Réseaux de contraintes valuées

Thomas Schiex
INRA, Toulouse, France
`www-bia.inra.fr/T/schiex`

Résumé

Une des limitations du formalisme des réseaux de contraintes réside dans l'absence de prise en compte d'un critère à optimiser. L'introduction de mécanisme d'optimisation ad-hoc dans les langages de programmation (logique) par contraintes montre bien importance de cette limitation. Plusieurs formalismes, plus ou moins généraux, ont été proposés dans le but de combler cette lacune : réseaux de contraintes flous, partiels, avec semi-anneau, valués...

Je présenterai le formalisme des réseaux de contraintes valués en le motivant et en le situant vis-à-vis de quelques-uns des formalismes existants. La définition du formalisme résulte d'un compromis entre généralité et spécificité. La généralité vise à couvrir le maximum d'extensions existantes, la spécificité vise à accumuler suffisamment de propriétés pour pouvoir construire des algorithmes et des propriétés génériques. Je détaillerai donc ces propriétés et ferai un état de l'art des algorithmes qui permettent de manipuler un réseau de contraintes valuées.

Introduction

La nécessité d'ajouter un critère à optimiser au formalisme des réseaux de contraintes est évidente même si elle a été longtemps ignorée sous le prétexte qu'il est inutile de chercher une solution optimale puisqu'il suffit de chercher une solution qui convienne *i.e.* qui respecte l'ensemble des propriétés que l'on souhaite voir satisfaites et qui peuvent être exprimées sous formes de contraintes.

Il existe pourtant une différence de nature fondamentale entre une contrainte qui résulte des propriétés physiques de l'objet modélisé (ressource non partageable en ordonnancement, capacité limitée d'un bus de communication ou d'un entrepôt. . .) et une propriété que l'on souhaite voir satisfaite (dates de livraisons en ordonnancement, absence de cours après 18 heures. . .). La modélisation sous forme de contraintes de ces deux types de propriétés peut mener à un réseau sans solution. On parle de problème sur-contraint. On peut se limiter à résoudre le sous-réseau défini uniquement par les propriétés physiques. S'il est incohérent alors il faut se résigner : le problème est insoluble. Sinon, les solutions trouvées ont de bonnes chances d'être peu satisfaisantes : le problème est généralement sous-contraint.

Dans ces cas, il est envisageable d'utiliser un outil interactif pour rechercher un réseau qui soit cohérent et qui contienne les contraintes les plus importantes, et les techniques de CSP dynamiques peuvent alors être utilisées. Mais le nombre des sous-réseaux possibles et la complexité du simple problème de satisfaction rendent cette approche assez lourde et peu fiable. L'approche interactive reste toutefois intéressante lorsque le critère à optimiser est mal défini ou inexprimable formellement pour des raisons de complexité, de confidentialité, *etc.* Dans les autres cas, il est souhaitable de pouvoir exprimer formellement le critère que l'on souhaite optimiser.

Ce sont [RHZ76] qui, à ma connaissance, ont introduit pour la première fois un critère explicite dans la formulation d'un réseau de contraintes. Il s'agissait de la première formulation des réseaux de contraintes floues. Dans ce formalisme, chaque tuple t apparaissant dans une contrainte c est étiqueté par un réel noté $\mu_c(t)$ entre 0 et 1. $\mu_c(t)$ indique à quel point l'utilisation de t est possible dans une solution et est appelé degré d'appartenance du tuple à la contrainte. La qualité d'une instanciation complète est définie par $\min_{c \in C} \mu_c(t_{\downarrow X_c})$ et le problème considéré est celui de la recherche d'une instanciation de qualité maximum. Il s'agit donc d'un problème d'optimisation max-min. L'approche est intéressante car elle est homogène dans le sens où le même type d'objet est utilisé pour exprimer à la fois les contraintes et le critère tout comme en programmation linéaire.

Pour traiter ce type de réseau de contraintes, [RHZ76] définissent un algorithme de filtrage par cohérence d'arc étendu. Cet algorithme a été amélioré (en utilisant des principes qui rappellent ceux de l'algorithme α-β) dans [SF90]. Le formalisme des réseaux de contraintes floues ou max-min a été reformulé et étudié par la suite dans de nombreux travaux [Sch92, MC92, Rut94]. Il a été ensuite analysé en détail par H. Fargier [Far94] et l'essentiel des propriétés et algorithmes des CSP classiques étendus à ce cadre.

Il existe un lien très fort entre réseaux de contraintes floues et classiques qui explique le fait que l'extension des algorithmes et propriétés classiques se fasse sans trop de difficultés. Si l'on considère un réseau de contraintes floues P, il utilise un nombre de degrés d'appartenance fini k. Pour un degré d'appartenance α donné, on peut construire un réseau de contraintes classiques, noté P_α, ayant la même structure que le réseau flou original et dont chaque contrainte est obtenue à partir d'une contrainte floue c, en sélectionnant les tuples tels que $\mu_c(t) \geq \alpha$. Toute question sur le réseau flou P peut alors se ramener à un nombre limité de questions sur les CSP classiques P_α. Ainsi, pour déterminer une solution optimale de P, on peut chercher une solution de chacun des problèmes P_α. Il suffit de prendre la solution du problème P_α qui maximise α (et qui est cohérent). Une approche dichotomique permet de résoudre le problème en $O(log_2(k))$ appels à un algorithme de satisfaction classique. Cette technique peut être utilisée quasi-systématiquement pour étendre propriétés et algorithmes classiques.

