Problèmes d'optimisation sous contraintes : Encadrement ANYTIME de l'optimum

Simon DE GIVRY, Gérard VERFAILLIE
CERT-ONERA, 2, av. E. Belin, 31400 Toulouse
{degivry,verfaillie}@cert.fr

Résumé

Dans cet article, nous décrivons une méthode dédiée à la résolution ANYTIME de problèmes d'optimisation sous contraintes, et plus particulièrement de problèmes additifs. Alors que les méthodes classiques, qu'elles soient exactes ou inexactes, ne fournissent en cas d'interruption qu'une borne supérieure de l'optimum, la méthode proposée, reposant sur un enchaînement de simplifications, fournit un encadrement de l'optimum, dont la précision croît avec le temps : une borne supérieure, correspondant à la meilleure solution trouvée et une borne inférieure indiquant la distance maximale à l'optimum.

Introduction

Les problèmes de satisfaction de contraintes valuées (VCSP) [SFV95] offrent un cadre général pour la modélisation des nombreux problèmes d'optimisation discrète rencontrés en Intelligence Artificielle. Parmi eux, les problèmes additifs sont de loin les plus utilisés. Ce sont aussi les plus difficiles à résoudre.

Si les méthodes exactes (recherche arborescente) sont capables de produire et surtout de prouver l'optimum, les solutions qu'elles fournissent en temps limité sont souvent de qualité médiocre. A l'inverse, si les méthodes inexactes (recherche locale) fournissent, en général, rapidement des solutions de bonne qualité, elles ne permettent pas de prouver l'optimum.

Nous situant dans un contexte temps réel, où l'algorithme d'optimisation peut être interrompu à tout moment [DB88, Zil96, AdG95], nous proposons une méthode capable de fournir à tout moment un encadrement de l'optimum : la borne supérieure est la qualité de la meilleure solution trouvée jusqu'alors et la borne inférieure indique la distance maximale à l'optimum. Avec le temps, la précision de cet encadrement croît jusqu'à l'obtention et la preuve de l'optimum.

Cette méthode repose sur la résolution de simplifications par des méthodes exactes et l'utilisation de leurs résultats pour encadrer l'optimum du problème cible. Avec le temps, les simplifications utilisées sont de moins en moins grossières, mais de plus en plus complexes. Celles que nous avons expérimentées sont la relaxation des contraintes par niveaux et le changement de structure de valuation.

1 Problèmes de Satisfaction de Contraintes Valuées

Les VCSP [SFV95] sont une extension du formalisme CSP [Mac92], qui permet de traiter les problèmes sur-contraints et les préférences entre solutions. Elle consiste à ajouter à un CSP une structure et une fonction de valuation.

La structure de valuation S est un triplet $(E, \succ, \otimes)$. E est un ensemble de valuation totalement ordonné par $\succ$, comprenant un élément minimum $\bot$ et un élément maximum $\top$. $\otimes$ est un opérateur d'agrégation commutatif, associatif et monotone, ayant $\bot$ comme élément neutre et $\top$ comme élément absorbant.

La fonction de valuation φ associe à chaque contrainte une valuation dans E.

Un VCSP est donc un quintuplet (V, D, C, S, φ), V étant l'ensemble des variables, D l'ensemble des domaines, C l'ensemble des contraintes, S la structure de valuation et φ la fonction de valuation.

La valuation d'une affectation complète A des variables est le résultat de l'agrégation des valuations des contraintes insatisfaites : $\mathcal{V}_{\mathcal{P}}(A) = \bigotimes_{\substack{c \in C \\ c \text{ insatisfaite dans } A}} \varphi(c)$. L'objectif est de produire une affectation de valuation minimum.

Différentes classes de VCSP ont été proposées. La figure 1 montre la structure de valuation utilisée par chacune de ces classes.

Classe	Notation	E	$\otimes$	$\bot$	$\top$	$\succ$
CSP classiques	$\wedge$-VCSP	$\{\text{vrai}, \text{faux}\}$	$\wedge$	vrai	faux	faux $\succ$ vrai
VCSP possibilistes	max-VCSP	$\overline{\mathcal{N}}$	max	0	$+\infty$	$>$
VCSP lexicographiques	lex-VCSP	$\overline{\mathcal{N}}^*$	$\cup$	$\emptyset$	$\{+\infty\}$	$>^*$
VCSP additifs	Σ-VCSP	$\overline{\mathcal{N}}$	$+$	0	$+\infty$	$>$

Figure 1: Différentes classes de VCSP

$\overline{\mathcal{N}}$ désigne l'ensemble des entiers naturels auquel on a ajouté l'élément $+\infty$. $\overline{\mathcal{N}}^*$ désigne l'ensemble des multi-ensembles [1] d'éléments de $\overline{\mathcal{N}}$. $>^*$ désigne l'opérateur de comparaison lexicographique [2] sur ce multi-ensemble.

