Article 34. Modélisation et Résolution du Problème de l'Emploi du Temps en tant que Problème de Satisfaction de Contraintes

T. A. Ralijaona
École du Génie du Management d'Entreprises et du Commerce,
Institut Supérieur de Technologie d'Antananarivo,
tiana.ralijaona@ist-tana.mg

Mots clés : Problème de l'emploi du temps, Problème de satisfaction de contraintes, Backtracking, Intelligence artificielle, Recherche opérationnelle.

Résumé

Le problème de l'emploi du temps (dans un cadre scolaire ou universitaire) est un cas particulier non trivial du problème de satisfaction de contraintes. Globalement, il consiste à planifier les affectations de groupes d'étudiants, de salles de classe, de tranches horaires, de matières et d'enseignants étant donné les contraintes telles que la disponibilité des enseignants, le nombre de salles de classe, … Nous proposons donc ici une modélisation mathématique du problème de l'emploi du temps dans un cadre scolaire ou universitaire sous la forme d'un problème de satisfaction de contraintes. Nous avons aussi adapté, spécialisé et amélioré l'algorithme de backtracking qui résout le problème de satisfaction de contraintes pour résoudre le problème de l'emploi du temps

I. Introduction

En optimisation combinatoire, les problèmes de satisfaction de contraintes sont des problèmes où l'on cherche à déterminer l'état ou la valeur de variables de décision satisfaisant certaines contraintes ou critères. Ces problèmes sont au cœur même de la recherche en intelligence artificielle et en recherche opérationnelle car elles fournissent un modèle générique pour résoudre une multitude de problème dans ces domaines tel que les problèmes de planification, d'ordonnancement et d'allocation de ressources [5].

Classiquement, le problème de l'emploi du temps est modélisé sous la forme d'un problème d'optimisation linéaire en nombre entier binaire ou une de ses variantes [1, 4, 7, 9]. De ce fait, l'espace des solutions est énorme de par sa nature binaire même. Par conséquent, ceci est une pénalité certaine dans sa résolution. De plus, les différentes techniques de résolution ne considèrent pas la structure et les propriétés intrinsèques du problème. Ainsi, les techniques de résolution ne profitent pas de toutes les informations inhérentes au problème [1]. Cependant, le problème de satisfaction de contraintes offre une structure générique et une formulation plus naturelle pour le problème de l'emploi du temps. En effet, le problème de l'emploi du temps est un cas particulier de problème de planification [13]. Ce qui amène à poser la question : comment modéliser et résoudre le problème de l'emploi du temps en tant que problème de satisfaction de contraintes ?

L'objectif est donc de modéliser le problème de l'emploi du temps sous la forme d'un problème de satisfaction de contraintes. Dans le but de, non seulement, bénéficier d'une structure générique et compacte par rapport au modèle d'optimisation linéaire en nombre entier binaire, mais aussi de capter la nature et les propriétés intrinsèques du problème de l'emploi du temps même. Et ceci afin de proposer une technique de résolution efficace.

Dans les sections suivantes, nous allons tout d'abord définir le problème de satisfaction de contraintes et son extension valuée ainsi que l'algorithme générique de backtracking qui va nous permettre de le résoudre. Puis, nous allons modéliser le problème de l'emploi du temps sous la forme d'un problème de satisfaction de contraintes valuée et adapter, spécialiser et améliorer l'algorithme de backtracking pour résoudre ce dernier. Ensuite, avant de conclure, nous ferons quelques observations sur les possibilités d'extensions à ces recherches.

II. Matériels et méthodes

En premier lieu, nous allons définir les modèles de base qui vont servir à formaliser le problème de l'emploi du temps.

2.1 Problème de satisfaction de contraintes

Un *problème de satisfaction de contraintes* (ou CSP) est un triplé $\mathcal{P} = \langle X, D, C \rangle$ [3] où :

- $X = \{X_1, X_2, ..., X_n\}$ est un ensemble de variables,
- $D = \{D_1, D_2, ..., D_n\}$ est l'ensemble des domaines de valeurs respectifs des variables,
- $C = \{C_1, C_2, ..., C_m\}$ est l'ensemble des contraintes qui vont restreindre les valeurs que peuvent prendre les variables.