Cependant, le critère exprimé par ce type de réseau est finalement assez peu expressif car il suffit qu'un tuple de degré d'appartenance α soit utilisé dans une instanciation pour que tous les tuples de degrés supérieurs à α soient considérés comme équivalents. On a parlé d'« effet de noyade ». [FLS93] propose une sophistication du critère max-min permettant d'éviter ce comportement. Il consiste à évaluer la qualité d'une solution potentielle non pas sur le degré d'appartenance $\mu_c(t_{\downarrow X_c})$ le plus faible mais sur le vecteur ordonné des degrés d'appartenance $[\mu_c(t_{\downarrow X_c})]_{c \in C}$. L'ensemble de ces vecteurs est ordonné par un ordre lexicographique. Malheureusement, aucun algorithme classique de résolution en dehors des algorithmes `BT` et `FC` n'a pu être étendu à ce critère. En particulier, il semble impossible d'étendre la propriété de cohérence d'arc ou les algorithmes de filtrage associés, même les plus simples.

Un autre type de critère a été introduit assez tôt [SH81]. Il s'agit d'un critère additif définissant un problème maintenant désigné, par analogie avec le problème MAX-SAT, sous le nom de MAX-CSP. Il s'agit simplement d'identifier une instanciation complète qui maximise le nombre de contraintes satisfaites. Il faut attendre [FW92] pour que le problème d'optimisation soit explicitement considéré et que des extensions des algorithmes tels que `BT` ou `FC` voient le jour pour traiter ce problème. Ce sont encore une fois des algorithmes de type séparation-évaluation en profondeur d'abord utilisant la meilleure solution courante comme majorant de l'optimum et une évaluation (ou minorant) dérivée des mécanismes de filtrage. L'efficacité de ces algorithmes est exécrable et seuls des problèmes de tailles très réduites peuvent être résolus dans des temps raisonnables. Le diagnostic de cette inefficacité est simple : les minorants utilisés sont de qualité trop médiocre. La voie d'amélioration évidente passe par un filtrage accru. Malheureusement, la propriété de cohérence d'arc et les algorithmes de filtrage associés ne semblent pas pouvoir s'étendre simplement.

Il y a malgré tout une classe de techniques qui s'étend systématiquement aussi bien aux réseaux de contraintes floues qu'au MAX-CSP : il s'agit des techniques dérivées de la programmation dynamique non sérielle [BB72] telles que le regroupement en hyperarbre [DP88] ou la cohérence adaptative [DP89]. Ce résultat a été popularisé par [DDP90] et découle en particulier des travaux sur l'axiomatique de la programmation dynamique de [SS88] plus particulièrement étudiée dans le cadre de l'optimisation discrète dans [She91]. Ces techniques définissent une des rares classes polynomiales pour le MAX-CSP, celle des problèmes structurés en arbre.

En dehors de ce résultat général, la situation à l'époque était quelque peu confuse : un nombre important d'extensions du cadre CSP classique avaient vu le jour et dans chacune de ces extensions, un sous-ensemble des algorithmes CSP classiques étaient adaptés. Toutes ces adaptations semblaient fortement redondantes. Le formalisme des réseaux de contraintes floues semblait très particulier dans le sens où il était le seul pour lequel l'algorithmique de filtrage par arc cohérence avait pu être étendue.

Cette situation nous a conduit à définir un formalisme général permettant d'unifier ces travaux et, peut-être, de mieux comprendre les raisons de la singularité du formalisme des réseaux de contraintes floues. Ce formalisme devait être un compromis entre généralité et spécificité. La généralité devait être suffisante pour recouvrir l'essentiel des travaux existants, et la spécificité devait permettre de disposer de suffisamment de propriétés pour construire des algorithmes et des théorèmes génériques non triviaux. Il existait bien un cadre général qui recouvrait la majorité des extensions existantes, celui des *Partial CSP* introduit dans [Fre89, FW92]. Mais sa généralité extrême rendait difficile toute analyse. C'est pourquoi nous avons défini un cadre axiomatique mieux adapté à nos besoins : les réseaux de contraintes valuées.

1 Réseaux de contraintes valuées

Le formalisme résulte d'une généralisation des travaux existants : réseaux de contraintes floues, lexicographiques, possibilistes, probabilistes, additifs… Dans ces formalismes, une étiquette, le plus souvent numérique, est associée à chaque contrainte (resp. tuple à l'intérieur des contraintes). La qualité d'une instanciation complète est alors simplement definie par la combinaison, via un opérateur dépendant du cadre, les étiquettes de toutes les contraintes violées par l'instanciation (resp. les étiquettes de tous les tuples de contraintes utilisés dans l'instanciation). Le problème habituellement considéré est la recherche d'une instanciation complète optimale (de qualité maximum ou minimum suivant l'ordre utilisé).

La définition des réseaux de contraintes valuées que nous allons donner suppose que ce sont les contraintes qui sont étiquetées. La version plus fine consistant à étiqueter les tuples à l'intérieur des contraintes n'est pas, comme l'ont montré [BFM+96, BFM+99], fondamentalement différente et est utilisée par exemple dans [Sch00a, Sch00b]. Les étiquettes seront appelées *valuations* et l'on suppose que ces valuations vérifient un certain nombre de propriétés, regroupées dans ce que nous appelons une structure de valuation :

DÉFINITION 1 *Une structure de valuation* $(E, \oplus, \succeq)$ *vérifie :*
- E *est un ensemble totalement ordonné par* $\succeq$*, muni d'un élément minimum noté* $\perp$ *et d'un élément maximum noté* $\top$.

