérence des contraintes 3 et 4 est maintenue par des contraintes de somme et de table à l'extérieur de la contrainte. Ces dernières pourraient être intégrées au filtrage de la `soft-multicost-regular` de manière à éviter les pertes de performances dues aux mécanismes internes du solveur.

Calculer les bornes de z au sein de `soft-multicost-regular` revient à résoudre les problèmes suivants :

$$\begin{aligned} \underline{z} = \min \sum_r f_r(y_r) & \\ \text{s.t. } y_r = \sum_{(e)\in\Pi_n} c_{er}\delta_e^x & \qquad \forall r \\ \delta_e^x \in \Gamma, & \end{aligned}$$

et

$$\begin{aligned} \overline{z} = \max \sum_r f_r(y_r) & \\ \text{s.t. } y_r = \sum_{(e)\in\Pi_n} c_{er}\delta_e^x & \qquad \forall r \\ \delta_e^x \in \Gamma. & \end{aligned}$$

Par souci de concision, nous ne considérons maintenant que le calcul de $\underline{z}$, le calcul de $\overline{z}$ étant symétrique. Ce problème d'optimisation sous-jacent à `soft-multicost-regular` possède la même structure que celui sous-jacent à `multicost-regular`. Le filtrage basé sur la relaxation lagrangienne s'adapte de la manière suivante :

Un nouveau sous problème lagrangien est défini pour tout $\lambda \in \mathbb{R}^R$:

$$SP(\lambda): \quad g(\lambda) = \min \sum_{(r)\in R} f_r(y_r) - \lambda_r y_r + \min_{\delta_e^x \in \Gamma} \sum_{e \in \Pi_n} \delta_e^x \sum_r \lambda_r c_{er}$$

Pour tout vecteur λ, résoudre $SP(\lambda)$ revient à résoudre $R+1$ problèmes indépendants :

Pour tout r on cherchera $y_r \in \mathbb{Z}_+$ qui minimise

$$\hat{g}_r(\lambda) = f_r(y_r) - \lambda_r y_r$$

. La mesure de pénalité f_r n'étant pas forcément monotone, il faut calculer $\hat{g}_r(\lambda)$ pour toutes les valeurs y_r. y_r étant un compteur d'occurrences cela ne concernera en pratique jamais plus que quelques dizaines de valeurs. Notons que plus l'on connaîtra les propriétés de f_r plus on pourra optimiser ce calcul.

On cherche également à minimiser

$$\sum_{e \in \Pi_n} \delta_e^x \sum_r \lambda_r c_{er}$$

Il s'agit ici de trouver le plus court chemin dans le graphe Π_n dont le poids des arcs e est $\sum_r \lambda_r c_{er}$.

Résoudre le dual lagrangien, revient à trouver le vecteur $\lambda*$ qui maximise g. Cela peut être fait en utilisant une implémentation de la méthode du sous-gradient telle celle décrite dans [6].

Cette modification de l'algorithme de la `multicost-regular` permet donc d'agréger les coûts de violation et d'affectation en un compteur global tout en fournissant des bornes de bonne qualité à ce compteur. Au sein d'un problème de conception d'horaires l'usage de cette variable est double : accéder rapidement à une bonne estimation du coût de violation global pour un employé et être capable d'imposer une borne sur ce coût de violation.

5 Étude expérimentale

Nous avons débuté les expérimentations sur les instances NRP publiées sur `http://www.cs.nott.ac.uk/~tec/NRP/`. Ces instances issues de problématiques réelles de planification de personnel dans le domaine hospitalier principalement présentent une grande variété de contraintes d'horaire et de couverture. Toutes sont disponibles dans un format de données générique sous XML. Nous avons ainsi implémenté un lecteur de données basé sur ce format d'instance. Le lecteur construit le modèle de satisfation ou d'optimisation de contraintes de manière systématique, par traduction et analyse des balises. La principale difficulté consiste à redécouvrir la sémantique perdue de certaines règles spécifiées (motifs interdits, occurrences de valeurs, occurrences de motifs) afin de

construire l'objet idoine dans le modèle (automate ou compteur et automates avec coûts). Les contraintes du modèle sont alors créées sur les variables de décision :

$$x_{et} \in \mathcal{A}, \qquad \forall e, \forall t. \tag{6}$$

Les contraintes d'affectation dures sont modélisées par de simples contraintes unaires :

$$x_{et} = a. \tag{7}$$

Les contraintes de couverture sont spécifiées pour chaque temps t et pour chaque activité a. Un ensemble E d'employés couverts par la contrainte est parfois spécifié (par défaut, E est l'ensemble de tous les employés). On définit alors un compteur y^E_{at} et, si la contrainte est souple, un coût de violation z^E_{at}.