Pour un CSP, chaque contrainte $C_j \in C$ est un couple $\langle t_j, R_j \rangle$ où :

- $t_j \subseteq X$ est un sous-ensemble de k variables,

- R_j est une relation *k*-aire sur les domaines correspondants aux variables.

Soulignons que c'est bien chaque relation R_j qui va restreindre les valeurs que peuvent prendre simultanément les k variables de t_j.

Une *solution* pour un CSP est une affectation de valeurs pour toutes les variables provenant de leurs domaines respectifs telle que toutes les contraintes soient satisfaites. Dans ce cas, le problème est dit *satisfiable*. Dans le cas contraire, il est *insatisfiable*.

2.2 Problème de satisfaction de contraintes valuée

Un *problème de satisfaction de contraintes valuée* (ou VCSP) est un quintuplé $\mathscr{R} = \langle X, D, C, S, \varphi \rangle$ [10] défini par :

- un CSP classique $\langle X, D, C \rangle$,
- une structure de valuation S,
- une application $\varphi : C \rightarrow S$,

où pour toutes contraintes $C_j \in C$, $\varphi(C_j)$ est appelé *valuation* de C_j. Dans ce cas, une valuation servira à mesurer l'impact de la violation ou non des contraintes.

Ainsi, trouver une *solution* a un VCSP revient donc à chercher une affectation de valeurs pour toutes les variables provenant de leurs domaines respectifs telle que la *valuation du problème* soit *minimale* (voir [10] pour plus de détails sur la structure de valuation et la valuation d'un VCSP). Elle fournit donc une mesure graduelle d'inconsistance en rapport avec la violation ou non des contraintes par l'affectation de valeurs en question.

2.3 Algorithme de backtracking

Une méthode pour résoudre un CSP est l'*algorithme de backtracking*. Elle consiste à affecter à la suite aux variables du problème des valeurs. La validité d'une contrainte est alors vérifié aussitôt que les variables correspondantes à cette contrainte sont toutes instanciées (c'est à dire, affectées de valeurs). Si une instanciation partielle viole une des contraintes alors la valeur de la dernière variable instanciée est changée par la prochaine valeur disponible. Au cas où toutes les valeurs ont été épuisées pour cette dernière alors on remonte à l'avant dernière variable et ainsi de suite [8].

Un pseudocode de l'algorithme de backtracking est donné ci-dessous :

```
Procédure Backtrack(candidat):
Si Rejeter(candidat) alors retourner
Si Accepter(candidat) alors retourner candidat
extension ⟵ PremierExtension(candidat)
Tant que extension ¬ = ∅:
Backtrack(extension)
extension ⟵ ProchainExtension(extension)
FinProcédure
```

où

- Rejeter(candidat) : retourne Vrai si le candidat partiel ne peut pas être complété,
- Accepter(candidat) : retourne Vrai si le candidat est une solution,
- PremierExtension(candidat) : génère la première extension de candidat,
- ProchainExtension(extension) : génère la prochaine extension.

Généralement, l'algorithme de backtracking présenté précédemment est utilisé avec d'autres techniques algorithmiques afin d'améliorer sa performance. En effet, l'algorithme de backtracking simple ne vérifie que les contraintes entre la variable courante et les anciennes variables. Tandis que les techniques comme *forward checking* (FC) et *maintaining arc consistency* (MAC) vérifient les contraintes entre la variable courante, les anciennes et les futures variables. Ce qui permet de réduire énormément l'espace des candidats potentiels [3, 8].