D'un point de vue algorithmique, il apparaît que les propriétés de l'opérateur d'agrégation jouent un rôle essentiel. S'il est idempotent [3] (cas des VCSP possibilistes), les techniques de renforcement de cohérence utilisées dans le cadre des CSP classiques peuvent être étendues. S'il ne l'est pas (cas des VCSP lexicographiques et additifs), ces techniques ne peuvent pas être aussi facilement étendues.

Ceci est confirmé par les résultats expérimentaux qui montrent une nette différence de complexité entre les VCSP possibilistes, d'une part, et les VCSP lexicographiques et additifs, d'autre part. Malheureusement, nombre de problèmes réels font appel à des VCSP additifs.

Nous considérons par la suite que le problème cible à résoudre est un VCSP additif, utilisant les niveaux de valuation $l_1 \ldots l_i \ldots l_k$ ordonnés de façon croissante.

2 Raisonnement sur des simplifications

La méthode que nous proposons repose sur la résolution, grâce à des méthodes exactes, de simplifications du problème cible et l'utilisation de leurs résultats pour encadrer son optimum.

[1] Un multi-ensemble est un ensemble qui peut contenir plusieurs fois le même élément.

[2] Soit E un ensemble totalement ordonné par $\succ$. Soit E^* l'ensemble des multi-ensembles d'éléments de E. L'ordre $\succ^*$ sur E^*, induit par $\succ$, peut être défini de la façon suivante : soient $a, b \in E^*$, soit m l'élément maximum de $a \cup b$ selon $\succ$, soient a_m et b_m les sous-ensembles de a et b constitués des éléments égaux à m, $a \succ^* b \Leftrightarrow [|a_m| > |b_m|] \vee [(|a_m| = |b_m|) \wedge (a - a_m \succ^* b - b_m)]$.

[3] Un opérateur $\otimes$ est idempotent ssi $\forall a \in E, a \otimes a = a$.

2.1 Simplifications considérées

Un premier type de simplification consiste à éliminer les contraintes de plus faible valuation. La relaxation de niveau i, notée $\mathcal{P}_i$, consiste à ne retenir que les contraintes de valuation supérieure ou égale à l_i. Un second type de simplification consiste à changer de structure de valuation. Les problèmes $\mathcal{P}^s$, $s \in \{\wedge, \max, \text{lex}, \Sigma\}$ représentent le problème $\mathcal{P}$ à résoudre dans différentes structures de valuation. La combinaison des deux donne des simplifications notées $\mathcal{P}_i^s$.

Le problème cible est le problème $\mathcal{P}_1^\Sigma$. A noter que les variables, les domaines et la fonction de valuation ne sont pas modifiés dans les simplifications considérées.

2.2 Encadrement de l'optimum

Soient $\mathcal{P}$ et $\mathcal{P}'$ deux problèmes ayant mêmes ensembles de variables et de domaines. Montrons comment la résolution du problème $\mathcal{P}'$ permet d'encadrer l'optimum $\mathcal{V}_\mathcal{P}^{\text{opt}}$ du problème $\mathcal{P}$.

Borne supérieure De façon évidente, toute affectation complète A de $\mathcal{P}'$, évaluée dans $\mathcal{P}$, fournit une borne supérieure de l'optimum de $\mathcal{P}$:

$$\forall A \text{ , } \textit{affectation complète} \text{ , } \mathcal{V}_\mathcal{P}(A) \succeq_\mathcal{P} \mathcal{V}_\mathcal{P}^{\text{opt}} \tag{1}$$

Borne inférieure L'obtention d'une borne inférieure s'appuie sur la notion de fonction de transfert :