- *E est muni d'une loi de composition interne commutative et associative notée $\oplus$, dite de combinaison, qui vérifie :*
  - *monotonie : $\forall a, b, c \in E$ tels que $a \succeq c$, on a $(a \oplus b) \succeq (c \oplus b)$.*
  - *élément neutre : $\forall a \in E, a \oplus \bot = a$;*
  - *élément absorbant*[1] *: $\forall a \in E, a \oplus \top = \top$.*

Cette structure de monoïde commutatif totalement ordonné dont l'opérateur est monotone est assez classique dans la communauté de la représentation de l'incertain (pour $E = [0, 1]$, $\oplus$ est une co-norme triangulaire [DP82]). Les éléments de l'ensemble E sont utilisés pour étiqueter les contraintes, l'opérateur $\oplus$ pour combiner les valuations de deux contraintes et l'ordre $\preceq$ pour comparer les valuations entre elles. On définit alors un réseau de contraintes valuées :

DÉFINITION 2 *Un réseau de contraintes valuées (ou VCSP) est défini par un réseau de contraintes classique $\langle X, D, C\rangle$, une structure de valuation $S = (E, \oplus, \succeq)$ et une application φ de C dans E associant une valuation à chaque contrainte du réseau. On le notera comme un quintuplet $\langle X, D, C, S, \varphi\rangle$.*

La notion habituelle de satisfaction est remplacée par une notion graduelle de valuation d'une instanciation, obtenue en combinant les valuations des contraintes violées par l'instanciation.

DÉFINITION 3 *Étant donné un réseau de contraintes valuées P et un tuple t défini sur $V \subset X$, la valuation de t dans P est définie par :*

$$v_P(t) = \bigoplus_{\substack{c \in C \\ t \text{ viole } c}} \varphi(c)$$

Les axiomes d'une structure de valuation sont assez naturels et se justifient simplement. La commutativité et l'associativité de $\oplus$ garantissent que la valuation d'une instanciation ne dépend pas de l'ordre dans lequel les combinaisons sont effectuées et la monotonie de $\oplus$ garantit que la violation d'une contrainte supplémentaire ne peut pas diminuer la valuation d'une instanciation. Elle montre également que la valuation d'une instanciation est un minorant directement utilisable dans la construction d'algorithme de type séparation-évaluation. L'élément minimum $\bot$ permet d'exprimer la satisfaction totale et l'élément $\top$ la nature impérative de certaines contraintes et l'insatisfaction totale.

On définit une notion de relaxation qui permet de faire un lien entre réseaux de contraintes valuées et réseaux de contraintes classiques.

DÉFINITION 4 *Étant donné un réseau de contraintes valuées $P = \langle X, D, C, S, \varphi\rangle$, on dira qu'un réseau de contraintes classique $P' = \langle X, D, C'\rangle$ est une relaxation de P si et seulement si $C' \subset C$.*

Une relaxation rejette donc un sous-ensemble des contraintes du problème. Cette notion est le pendant de la notion de sous-base en logique propositionnelle non-monotone.

On peut définir la valuation d'une relaxation par :

DÉFINITION 5 *Étant donné un réseau de contraintes valuées $P = \langle X, D, C, S, \varphi\rangle$ et une relaxation $\langle X, D, C'\rangle$ de P, la valuation de cette relaxation est définie par :*

$$V_P(\langle X, D, C'\rangle) = \bigoplus_{c \in C \setminus C'} \varphi(c)$$

[1] En fait, cette propriété découle des autres axiomes : $\bot$ étant élément neutre on a $(\bot \oplus \top) = \top$; $\bot$ étant minimum, on sait que $\forall a \in E, (a \oplus \top) \succeq (\bot \oplus \top) = \top$; $\top$ étant maximum, $\forall a \in E, (a \oplus \top) \preceq \top$ et donc $(a \oplus \top) = \top$.

La valuation du sommet du treillis des relaxations, le CSP $\langle X, D, C \rangle$, est évidemment $\perp$. Les valuations des autres relaxations peuvent être interprétées, dans l'esprit des *Partial CSP* de [Fre89], comme une « distance » à ce problème idéal (mais éventuellement incohérent).

Le problème central des réseaux de contraintes valuées, extension du problème de satisfaction, est un problème d'optimisation : produire une instanciation complète des variables de valuation minimum dite instanciation optimale. La propriété suivante montre que ce problème d'optimisation dans l'espace des instanciations complètes peut se ramener à un problème d'optimisation dans l'espace des relaxations cohérentes.

PROPRIÉTÉ 1 *La valuation d'une relaxation cohérente de valuation minimum est égale à la valuation d'une instanciation optimale.*

La généralité du formalisme est effectivement suffisante pour capturer l'essentiel des propositions existantes comme le montre le tableau 1[2].

Classe	E	$\oplus$	$\perp$	$\top$	$\succ$
Classiques	$\{vrai, faux\}$	$\wedge = \max$	*vrai*	*faux*	$faux \succ vrai$
Possibilistes	$[0,1]$	$\max$	0	1	$>$
Max-CSP	$\overline{\mathbb{N}}$	$+$	0	$+\infty$	$>$
Probabilistes	$[0,1]$	$x+y-xy$	0	1	$>$
Lexico.	$[0,1]^* \cup \{\top\}$	$\cup$	$\varnothing$	$\top$	lex.

TAB. 1 – Différentes classes de réseaux de contraintes valuées

Un des premiers buts des VCSP est de permettre la comparaison de ces différentes classes. Dans ce but, nous avons introduit une notion de *raffinement polynomial*, inspirée de la notion classique de transformation polynomiale en complexité. Cette notion, proche de la notion de *metric reduction* de [Kre88, Kre92], est décrite dans [Sch94, SFV95] et a permis de mettre en évidence une partition entre CSP classiques et flous d'un coté et Max-CSP, CSP lexicographiques et probabilistes d'autre part. Les premiers vérifient une propriété d'idempotence de $\oplus$, les seconds une propriété de monotonie stricte. On montre que ces deux propriétés sont incompatibles si E contient plus de deux éléments.