Dans le cas du VCSP, l'algorithme de backtracking ainsi que ces raffinements (FC et MAC) peuvent être adaptés afin d'intégrer le concept de valuation du problème. En effet, étant donné que le backtracking utilise comme condition de retour (c'est à dire, un changement de valeurs d'une variable) la violation d'une contrainte alors, pour l'étendre au cas du VCSP, une borne supérieure de la valuation du problème sera utilisée comme condition de retour [10].

III. Résultats et discussions

En second lieu, nous allons présenter les résultats de notre travail. À savoir, la modélisation et la résolution du problème de l'emploi du temps en utilisant le formalisme du VCSP et l'algorithme de backtracking raffiné et adapté.

3.1 Modélisation du problème de l'emploi du temps

Pour modéliser le problème de l'emploi du temps, nous aurons tout d'abord besoin des ensembles et applications de base définis ci-après.

Ensembles de base

- l'ensemble des enseignants : $E = \{e_1, e_2, \ldots, e_\alpha\}$,
- l'ensemble des matières enseignées : $M = \{m_1, m_2, \ldots, m_\beta\}$,
- l'ensemble des classes d'étudiants : $C = \{c_1, c_2, \ldots, c_\gamma\}$,
- l'ensemble des horaires de l'emploi du temps : $H = \{h_1, h_2, \ldots, h_\delta\}$,
- et l'ensemble des salles utilisées : $S = \{s_1, s_2, \ldots, s_\varepsilon\}$.

Applications de base

- la disponibilité des enseignants est donnée en tant qu'un ensemble d'horaires :

  $D : E \rightarrow P(H)$, $e_i \mapsto D(e_i) = D_{ei}$;

- chaque matière est associée à son enseignant :

  $\mathcal{E} : M \rightarrow E$, $m_i \mapsto \mathcal{E}(m_i) = e_{mi}$;

- la liste des matières enseignées dans une classe est donnée en tant qu'un ensemble de matières :

  $\mathcal{M} : C \rightarrow P(M)$, $c_i \mapsto \mathcal{M}(c_i) = \mathcal{M}_{ci}$;

- la liste des salles utilisées pour une matière est donnée en tant qu'un ensemble de salles :

  $S : M \rightarrow P(S)$, $m_i \mapsto S(m_i) = S_{mi}$;

- le volume horaire de chaque matière est mesuré en nombre d'heures nécessaire à son achèvement :

  $V : M \rightarrow \mathbb{N}^*$, $m_i \mapsto V(m_i) = v_{mi}$;

- la durée de chaque tranche horaire doit être définie pour pouvoir calculer le volume horaire total réalisé pour une matière :

  $\mathcal{H} : H \rightarrow \mathbb{N}^*$, $h_i \mapsto \mathcal{H}(h_i) = d_{hi}$;

- la liste des classes associées à une matière est donnée en tant qu'un ensemble de classes :

  $C : M \rightarrow P(C)$, $m_i \mapsto C(m_i) = C_{mi}$;

- la capacité des salles est mesurée en nombre de places étudiants disponibles :

  $K : S \rightarrow \mathbb{N}^*$, $s_i \mapsto K(s_i) = k_{si}$;

- et l'effectif des étudiants dans une classe doit être donné :

  $N : C \rightarrow \mathbb{N}^*$, $c_i \mapsto N(c_i) = n_{ci}$

Ce qui nous permet alors de modéliser le problème de l'emploi du temps sous la forme d'un VCSP.

Modélisation VCSP

- l'ensemble des variables du problème est un ensemble de variables indicées par les tranches horaires et les salles :

  $$X = \{X_{hi,sj} : (h_i, s_j) \in H \times S\},$$

- le domaine des valeurs est le produit cartésien des matières enseignées :

  $$D = M^{\delta\varepsilon},$$

- les contraintes sont alors :

  - la non ubiquité des matières enseignées (c'est à dire, qu'une matière ne peut être enseignée dans deux salles différentes dans une même tranche horaire) :

    $\forall (h_i, s_j)$, $X_{hi,sj\neg} = \emptyset \Rightarrow X_{hi,sj\neg} = X_{hi,sk}$ pour $k\neg = j$;