Définition 1 *Une fonction $f_{\mathcal{P}'\mathcal{P}}$ de E' dans E est dite fonction de transfert ssi*

$$\forall A \text{ , } \textit{affectation complète} \text{ , } \forall \nu' \text{ , } \mathcal{V}_{\mathcal{P}'}(A) \succeq_{\mathcal{P}'} \nu' \Rightarrow \mathcal{V}_\mathcal{P}(A) \succeq_\mathcal{P} f_{\mathcal{P}'\mathcal{P}}(\nu')$$

On peut prouver simplement le théorème suivant :

Théorème 1 *Soit $f_{\mathcal{P}'\mathcal{P}}$ une fonction de transfert* [4]*. L'optimum de $\mathcal{P}'$ permet d'obtenir une borne inférieure de l'optimum de $\mathcal{P}$:*

$$\mathcal{V}_\mathcal{P}^{\text{opt}} \succeq_\mathcal{P} f_{\mathcal{P}'\mathcal{P}}(\mathcal{V}_{\mathcal{P}'}^{\text{opt}})$$

Dans le cadre des simplifications considérées, on peut mettre en évidence les fonctions de transfert suivantes :

$$\left\{ \begin{array}{llcl} \forall i \, \forall j \text{ , } i < j \text{ , } \forall s \in \{\wedge, \max, \text{lex}, \Sigma\}, & f_{\mathcal{P}_j^s \mathcal{P}_i^s}(\nu) & = & \nu \\ \forall i \, \forall j \text{ , } i < j \text{ , } \forall s \in \{\max, \text{lex}, \Sigma\}, & f_{\mathcal{P}_j^\wedge \mathcal{P}_i^s}(\text{vrai}) & = & 0 \text{ , } f_{\mathcal{P}_j^\wedge \mathcal{P}_i^s}(\text{faux}) = l_j \\ \forall i \, \forall j \text{ , } i < j \text{ , } \forall s \in \{\text{lex}, \Sigma\}, & f_{\mathcal{P}_j^{\max} \mathcal{P}_i^s}(\nu) & = & \nu \\ \forall i \, \forall j \text{ , } i < j \text{ , } & f_{\mathcal{P}_j^{\text{lex}} \mathcal{P}_i^\Sigma}(m_j, \ldots, m_k) & = & \textit{lexadd}(m_j, \ldots, m_k) \end{array} \right. \tag{2}$$

Soit n_i le nombre total de contraintes et m_i le nombre de contraintes insatisfaites pour un niveau i. La fonction *lexadd* se définit de façon récursive :

$$\left[\begin{array}{lcl} \textit{lexadd}(m_j, \ldots, m_k) & = & \texttt{si } m_k = n_k \texttt{ alors } \quad n_k \times l_k + \textit{lexadd}(m_j, \ldots, m_{k-1}) \\ & & \texttt{sinon } \quad \min[(m_k + 1) \times l_k \text{ , } m_k \times l_k + \textit{lexadd}(m_j, \ldots, m_{k-1})] \\ \textit{lexadd}() & = & 0 \end{array} \right.$$

---

[4] Une telle fonction existe toujours. Par exemple, $\forall \nu' \in E' \ : \ f_{\mathcal{P}'\mathcal{P}}(\nu') = \bot$.

A noter qu'il existe d'autres méthodes permettant d'obtenir une borne inférieure. Citons :

- la méthode *Iterative Deepening* [Kor85] qui résout successivement le même problème avec une borne supérieure initiale croissante.
- la suppression ou la modification de contraintes de façon à obtenir des structures particulières plus simples (exemples: arbre de contraintes, interchangeabilité des valeurs [Fre91]).
- la relaxation de la contrainte d'intégrité, utilisée en programmation linéaire en nombres entiers.

2.3 Résolution plus simple

Utiliser des simplifications pour encadrer l'optimum du problème cible n'est intéressant que si leur résolution est nettement plus simple que celle du problème cible. Les expérimentations montrent un temps de résolution croissant suivant le nombre de contraintes et la structure de valuation employée : $\wedge$-VCSP $\prec$ max-VCSP $\prec$ lex-VCSP $\prec$ Σ-VCSP. En utilisant la notion de raffinement fort définie dans [SFV95], il est possible d'obtenir une confirmation théorique, au moins partielle, de ces résultats :