Les raffinements polynomiaux exhibés entre classes sont suffisamment simples pour être d'intérêt pratique et la hiérarchie de classes construite au moyen de cette notion permet d'identifier le problème MAX-CSP comme une cible d'intérêt particulier du fait de sa simplicité de formulation et de sa généralité. Il est clair également que le développement d'algorithmes spécifiques pour les CSP probabilistes ou lexicographiques a peu d'intérêt car ces problèmes se traduisent simplement en MAX-CSP. Enfin, on identifie

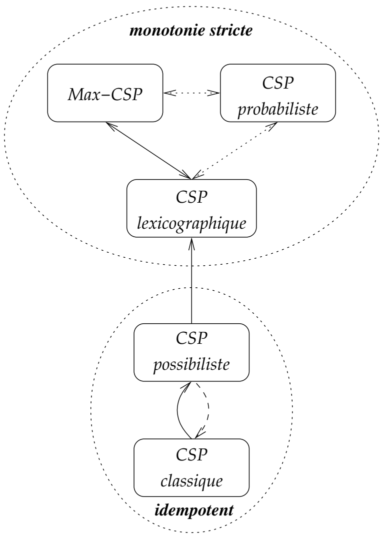

FIG. 1 – Raffinements polynomiaux

[2] $[0,1]^*$ dénote l'ensemble des multi-ensembles sur $[0,1]$ et lex. l'ordre lexicographique sur les multi-ensembles

clairement la raison de la spécificité des CSP possibilistes/flous : l'idempotence de $\oplus$, cruciale pour l'applicabilité des algorithmes de filtrage.

Un autre intérêt des réseaux de contraintes valuées est de permettre la définition d'algorithmes génériques, s'appliquant à toutes les structures. Ces algorithmes (ainsi que d'autres développés par la suite) font maintenant partie de la librairie CommonLisp LVCSP, principalement développée par M. Lemaître et L. Lobjois (CERT-ONERA) et dédiée à la résolution des réseaux de contraintes valuées (ftp ://ftp.cert.fr/pub/lemaitre/LVCSP).

Simultanément à la publication du formalisme des réseaux de contraintes valuées, [BMR95] ont défini une extension similaire utilisant un demi-anneau au lieu d'un demi-groupe, et s'appuyant donc sur deux opérateurs. Le premier est un opérateur de combinaison, le second est un opérateur de maximisation. L'utilisation d'une structure de demi-anneau est traditionnelle en programmation dynamique et a été, en particulier, utilisée pour définir les algèbres de chemins dans [Min76]. La différence essentielle entre les réseaux de contraintes valuées et les *semiring-CSP* tient dans la possibilité qu'ont ces derniers de capturer un ordre partiel (un treillis) sur l'équivalent de nos valuations. Nous avons montré dans [BFM+96, BFM+99] qu'une hypothèse d'ordre total suffit à confondre les deux formalismes.

2 Recherche d'une solution optimale

Tous les algorithmes classiques d'optimisation combinatoire, des méthodes heuristiques de recherche locale aux méthodes complètes de type programmation dynamique ou recherche arborescente (séparation-évaluation) sont *a priori* utilisables pour attaquer le problème de recherche d'une instanciation optimale. Nous ne nous sommes intéressés qu'aux méthodes avec garantie d'optimalité. La programmation dynamique reste, même dans les CSP classiques, d'usage assez rare car elle n'est raisonnablement applicable que sur des réseaux bien structurés. Nous nous sommes donc essentiellement intéressés par la suite à la construction d'algorithmes de recherche arborescente de type séparation-évaluation en utilisant une stratégie en profondeur d'abord. Par rapport à une classique stratégie en meilleur d'abord, elle a l'avantage de conserver une complexité spatiale raisonnable. La mécanique de l'algorithme est très proche de celle de l'algorithme BT.

Nous rappelons brièvement son fonctionnement (cf. algorithme 1) : l'algorithme recherche une solution dont la valuation est strictement inférieure à ub (*upper bound*), initialement fixée à $\top$. Cette valuation ub est remise à jour à chaque fois qu'une nouvelle solution est trouvée. À chaque nœud de l'arbre de recherche, l'instanciation courante t est « évaluée ». Cette évaluation consiste à calculer un minorant de la valuation de la meilleure solution qui contient t. On note lb (*lower bound*) cette fonction d'évaluation. Si $lb(t) \succeq ub$, on cesse l'extension de l'instanciation courante car elle ne peut mener à une solution de meilleure qualité que la meilleure solution connue. Étant donné un réseau de contraintes valuées et une fonction lb de calcul de minorant, l'appel $\mathsf{DFBB}(\varnothing, \top)$ retourne la valuation d'une solution optimale du réseau. L'algorithme suppose que lb est exacte sur les instanciations complètes.

L'efficacité de cet algorithme, comme celle de BT, dépend :

- de l'ordre utilisé pour instancier les variables : il faut limiter la croissance de l'arbre de recherche en utilisant les domaines de petite taille en premier et en faisant grimper le minorant le plus vite possible (*first fail principle*).
- de l'ordre sur les valeurs : il affecte l'évolution de ub. Idéalement, il serait intéressant de commencer la recherche par l'obtention d'une solution optimale ou d'une bonne solution (qui peut éventuellement être obtenue par recherche locale). Cependant, la preuve d'optimalité

Algorithme 1: Séparation-évaluation

Fonction DFBB(t : instanciation, ub : valuation) : valuation
- $v \leftarrow lb(t)$;
- **si** $v \prec ub$ **alors**
  - **si** $(|t| = n)$ **alors retourner** v;
  - **soit** i une variable non instanciée;
  - **pour chaque** $a \in d_i$ **faire**
    - $ub \leftarrow \min(ub, \text{DFBB}(t \cup \{(i,a)\}, ub))$;
  - **retourner** ub;
- **retourner** $\top$;

prenant généralement sensiblement plus de temps que la découverte d'une solution optimale, cet ordre n'est pas crucial.