$$y^E_{at} \in [L^E_{at}, U^E_{at}],\ z^E_{at} \in [0, U], \qquad \forall a, \forall t, \forall E \tag{8}$$

$$z^E_{at} = f^E_{at}(y^E_{at}), \qquad \forall a, \forall t, \forall E \tag{9}$$

$$\texttt{gcc}(\{x_{et}\}_{e \in E}, \{y^E_{at}\}_a), \qquad \forall t, \forall E. \tag{10}$$

Des contraintes de couverture sont également parfois spécifiées pour un sous-ensemble d'activités A :

$$y^E_{At} \in [L^E_{At}, U^E_{At}],\ z^E_{At} \in [0, U], \qquad \forall A, \forall t, \forall E \tag{11}$$

$$z^E_{At} = f^E_{At}(y^E_{At}), \qquad \forall A, \forall t, \forall E \tag{12}$$

$$\texttt{sum}(\{y^E_{at}\}_{a \in A}) = y^E_{At}, \qquad \forall A, \forall t, \forall E. \tag{13}$$

Toutes les contraintes d'horaires, dures et souples, et les contraintes d'affectation souples sont modélisées par un automate Π_e et un ensemble de compteurs $\{y_{er}\}_r$ au sein d'une contrainte `multicost-regular` par employé.

$$\texttt{multicost-regular}(\{x_{et}\}_t, \{y_{er}\}_r, \Pi_e), \qquad \forall E. \tag{14}$$

Toute contrainte dure de motif interdit est entièrement modélisée dans l'automate Π_e. Si la contrainte est souple (une fonction de pénalité est spécifiée), alors Π_e est modifié selon la procédure décrite section 3, et un compteur y_{er} et un coût de violation z_{er} sont créés et liés par une contrainte binaire donnée en extension (à l'heure actuelle, nous n'avons pas encore implémenté le modèle avec `soft-multicost-regular` décrite section 4) :

$$y_{er} \in [L_{er}, U_{er}],\ z_{er} \in [0, U], \qquad \forall e \tag{15}$$

$$z_{er} = f_{er}(y_{er}), \qquad \forall t. \tag{16}$$

Les instances NRP présentent également des contraintes dures ou souples limitant (inférieurement ou supérieurement) le nombre d'apparitions d'un motif. Ces contraintes se modélisent pareillement aux contraintes souples de motifs interdits ; la variable de coût z_{er} n'étant définie que si la contrainte est spécifiée

souple. Il en est de même aussi pour les contraintes dures et souples d'occurrences de valeur (ou de groupe de valeurs). À noter que dans ce cas, la pondération de l'automate se fait trivialement en affectant un coût de 1 à toute transition étiquetée par la valeur considérée, et 0 sinon. Enfin, les contraintes souples d'affectation pour un employé sont modélisées par un unique compteur-coût $z_{er} = y_{er}$, chaque transition étant pondérée par la pénalité correspondante f_{eta}.

La variable objectif à minimiser est la somme des coûts de violation de l'ensemble des contraintes souples :

$$z \in [0, U], \texttt{ minimize } z \tag{17}$$

$$\texttt{sum}(\{z^E_{at}\}, \{z^E_{At}\}, \{z_{er}\}) = z. \tag{18}$$

À défaut d'une heuristique de branchement générique et efficace, capable de tenir compte simultanément des contraintes orthogonales de couverture et d'horaire, et de la minimisation des coûts, nous utilisons actuellement ce modèle pour calculer des bornes inférieures et supérieures sur le coût minimum des violations. La borne inférieure est calculée en augmentant progressivement d'une unité la valeur maximale U du coût z, tant que le modèle est prouvé irréalisable. Si une solution est trouvée, elle est nécessairement optimale. Notre modèle trouve ainsi une solution optimale de coût 0 pour les instances suivantes :

instance	temps (s)	#noeuds	#backtracks
Azaiez	9.5	4006	5574
Sintef	2.2	165	53
Millar-2S-1.1	0.7	29	0
Millar-2S-1	0.4	25	0
Ozkarahan	0.3	24	5

Un temps maximum d'exécution de 1 minute par tentative U est posé. La borne inférieure déduite est la dernière valeur U testée pour laquelle l'infaisabilité n'est pas prouvée. On trouve par exemple la borne inférieure $LB = 3$ (resp. $LB = 4$) pour l'instance GPost-B d'optimum 3 (resp. GPost d'optimum 5) en 1 minute et 20 secondes (resp. 1 minute et 5 secondes).

Pour le calcul de borne supérieure, une heuristique de branchement sur les variables de décision est lancée avec une borne supérieure initialement non bornée $UB = +\infty$. La meilleure solution trouvée en un temps imparti est conservée. Pour l'instance GPost-B par exemple, 8 solutions dont la meilleure de coût 4 sont trouvées en 630 secondes et 234.000 noeuds. Pour l'instance GPost, 12 solutions dont la meilleure de coût 17 sont trouvées en 276 secondes et 91.000 noeuds.