Définition 2 *$\mathcal{P}$ est un raffinement fort de $\mathcal{P}'$ ssi*

$$\forall A \quad \forall A' \text{ , affectations partielles , } \mathcal{V}_{\mathcal{P}'}(A) \prec_{\mathcal{P}'} \mathcal{V}_{\mathcal{P}'}(A') \Rightarrow \mathcal{V}_{\mathcal{P}}(A) \prec_{\mathcal{P}} \mathcal{V}_{\mathcal{P}}(A')$$

On peut alors prouver le théorème suivant [5] :

Théorème 2 *Si $\mathcal{P}$ est un raffinement fort de $\mathcal{P}'$ alors l'ensemble des solutions optimales de $\mathcal{P}$ est inclus dans l'ensemble des solutions optimales de $\mathcal{P}'$ et l'arbre de recherche de $\mathcal{P}'$ est inclus dans l'arbre de recherche de $\mathcal{P}$.*

Ce théorème induit un ordre partiel sur les simplifications considérées (voir fig. 2). Une flèche d'un problème $\mathcal{P}'$ vers un problème $\mathcal{P}$ indique que $\mathcal{P}$ est un raffinement fort de $\mathcal{P}'$ et donc que la résolution de $\mathcal{P}'$ est plus simple que celle de $\mathcal{P}$.

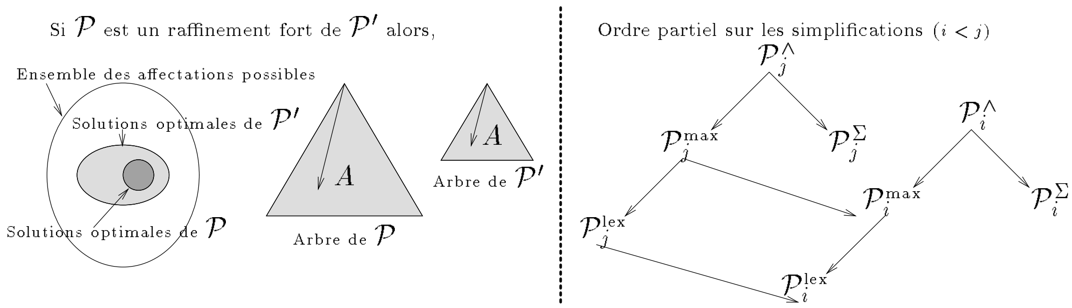

Figure 2: Ordre partiel sur les simplifications

[5] Valable pour un algorithme de type *branch and bround + backward checking*, utilisant un ordre statique sur les variables et les valeurs.

3 Enchaînement des simplifications

L'enchaînement des simplifications, permettant un encadrement ANYTIME de l'optimum, suit un ordre de complexité globalement croissante : changements de structure de valuation suivant l'ordre $\wedge$-VCSP $\prec$ max-VCSP $\prec$ lex-VCSP $\prec$ Σ-VCSP et, pour chaque structure de valuation, résolution des relaxations du niveau k (contenant uniquement les contraintes de plus forte valuation) au niveau 1 (incluant toutes les contraintes). De façon plus précise :

- en $\wedge$-VCSP, résolution immédiate du niveau 1;
- en max-VCSP, résolution par niveau, à chaque fois sous la forme d'un CSP classique, du niveau k au premier niveau où l'incohérence est rencontrée (voir l'étape 4 de la figure 3);
- en lex-VCSP, résolution par niveau, du niveau où la résolution en max-VCSP s'est arrêtée au premier niveau tel que l'on est certain que la résolution des niveaux suivants n'améliorera pas la borne inférieure du problème cible; c'est le cas lorsque :

$$lexadd(\mathcal{V}_{\mathcal{P}_i^{\mathrm{lex}}}(A^{\mathrm{opt}}_{\mathcal{P}_i^{\mathrm{lex}}})) \prec \mathcal{V}_{\mathcal{P}_i^{\Sigma}}(A^{\mathrm{opt}}_{\mathcal{P}_i^{\mathrm{lex}}}) \qquad \text{(voir l'étape 6 de la figure 3)}$$

- en Σ-VCSP, résolution par niveau, du niveau où la résolution en lex-VCSP s'est arrêtée au niveau 1; pour les niveaux précédents, on connait l'optimum puisque :

$$lexadd(\mathcal{V}_{\mathcal{P}_i^{\mathrm{lex}}}(A^{\mathrm{opt}}_{\mathcal{P}_i^{\mathrm{lex}}})) = \mathcal{V}_{\mathcal{P}_i^{\Sigma}}(A^{\mathrm{opt}}_{\mathcal{P}_i^{\mathrm{lex}}})$$

La borne inférieure du problème cible est mise à jour chaque fois que la résolution d'une simplification est terminée (équation 2) , la borne supérieure chaque fois que la résolution de la simplification courante produit une meilleure solution (équation 1). Ces deux équations sont aussi utilisées pour accélérer la résolution des simplifications intermédiaires (initialisation des bornes inférieure et supérieure). La détection des composantes connexes à chaque niveau est un autre moyen d'accélération utilisé. L'enchaînement des simplifications se termine dès que les bornes supérieure et inférieure du problème cible se rejoignent.