– de la qualité du minorant lb : ce terme de qualité regroupe la facilité de calcul du minorant et son pouvoir de coupure. C'est le pendant du filtrage systématique effectué dans les algorithmes classiques. C'est ce dernier point qui, de loin, est le plus important.

C'est donc sur la définition de minorants de bonne qualité qu'ont porté l'essentiel des efforts. Le premier minorant qui vient à l'esprit est fourni par la valuation $v_P(t)$ de l'instanciation courante t. Le minorant ne prend en compte que les contraintes déjà violées par l'instanciation courante et l'algorithme correspondant est malheureusement d'une inefficacité redoutable.

En s'inspirant des extensions de l'algorithme FC proposées dans [FW92] sur les Max-CSP binaires, on peut construire un minorant exploitant un principe de type *forward checking* pour les réseaux de contraintes valuées binaires. On associe à chaque valeur a de chaque variable i non instanciée une valuation[3], notée fc_{ia}, qui contiendra la combinaison des valuations des contraintes qui relient i à une variable instanciée et dont la valeur est incompatible avec (i,a) (voir algorithme 2). Si cette valeur (i,a) est utilisée dans le futur, ces contraintes seront nécessairement violées. Chaque variable devant prendre une valeur, et en prenant en compte les contraintes déjà violées par t, on obtient un minorant égal à :

$$lb_{fc}(t) = v_P(t) \oplus \Big(\bigoplus_{i \text{ future}} \min_{a \in d_i} fc_{ia} \Big)$$

Pour une valeur b de la variable future j, on peut construire un minorant de la valuation optimale d'un tuple qui étend t et qui utilise (j,b) :

$$lb_{fc}^{jb}(t) = v_P(t) \oplus \Big(\bigoplus_{\substack{i \text{ future} \\ i \neq j}} \min_{a \in d_i} fc_{ia} \Big) \oplus fc_{jb}$$

Si ce minorant dépasse ub, on peut, dans l'esprit de l'algorithme FC, effacer la valeur (j,b). Ces effacements n'affectent pas le minorant global $lb_{fc}(t)$ et ne changent donc pas la taille de l'arbre exploré. Comme dans l'algorithme FC, il est nécessaire de mettre en place une mécanique de sauvegarde/restauration du contexte (domaines et compteurs fc_{ia}). Le minorant $lb_{fc}(t)$ exploite les contraintes violées par l'instanciation courante mais aussi celles qui relient les variables instanciées à des variables futures. L'efficacité s'améliore sensiblement, mais reste très décevante. La voie de l'amélioration passe clairement par la prise en compte des contraintes entre variables futures.

[3]Il faut noter que les compteurs fc_{ia} ne définissent pas des contraintes valuées unaires que l'on ajoute au problème car,

Algorithme 2: Mise à jour des fc_{ia}

Fonction FC-propagation-val(i : variable, a : valeur)
  pour chaque j non instanciée tq. $c_{ij} \in C$ **faire**
    pour chaque $b \in d_j$ tq. $((i,a),(j,b)) \notin c_{ij}$ **faire**
      $fc_{jb} \leftarrow fc_{jb} \oplus \varphi(c_{ij})$;

C'est une des propriétés de l'algorithme proposé dans [Wal94] pour les Max-CSP binaires. On suppose qu'un ordre fixe d'instanciation des variables est utilisé. Le minorant utilise des compteurs notés dac_{ia} et appelés compteurs d'arc cohérence orientée (*directed arc consistency counts* ou *DAC*). Le compteur dac_{ia} est égal à la combinaison des valuations des contraintes c_{ij} qui relient i à une variable j située après i dans l'ordre utilisé et telles qu'aucune valeur de d_j n'est un support de (i, a). Ces compteurs sont calculés en prétraitement.. Le même raisonnement que précédemment permet d'aboutir à un minorant égal à :

$$lb_{dac}(t) = lb_{fc}(t) \oplus \Big(\bigoplus_{i \text{ future}} \min_{a \in d_i} dac_{ia} \Big)$$

Il est possible, comme pour lb_{fc}, de définir un minorant lb_{dac}^{jb} associé à chaque valeur et d'effacer les valeurs dont le minorant associé dépasse ub. Les dac_{ia} étant calculés une fois pour toute, cela n'affecte pas le minorant global $lb_{dac}(t)$ et ne change pas la taille de l'arbre exploré. Le minorant $lb_{dac}(t)$ est toujours supérieur au minorant $lb_{fc}(t)$ mais avec un gain des performances qui est d'autant plus limité que l'algorithme impose l'utilisation d'un ordre statique. Cet algorithme a été ensuite amélioré par [LM96] qui ont noté que les compteurs fc_{ia} et dac_{ia} pouvaient être combinés à l'intérieur du min :

$$lb'_{dac}(t) = v_P(t) \oplus \Big(\bigoplus_{i \text{ future}} \min_{a \in d_i} (fc_{ia} \oplus dac_{ia}) \Big)$$

L'algorithme reste contraint à utiliser un ordre statique des variables.

Une nette amélioration du minorant a été obtenue dans l'algorithme *russian doll search* ou `RDS` [VLS96]. Cet algorithme, comme les précédents, nécessite l'utilisation d'un ordre statique des variables. Le minorant calculé à la profondeur i résulte de la combinaison du minorant $lb_{fc}(t)$ et de la valuation d'une solution optimale du sous-réseau formé par les variables futures et les contraintes qui les relient. Pour calculer cette valuation, on utilise l'algorithme `RDS`, récursivement. L'algorithme a un parfum de programmation dynamique et semble effectivement profiter des bonnes propriétés structurelles du réseau traité. Il a été utilisé pour résoudre optimalement, et dans des temps raisonnables (inférieurs à une heure), la quasi totalité des problèmes de programmation journalière du satellite SPOT5, sans contrainte mémoire, fournis par le CNES. Il a aussi permis, après application d'un algorithme de décomposition du graphe de contraintes, de prouver pour la première fois, l'optimalité de la meilleure solution connue[4] de l'instance 6 du problème d'affectation de fréquences du CELAR [dVS97, CdL$^+$99]. L'algorithme est disponible dans la librairie LVCSP.