- la non ubiquité des enseignants :

  $\forall (h_i, s_j), X_{hi,sj} = m_k \Rightarrow X_{hi,sl} \notin \mathcal{E}^{-1}(e_{mk})$ pour $l\neg = j$;

- la non ubiquité des classes d'étudiants :

  $\forall (h_i, s_j), X_{hi,sj} = m_k \Rightarrow \forall c_l \in C_{mk}, X_{hi,sp} \notin \mathcal{M}_{cl}$ pour $p\neg = j$;

- le respect de la disponibilité des enseignants :

  $\forall m_i, \forall s_j, \forall h_k \notin D(e_{mi}), X_{hk,sj}\neg = m_i$;

- le respect des salles appropriées pour chaque matière :

  $\forall m_i, \forall h_j, \forall s_k \notin S_{mi}, X_{hj,sk}\neg = m_i$;

- et le respect de la capacité des salles :

  $\forall h_i, \forall s_j, \forall m_k, N(C_{mk}) \geq K(s_j) \Rightarrow X_{hi,sj}\neg = m_k$;

- la structure de valuation est l'ensemble :

$$S = \mathbb{R}^+ \cup \{\infty\};$$

- et l'application de valuation va mesurer la réalisation ou non des volumes horaires pour chaque matière :

$$\Phi(X) = \sum_{i=1}^{\beta} p_i \, | V(m_i) - W_X(m_i) |$$

où

$$W_X : M \rightarrow \mathbb{N}, \; m_i \mapsto W_X(m_i) = w_{mi}$$

est l'application qui donne le volume horaire total réalisé pour chaque matière étant donné une instanciation des variables de X. Et les $p_i \in S$ sont des poids donnés à la réalisation ou non du volume horaire d'une matière bien définie. Ceci étant, la *résolution du problème de l'emploi du temps* revient donc à trouver une instanciation des variables de X minimisant l'application de valuation Φ.

3.2 Résolution du problème de l'emploi du temps

Pour résoudre le problème de l'emploi du temps, nous présentons alors ci-après l'algorithme de backtracking raffiné par la technique MAC et adapté au cas du VCSP (tel qu'il est suggéré par [10]).

Tout d'abord, nous présentons ci-après des sous-procédures qui seront utilisées dans l'algorithme de backtracking raffiné MAC.

Procédure de balayage de domaine

Procédure BalayageDomaine($X_{hi,sj}$):

Pour chaque $m_l^{i,j} \in M$:
Si $dom_l^{i,j}$ n'est pas marqué alors retourner VRAI
Retourner FAUX
FinProcédure

Procédure de restauration de domaine

Procédure RestaurationDomaine(X,Niv):

Pour chaque $X_{hi,sj} \in X$:
Pour chaque $m_l^{i,j} \in M$:
Si $dom_l^{i,j}$ est marqué à Niv alors enlever le marquage
FinProcédure

Procédure de révision de la consistance des variables

Procédure Réviser($X_{hi,sj}$, $X_{hg,sh}$,Niv):

Effacé ⟵ FAUX
Pour chaque $m_l^{i,j} \in M$ tel que $dom_l^{i,j}$ n'est pas marqué:
Trouvé ⟵ FAUX
Pour chaque $m_k^{g,h} \in M$ tel que $dom_k^{g,h}$ n'est pas marqué:
Si ($m_l^{i,j}, m_k^{g,h}$) satisfait aux contraintes entre $X_{hi,sj}$ et $X_{hg,sh}$ alors:
Trouvé ⟵ VRAI

Sortir
Si pas Trouvé alors:
Marquer $dom_l^{i,j}$ à Niv
Effacé ⟵ VRAI
retourner Effacé
FinProcédure