La figure 3 montre un exemple d'enchaînement de simplifications sur un problème contenant des contraintes impératives (de valuation $+\infty$) et des contraintes de valuation 1000,100,10 et 1. A chaque case, est associée une simplification. L'optimum obtenu y est indiqué. Dans l'exemple, à l'étape 6, l'optimum trouvé en résolvant le problème simplifié $\mathcal{P}^{\mathrm{lex}}_{10}$ vaut $\{100^7, 10^{13}\}$. La fonction de transfert de lex vers Σ (équation 2) produit une borne inférieure du problème cible [6]:

$$\begin{aligned} lexadd(13,7,0,0) &= \min[+\infty, \min[(0+1)\times 1000, 0 \times 1000 + \\ &\qquad \min[(7+1)\times 100, 7\times 100 + \min[(13+1)\times 10, 13\times 10 + 0]]]] \\ &= \min[+\infty, 1000, 800, 840, 830] \\ &= 800 \end{aligned}$$

Évolution de la borne inférieure :

étape	valuation
0	0
1	1
4	100
5	700
6	800
7	820
8	835

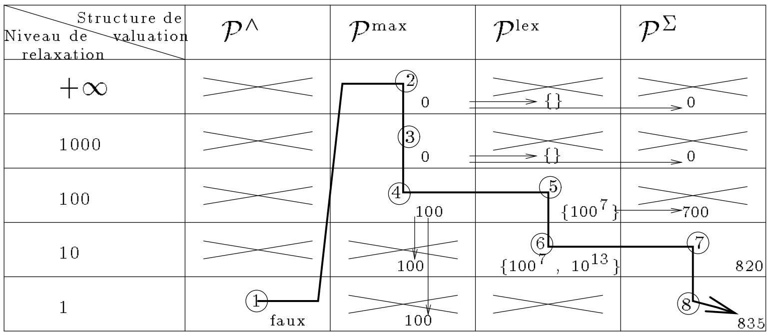

Figure 3: Exemple d'enchaînement de simplifications

[6] Dans cet ex., le nombre total de contraintes par niveau est supposé supérieur au nombre de contraintes violées.