[LMS99, LMSV98] ont sophistiqué l'algorithme de [Wal94, LM96] en faisant, entre autres, disparaître la restriction à un ordre statique des variables. Les compteurs dac ne sont alors plus définis

dès lors que $\oplus$ est strictement monotone, le problème ne serait plus équivalent au problème d'origine. Il s'agit de valuations externes au problème.

[4]Solution construite par A. Kolen, univ. de Maastricht.

par l'ordre d'instanciation des variables : chaque contrainte c_{ij} peut être orientée vers une de ses deux variables i ou j et les éventuelles absences de support sont comptabilisées sur la variable indiquée par cette orientation. Cette liberté supplémentaire peut être exploitée pour améliorer la force du minorant en optimisant rapidement, et après chaque instanciation, les directions qui le maximisent. Cette optimisation est réalisée par un algorithme glouton. L'optimisation complète de ce minorant ne semble pas intéressante car elle définit un problème NP-difficile [Sch98]. D'autres améliorations ont été introduites, en particulier la propagation des effacements de valeurs, comme dans l'algorithme MAC, en utilisant une mécanique de type AC6 [Bes94]. Un effacement peut permettre d'augmenter un compteur dac_{ia} qui peut permettre un nouvel effacement, *etc.* Les algorithmes PFC-RDAC et PFC-MRDAC ainsi définis offrent, par rapport à la version précédente de [LM96], des gains en performance de plusieurs ordres de grandeurs aussi bien sur des problèmes aléatoires que réels. Il s'agit en l'occurrence des mêmes problèmes d'affectation de fréquence que ceux résolus avec RDS. Les algorithmes PFC-{M}RDAC sont aussi beaucoup plus efficaces que RDS sur ces problèmes. Ces expérimentations montrent combien il est dangereux de se limiter à des tests sur des problèmes aléatoires : le meilleur algorithme sur ces problèmes est PFC-RDAC, plus simple. Sur les problèmes d'affectation de fréquences, plus complexes, utilisant des poids qui ne sont pas tous égaux à 1 et qui étiquettent les tuples et non les contraintes, PFC-MRDAC tire son épingle du jeu.

Simultanément, [AB98] ont défini un minorant paramétrique pour le MAX-CSP. [ABS99] ont montré que ce minorant (s'appuyant sur des *weighted arc consistency counts*) est une généralisation de notre minorant basé sur les *reversible DAC*. Mais les degrés de libertés offerts par ce minorant ne sont pas évidents à exploiter et les algorithmes PFC-{M}RDAC restent, à notre connaissance, les meilleurs algorithmes de résolution du MAX-CSP dans le cas binaire. Il faut noter que l'algorithme RDS, bien qu'il soit nettement dominé sur nos tests par les algorithmes PFC-{M}RDAC, reste intéressant car il fournit un minorant non trivial sur les problèmes SPOT5 ce qui n'est pas le cas des autres algorithmes.

Dans leurs travaux, [BMR95] ne se sont pas intéressés à l'extension des algorithmes d'exploration arborescente des réseaux de contraintes classiques mais à l'applicabilité de procédures de type programmation dynamique non sérielle (voir aussi [DDP90, SS88]). Un second point concernait l'extension des algorithmes de filtrage que nous allons maintenant aborder.

3 Filtrage par cohérence d'arc

Le filtrage par cohérence d'arc est un outil fondamental des CSP classiques et il est assez naturel de vouloir l'étendre aux réseaux de contraintes valuées. Il devrait permettre la construction d'un minorant plus fort que ceux présentés dans la section précédente et qui sont, pour l'essentiel, dérivés de formes affaiblies de filtrage par cohérence d'arc. Comme nous l'indiquions, la qualité d'un minorant résulte d'un compromis entre la complexité de son calcul et son pouvoir de coupure. Il nous semble clair, à l'heure actuelle, que dans ce compromis temps/force, il serait judicieux de passer plus de temps pour construire un minorant plus fort.

[BMR95] ont proposé une extension systématique de l'algorithmique de filtrage par k-cohérence aux *semiring-CSP*. Cette extension s'appuie simplement sur une généralisation des opérations de jointure et de projection. Malheureusement, l'algorithme résultant peut ne pas se terminer ou fournir un résultat non équivalent au problème d'origine. L'idempotence de l'opérateur de combinaison est une condition suffisante pour récupérer terminaison et équivalence. Il faut bien analyser l'importance de cette restriction : jointe avec une hypothèse d'ordre total, elle réduit les *semiring-CSP* aux réseaux de contraintes floues pour lesquels le filtrage par cohérence locale a déjà été défini [RHZ76, Far94]. La contribution essentielle concerne donc les cas où l'ordre est partiel et l'opérateur de combinaison idempotent : c'est le cas pour le treillis des parties d'un ensemble avec des opérateurs $\cup$ et $\cap$ ou pour le produit de plusieurs mesures floues.