Procédure de propagation de contraintes

Procédure Propager(X,Niv):
Tant que $Q\neg = \emptyset$
Sélectionner et enlever $(X_{hi,sj}, X_{hg,sh})$ dans Q
Si Réviser($X_{hi,sj}$, $X_{hg,sh}$,Niv) alors:
Si BalayageDomaine($X_{hi,sj}$) alors retourner FAUX
Sinon:
Pour chaque $X_{he,sf} \in X$ tel que $(e,f)\neg = (g,h)$:
Si il existe une contrainte entre $X_{he,sf}$ et $X_{hi,sj}$ alors:
$Q \longleftarrow Q \cup (X_{he,sf}, X_{hi,sj})$
retourner VRAI
FinProcédure

Procédure de recherche de solution

Procédure Rechercher(X, Niv):

Sélectionner $X_{hi,sj} \in X$
Pour chaque $m_l^{i,j} \in M$ tel que $dom_l^{i,j}$ n'est pas marqué:
Solution ⟵ Solution + $(X_{hi,sj}, m_l^{i,j})$
Si $X_{hi,sj}$ est la seule variable dans X alors retourner VRAI
Sinon:
Pour chaque $m_l^{i,j} \in M \backslash \{m_l^{i,j}\}$ tel que $dom_l^{i,j}$ n'est pas marqué
Marquer $dom_l^{i,j}$ à Niv
Pour chaque $X_{hg,sh} \in X$:
Si il existe des contraintes entre $X_{hg,sh}$ et $X_{hi,sj}$ alors:
$Q \longleftarrow Q \cup (X_{hg,sh}, X_{hi,sj})$
Si Propager($X \backslash \{X_{hi,sj}\}$,$Niv$) ET Rechercher($X \backslash \{X_{hi,sj}\}$,$Niv$ + 1) ET $\Phi(X)$ ne se dégrade pas alors:
retourner VRAI
Sinon:
Solution ⟵ Solution − $(X_{hi,sj}, m_l^{i,j})$
Restaurer(X,Niv)
retourner FAUX
FinProcédure
Et finalement, voici l'algorithme de backtracking raffiné MAC.

Backtracking raffiné MAC

Procédure BacktrackingMAC($\langle X, D, C, S, \Phi \rangle$):

Pour chaque $X_{hi,sj} \in X$:
Pour chaque $m_l^{i,j} \in M$:
Enlever le marquage de $dom_l^{i,j}$
Solution ⟵ ∅
$Q \longleftarrow \emptyset$
Pour chaque $X_{hi,sj} \in X$:
Pour chaque $X_{hg,sh} \in X$:
Si il existe une contrainte entre $X_{hi,sj}$ et $X_{hg,sh}$ alors:
$Q \longleftarrow Q \cup (X_{hg,sh}, X_{hi,sj})$
Si Propager(X,0) alors retourner Rechercher(X,1)
Sinon retourner FAUX
FinProcédure

L'algorithme de backtracking raffiné MAC, détaillé par la procédure BacktrackingMAC ci-dessus, effectue les initialisations nécessaires et commence la recherche de solution. Plus précisément, la procédure de recherche de solution instancie une des variables et détermine le succès de l'instanciation en utilisant la procédure de propagation de contraintes et la valuation du problème. Si l'instanciation réussie alors la procédure de recherche de solution est appelée

de nouveau récursivement. Sinon elle revient en arrière. Notons aussi que c'est bien la procédure de propagation de contraintes qui va vérifier les contraintes entre la variable courante, les anciennes et les futures variables.

3.3 Discussions

Le VCSP offre bien une structure générique et compacte pour modéliser le problème de l'emploi du temps en comparaison avec le modèle d'optimisation linéaire en nombre entier binaire. Il a aussi l'avantage de pouvoir traduire une large gamme de contraintes, et notamment non linéaires, par rapport à ce dernier modèle. Toutefois, la performance de l'algorithme de backtracking, même dans ses versions raffinées, dépend de la structure du graphe sous-jacent au problème modélisé ; ce qui dans certains cas va limiter le gain de performance [3].