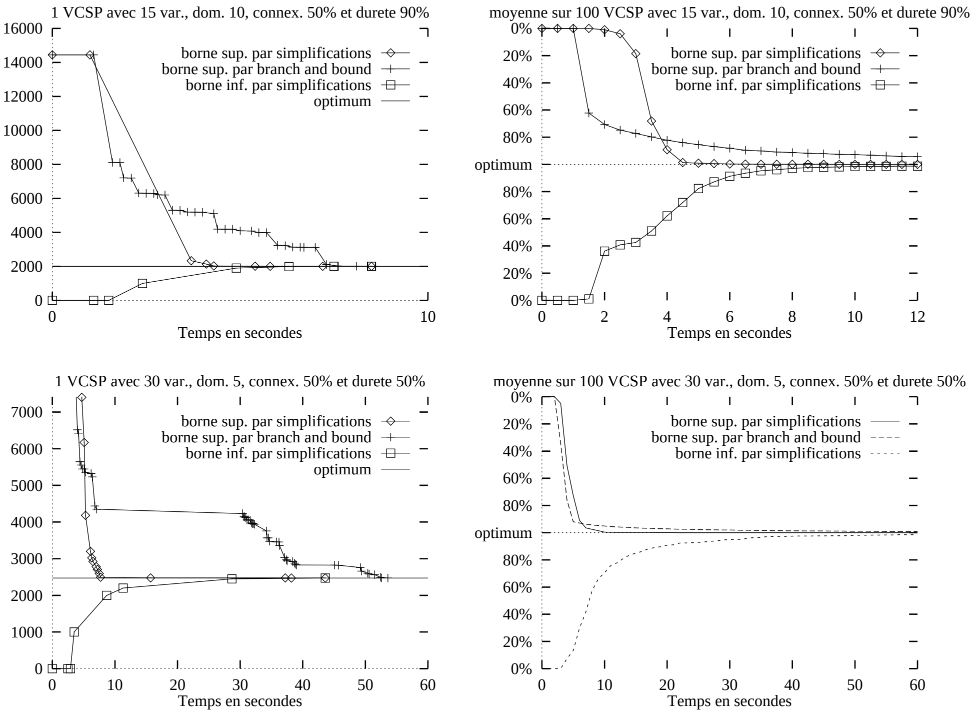

Figure 4: Profils de qualité (Σ-VCSP aléatoires binaires)

4 Expérimentations

Nous avons mené des expérimentations sur un ensemble de VCSP additifs binaires, générés de façon aléatoire, avec quatre niveaux de valuation (1, 10, 100 et 1000) et un nombre identique de contraintes par niveau.

La méthode exacte utilisée pour la résolution des simplifications est un *backtrack* ou un *branch and bound* avec *forward checking* [FW92] et ordonnancement dynamique des variables et des valeurs.

Les courbes (fig. 4) montrent l'évolution (en valeur absolue sur un problème, en % sur un ensemble de problèmes) de la borne supérieure fournie par un *branch and bound* résolvant directement le problème cible, ainsi que l'évolution des bornes inférieure et supérieure obtenues par l'enchaînement des simplifications.

On observe que l'efficacité de la méthode proposée est fortement liée à la configuration des valuations des contraintes : meilleure si les niveaux de valuation sont nombreux et espacés. On observe aussi qu'une dureté élevée des contraintes favorise une progression rapide de la borne inférieure et que les solutions obtenues dans le cadre des simplifications successives permettent, au moins sur des problèmes de petite taille, une décroissance rapide de la borne supérieure (comparable à celle obtenue grâce à des méthodes de type recherche locale).

Conclusions

Cette approche reste à valider sur des problèmes réels. Parallèlement, d'autres voies sont à explorer :

- coopération avec des méthodes inexactes capables de faire décroître encore plus rapidement la borne supérieure;
- utilisation de mécanismes d'apprentissage (*nogood recording* [SV94, DV96]) pour profiter au mieux des résolutions précédentes;
- investigation d'autres types de simplifications (décomposition en sous-problèmes par élimination de contraintes);
- mise en oeuvre d'autres moyens de calcul de borne inférieure (renforcement de cohérence, algorithmes de type *best-first branch and bound* [Kor93], passage en programmation linéaire en nombres entiers, ...).

Références

[AdG95] Martin Adelantado and Simon de Givry. Reactive/Anytime Agents: Towards Intelligent Agents with Real-Time Performance. In *IJCAI'95 Workshop on Anytime Algorithms and Deliberation Scheduling*, Montreal, Canada, August 21-25 1995.

[DB88] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proc. of AAAI-88*, pages 49–54, St. Paul, MN, 1988.

[DV96] Pierre Dago and Gérard Verfaillie. Nogood Recording for Valued Constraint Satisfaction Problems. In *Proc. of ICTAI-96*, Toulouse, France, 1996.

[Fre91] Eugene C. Freuder. Eliminating Interchangeable Values In Constraint Satisfaction Problems. In *Proc. of AAAI-91*, pages 227–233, Anaheim, CA, 1991.

[FW92] Eugene C. Freuder and Richard J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.

[Kor85] Richard Korf. Depth-First Iterative Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27:97–109, 1985.

[Kor93] Richard E. Korf. Linear-space Best-first Search. *Artificial Intelligence*, 62:41–78, 1993.

[Mac92] A. Mackworth. Constraint Satisfaction. In S. Shapiro, editor, *The Encyclopedia of Artificial Intelligence*, pages 285–293. John Wiley & Sons, 1992.

[SFV95] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proc. of IJCAI-95*, pages 631–637, Montréal, Canada, 1995.

[SV94] Thomas Schiex and Gérard Verfaillie. Nogood Recording for Static and Dynamic Constraint Satisfaction Problem. *International Journal of Artificial Intelligence Tools*, 3(2):187–207, 1994.

[Zil96] Shlomo Zilberstein. Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17(3):73–83, 1996.