Le cas des opérateurs strictement monotones, comme l'addition utilisée dans les MAX-CSP, reste donc ouvert. Nous avons présenté précédemment la notion de relaxation d'un réseau de contraintes valuées, permettant d'établir un lien avec les réseaux de contraintes classiques. Étant donné un filtrage par cohérence locale quelconque défini dans les réseaux de contraintes classiques, [SFV95] ont montré qu'il était toujours possible d'en déduire un minorant pour les réseaux de contraintes valuées. L'idée est simple : la valuation d'une solution optimale est égale à la valuation d'une relaxation optimale cohérente (voir propriété 1) et l'ensemble des relaxations cohérentes est par définition inclus dans l'ensemble des relaxations satisfaisant la propriété de cohérence locale. On peut donc conclure que :

PROPRIÉTÉ 2 *Étant donné une propriété de cohérence locale L, un minorant de la valuation d'une solution optimale est défini par la valuation d'une relaxation optimale parmi celles qui vérfient L.*

De tels minorants sont toujours définis, pour toutes les structures de valuation. Cette classe de minorants vérifie de plus deux propriétés intéressantes :

- elle est cohérente avec le comportement de L dans le cas classique. En effet, un réseau de contraintes classique vu comme un réseau valué n'a qu'une relaxation de valuation inférieure à $\top$: lui-même.
- une propriété de cohérence locale plus forte définit un minorant plus fort (mais plus cher).

Son utilisation avec des propriétés de cohérence locale correspondant par exemple au *forward checking* permet de reconstruire pour les réseaux de contraintes valuées l'extension de l'algorithme FC définie précédemment (algorithme 2). Nous attendions beaucoup du minorant induit par le filtrage par cohérence d'arc. Malheureusement, on montre que le calcul de ce minorant définit un problème NP-difficile lorsque l'opérateur $\oplus$ est strictement monotone [SFV95].

[Sch00b, Sch00a] introduit une propriété d'arc cohérence et un algorithme de filtrage associé qui satisfont pratiquement tous les critères d'une propriété de cohérence locale classique : le filtrage fournit un problème équivalent, il s'effectue en temps polynomial et retourne un problème qui satisfait la propriété de cohérence d'arc associée. L'algorithme ajoute des contraintes unaires au réseau mais modifie le reste du réseau pour conserver l'équivalence entre les problèmes. Il fonctionne pour une large sous-classe des réseaux de contraintes valuées qui utilisent un opérateur idempotent ou strictement monotone. Il n'est pas limité aux contraintes binaires et s'applique en particulier aux MAX-CSP. Seule différence avec l'arc cohérence classique : lorsque l'opérateur est strictement monotone, il peut y avoir plusieurs fermetures arc cohérentes distinctes.

Chaque fermeture arc cohérente définit un minorant qui peut prendre une valeur deux fois plus importante que celle prise par les minorants s'appuyant sur les *reversible DAC* [LMSV98] ou sur les *weighted AC counts* [AB98]. Il reste maintenant à injecter ce minorant dans un algorithme de type séparation-évaluation et à évaluer son intérêt. Indépendamment du résultat, ce travail permettra sans doute l'extension d'autres algorithmes de filtrage aux réseaux de contraintes valuées, permettant d'augmenter encore la force des minorants disponibles.

Conclusion

Les réseaux de contraintes valués permettent de représenter, dans un cadre uniforme, de nombreux problèmes d'optimisation sous contraintes. Il permettent à la fois le traitement de problèmes sur-contraints et le traitement de problèmes sous-contraint pour lesquels l'ensemble des solutions peut être ordonné par le critère induit par l'introduction de contraintes valuées.

Ces dernières années ont vu une augmentation importante de l'efficacité des algorithmes de résolution, de plusieurs ordres de grandeurs. Il reste néanmoins un travail important d'amélioration des algorithmes de résolution, en particulier des minorants utilisés dans les algorithmes de séparation-évaluation.

Références

[AB98] M. S. AFFANE et H. BENNACEUR. « A weighted arc consistency technique for Max-CSP ». Dans *Proc. of the 13th ECAI*, pages 209–213, Brighton, United Kingdom, 1998.

[ABS99] M. S. AFFANE, H. BENNACEUR, et T. SCHIEX. « Comparaison de deux minorants paramétriques pour Max-CSP ». Dans *Actes de JNPC'99*, pages 95–102, 1999.

[BB72] Umberto BERTELÉ et Francesco BRIOSHI. *Nonserial Dynamic Programming*. Academic Press, 1972.

[Bes94] Christian BESSIÈRE. « Arc-Consistency and Arc-Consistency Again ». *Artificial Intelligence*, 65 :179–190, 1994.

[BFM$^+$96] S. BISTARELLI, H. FARGIER, U. MONTANARI, F. ROSSI, T. SCHIEX, et G. VERFAILLIE. Semiring-based CSPs and Valued CSPs : Basic Properties and Comparison. Dans M. JAMPEL, E. FREUDER, et M. MAHER, éditeurs, *Over-Constrained Systems*, numéro 1106 dans LNCS, pages 111–150. Springer Verlag, 1996.

[BFM$^+$99] S. BISTARELLI, H. FARGIER, U. MONTANARI, F. ROSSI, T. SCHIEX, et G. VERFAILLIE. « Semiring-based CSPs and Valued CSPs : Frameworks, Properties and Comparison ». *Constraints*, 4 :199–240, 1999.

[BMR95] S. BISTARELLI, U. MONTANARI, et F. ROSSI. « Constraint Solving over Semirings ». Dans *Proc. of the 14th IJCAI*, Montréal, Canada, août 1995.

[CdL$^+$99] B. CABON, S. DE GIVRY, L. LOBJOIS, T. SCHIEX, et J.P. WARNERS. « Radio Link Frequency Assignment ». *Constraints Journal*, 4 :79–89, 1999.

[DDP90] R. DECHTER, A. DECHTER, et J. PEARL. Optimization in Constraint Networks. Dans R.M OLIVER et J.Q. SMITH, éditeurs, *Influence Diagrams, Belief Nets and Decision Analysis*, Chapitre 18, pages 411–425. John Wiley & Sons Ltd., 1990.

[DP82] D. DUBOIS et H. PRADE. « A class of fuzzy measures based on triangular norms. A general framework for the combination of uncertain information ». *Int. Journal of Intelligent Systems*, 8(1) :43–61, 1982.