L'algorithme de backtracking borné, tout en associant une technique de résolution comme le branch and bound, exploite cette structure de graphe sous-jacent au problème. Cette méthode mixte bénéficie alors de l'efficience des algorithmes énumératives tout en fournissant une garantie de temps d'exécution borné et optimal [11]. Ce qui fera l'objet de nos futurs travaux de recherche.

En intelligence artificielle, les techniques de résolution basées sur les algorithmes génétiques sont aussi utilisées pour résoudre le problème de l'emploi du temps. Bien que ces techniques soient généralement efficientes, elles ne fournissent pas la garantie d'une résolution exacte [2, 6, 12]. L'association de ces dernières techniques de résolution au backtracking fera aussi l'objet de nos prochains travaux de recherche.

IV. Conclusion

Nous avons vu que le problème de satisfaction de contraintes valuée offre un cadre générique et compact pour modéliser le problème de l'emploi du temps. Il a notamment l'avantage de fournir un formalisme riche pour modéliser les contraintes de ce dernier. Toutefois, l'algorithme de backtracking, qui permet de le résoudre, ainsi que ces raffinements, peuvent s'avérer inefficients dans certains cas. En effet, la performance de l'algorithme dépend en grande partie de la structure de graphe sous-jacent au problème à résoudre. Bien que les techniques de résolution issues de l'intelligence artificielle comme l'algorithme génétique sont plus efficientes pour résoudre, entre autres, le problème de l'emploi du temps, ces techniques ne garantissent pas toujours une résolution exacte. Ce qui nous amènera à considérer l'association des techniques de résolution exacte (backtracking et consort) et les techniques plus performantes de l'intelligence artificielle (algorithme génétique, recherche tabou et colonie de fourmis) dans nos prochaines recherches.

V. Références

[1] R. Acosta-Amado, M. V. Marulanda, «An Integer Programming Model for The Academic Timetabling Problem», Proceedings of the 2013 ISERC, 2013.

[2] A. Ansari, S. Bojewar, «Genetic Algorithm to Generate the Automatic Time-Table – An Over View», IJRITCC, vol. 2, 2014.

[3] S. C. Brailsford, C. N. Potts, B. M. Smith, «Constraint satisfaction problems: Algorithms and applications», European Journal of Operational Research, vol. 119, 1999.

[4] G. R. Filho, L. A. N. Lorena, «An Integer Programming Model for the School Timetabling Problem», 2006.

[5] K. Ghédira, «Constraint Satisfaction Problems : CSP Formalisms and Techniques», Wiley, 2013.

[6] S. K. Jha, «Exam Timetabling Problem Using Genetic Algorithme», IJRET, vol. 3, 2014.

[7] S. Kristiansen, M. Sørensen, T. R. Stidsen, «Integer Programming for the Generalized High School Timetabling Problem», Journal of Scheduling, vol. 18, 2015.

[8] V. Kumar, «Algorithms for Constraint Satisfaction Problems: A Survey», AI MAGAZINE, vol. 13, 1992.

[9] V. Pereira, H. G. Costa, «Linear Integer Model for the Course Timetabling Problem of a Faculty in Rio de Janeiro», Advances in Operations Research, vol. 2016, 2016.

[10] T. Schiex, H. Fargier, G. Verfaillie, «Valued Constraint Satisfaction Problems : Hard and Easy Problems», Proceedings of the 14th IJCAI, vol. 1, 1995.

[11] C. Terrioux, P. Jégou, «Bounded Backtracking for the Valued Constraint Satisfaction Problems», Proceedings of the 9th ICPPCP, 2003.

[12] S. Timilsina, R. Negi, Y. Khurana, J. Seth, «Genetically Evolved Solution to Timetable Scheduling Problem», IJCA, vol. 114, 2015.

[13] L. Zhang, S. K. Lau, «Constructing University Timetable using Constraint Satisfaction Programming Approach», Proceedings of the 2005 CIMCA-IAWTIC, 2005.