Des expérimentations plus poussées, à partir d'heuristiques de branchement plus performantes sont encore à mener. Dès à présent l'efficacité du filtrage proposé par notre approche se traduit en résultats comparables aux meilleurs existants `http://www.cs.nott.ac.uk/~tec/NRP/`. Il est à noter également que les résultats présentés ici tiennent compte du temps de construction des modèles, et surtout que cette construction est entièrement automatisée, indépendamment de l'instance traitée.

6 Conclusion

Dans cet article, nous poursuivons l'étude débutée dans [6] concernant la modélisation systématique des problèmes de planification de personnel en programmation par contraintes. Nous traitons ici des problèmes sur-contraints où la somme des coûts de violation de chaque règle doit être minimisée. Une nouvelle mesure de pénalité des règles de motifs interdits est notamment considérée : celle-ci nécessite de compter le nombre d'apparitions d'un motif dans une séquence. Nous montrons comment le nombre d'occurrences d'un motif peut être traité de manière similaire au nombre d'occurrences d'une valeur, par dérivation d'un automate avec coûts et d'un compteur, passés en arguments de la contrainte d'optimisation `cost-regular`. Nous présentons alors comment agréger l'ensemble des règles, dures ou souples, d'occurrences de valeurs et de motifs au sein d'une unique contrainte globale `multicost-regular` par employé. Nous montrons qu'une rapide adaptation de la relaxation lagrangienne sous-jacente au filtrage de cette contrainte permet de tenir compte des coûts de violation associés aux compteurs d'occurrences souples, et ce, quelque soit la nature des fonctions de pénalités considérées.

Ces travaux mettent de nouveau en avant l'expressivité des automates et la capacité des *meta-contraintes*-automates à modéliser des règles nombreuses et variées. Il est notamment plus facile de modifier un automate ou de le créer de manière systématique que de multiplier les algorithmes de filtrage pour les diverses contraintes. Surtout, ces contraintes offrent la possiblité d'un filtrage effectif en agrégeant un ensemble de règles intrinsèquement liées.

L'étape suivante de ces travaux va consister à gérer de manière systématique l'interaction des contraintes d'horaire et de couverture, par filtrage, heuristiques de branchement, ou encore par hybridation à d'autres méthodes de résolution, pour dériver une méthode efficace et automatique de résolution des problèmes sur-contraints de planification de personnel.

Références

[1] N. Beldiceanu, M. Carlsson, and J.X. Rampon. Global Constraint Catalog. Technical report, EMN, 2008.

[2] Christian Bessière, Emmanuel Hebrard, Brahim Hnich, Zeynep Kiziltan, Claude-Guy Quimper, and Toby Walsh. Reformulating global constraints : The SLIDE and REGULAR constraints. In *SARA*, pages 80–92, 2007.

[3] Sophie Demassey, Gilles Pesant, and Louis-Martin Rousseau. A cost-regular based hybrid column generation approach. *Constraints*, 11(4) :315–333, 2006.

[4] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Pearson Education. Addison-Wesley, 2001.

[5] M. Maher, N. Narodytska, C.-G. Quimper, and T. Walsh. Flow-Based Propagators for the *sequence* and Related Global Constraints. In *Proceedings of CP'2008*, volume 5202 of *LNCS*, pages 159–174, 2008.

[6] Julien Menana and Sophie Demassey. Sequencing and counting with the `multicost-regular` constraint. In *6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'09)*, volume 5547 of *Lecture Notes in Computer Science*, pages 178–192, Pittsburgh, USA, May 2009. Springer-Verlag.

[7] Jean-Philippe Métivier, Patrice Boizumault, and Samir Loudni. Solving nurse rostering problems using soft global constraints. *Principles and Practice of Constraint Programming - CP 2009*, pages 73–87, 2009.

[8] Gilles Pesant. A regular language membership constraint for finite sequences of variables. In *Principles and Practice of Constraint Programming (CP'04)*, pages 482–495, 2004.

[9] Gilles Pesant and Jean-Charles Régin. Spread : A balancing constraint based on statistics. *Principles and Practice of Constraint Programming - CP 2005*, pages 460–474, 2005.

[10] Thierry Petit and Emmanuel Poder. Global propagation of side constraints for solving over-constrained problems. *to appear in Annals of Operations Research*, 2010.

[11] Thierry Petit, Jean-Charles Régin, and Christian Bessière. Meta constraints on violations for over constrained problems. In *IEEE-ICTAI*, pages 358–365, 2000.

[12] Jean-Charles Régin and Jean-Francois Puget. A filtering algorithm for global sequencing constraints. In *CP*, pages 32–46, 1997.

[13] W.-J. van Hoeve, G. Pesant, L.-M. Rousseau, and A. Sabharwal. Revisiting the *sequence* Constraint. In *Proceedings of CP'2006*, volume 4204 of *LNCS*, pages 620–634, 2006.

[14] Willem Jan van Hoeve, Gilles Pesant, and Louis-Martin Rousseau. On global warming : Flow-based soft global constraints. *J. Heuristics*, 12(4-5) :347–373, 2006.