[DP88] Rina DECHTER et Judea PEARL. « Tree Clustering Schemes for Constraint Processing ». Dans *Proc. of AAAI'88*, pages 150–154, St. Paul, MN, 1988.

[DP89] Rina DECHTER et Judea PEARL. « Tree Clustering for Constraint Networks ». *Artificial Intelligence*, 38 :353–366, 1989.

[dVS97] Simon DE GIVRY, Gérard VERFAILLIE, et Thomas SCHIEX. « Bounding the Optimum of Constraint Optimization Problems ». Dans *Proc. of CP'97*, Austria, octobre 1997.

[Far94] Hélène FARGIER. « *Problèmes de satisfaction de contraintes flexibles et application à l'ordonnancement de production* ». Thèse de doctorat, Institut de Recherche en Informatique de Toulouse (Université Paul Sabatier), Toulouse, France, juin 1994.

[FLS93] H. FARGIER, J. LANG, et T. SCHIEX. « Selecting preferred solutions in Fuzzy Constraint Satisfaction Problems ». Dans *Proc. of the 1st European Congress on Fuzzy and Intelligent Technologies*, 1993.

[Fre89] Eugene C. FREUDER. « Partial Constraint Satisfaction ». Dans *Proc. of the 11th IJCAI*, pages 278–283, Detroit, MI, 1989.

[FW92] E.C. FREUDER et R.J. WALLACE. « Partial Constraint Satisfaction ». *Artificial Intelligence*, 58 :21–70, décembre 1992.

[Kre88] M. KRENTEL. « The Complexity of Optimization Problems ». *Journal of Computer and System Sciences*, 36 :490–509, 1988.

[Kre92] Mark W. KRENTEL. « Generalizations of OptP to the polynomial hierarchy ». *Theoretical Computer Science*, 97 :183–198, 1992.

[LM96] J. LARROSA et P. MESEGUER. « Exploiting the use of DAC in Max-CSP ». Dans *Proc. of CP'96*, pages 308–322, Boston (MA), 1996.

[LMS99] J. LARROSA, P. MESEGUER, et T. SCHIEX. « Maintaining Reversible DAC for Max-CSP ». *Artificial Intelligence*, 107(1) :149–163, janvier 1999.

[LMSV98] J. LARROSA, P. MESEGUER, T. SCHIEX, et G. VERFAILLIE. « Reversible DAC and Other Improvements for Solving Max-CSP ». Dans *Proc. of AAAI'98*, Madison, WI, juillet 1998.

[MC92] R. MARTIN-CLOUAIRE. « Dealing with Soft Constraints in a Constraint Satisfaction Problem ». Dans *Proc. of the International Conference on Information Processing of Uncertainty in Knowledge based Systems*, pages 37–40, Palma de Mallorca, juillet 1992.

[Min76] M. MINOUX. « Structures algébriques généralisées des problèmes de cheminement dans les graphes ». *Recherche opérationnelle*, 10(6) :33–62, 1976.

[RHZ76] A. ROSENFELD, R. HUMMEL, et S. ZUCKER. « Scene Labeling by Relaxation Operations ». *IEEE Trans. on Systems, Man, and Cybernetics*, 6(6) :173–184, 1976.

[Rut94] Zsofia RUTTKAY. « Fuzzy constraint satisfaction ». Dans *Proc. FUZZ-IEEE'94*, Orlando, Florida, 1994.

[Sch92] T. SCHIEX. « Possibilistic Constraint Satisfaction Problems or "How to handle soft constraints ?" ». Dans *Proc. of the 8th Int. Conf. on Uncertainty in Artificial Intelligence*, Stanford, CA, juillet 1992.

[Sch94] T. SCHIEX. « Préférences et incertitudes dans les problèmes de satisfaction de Contraintes ». Rapport Technique 2/7899 DERA, CERT, 1994.

[Sch98] T. SCHIEX. « Maximizing the reversible DAC lower bound in Max-CSP is NP-hard ». Rapport Technique 1998/02, INRA, juillet 1998.

[Sch00a] T. SCHIEX. « Arc cohérence pour contraintes molles ». Dans *Actes de JNPC'00*, Marseille, juin 2000.

[Sch00b] T. SCHIEX. « Arc consistency for soft constraints ». Dans *Actes de CP'00*, Singapour, septembre 2000.

[SF90] P. SNOW et E.C. FREUDER. « Improved Relaxation and Search Methods for Approximate Constraint Satisfaction with a Maximin Criterion ». Dans *Proc. of the 8th biennal conf. of the canadian society for comput. studies of intelligence*, pages 227–230, mai 1990.

[SFV95] T. SCHIEX, H. FARGIER, et G. VERFAILLIE. « Valued Constraint Satisfaction Problems : hard and easy problems ». Dans *Proc. of the 14th IJCAI*, pages 631–637, Montréal, Canada, août 1995.

[SH81] L. SHAPIRO et R. HARALICK. « Structural descriptions and inexact matching ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3 :504–519, 1981.

[She91] P. SHENOY. Valuation-based systems for discrete optimization. Dans BONISSONE, HENRION, KANAL, et LEMMER, éditeurs, *Uncertainty in AI*. North-Holland Publishers, 1991.

[SS88] G. SHAFER et P. SHENOY. « Local computations in hyper-trees ». Working paper 201, School of business, University of Kansas, 1988.

[VLS96] G. VERFAILLIE, M. LEMAÎTRE, et T. SCHIEX. « Russian Doll Search ». Dans *Proc. of AAAI'96*, pages 181–187, Portland, OR, 1996.

[Wal94] R. WALLACE. Directed Arc Consistency Preprocessing. Dans M. MEYER, éditeur, *Selected papers from the ECAI-94 Workshop on Constraint Processing*, numéro 923 dans LNCS, pages 121–137. Springer, Berlin, 1994.