Problèmes de satisfaction de contraintes valués

Thomas Schiex*, **Hélène Fargier**† **& Gérard Verfaillie**‡

* *INRA/SBIA, Chemin de Borde-Rouge, BP 27, 31326 Castanet-Tolosan*
† *IRIT, 118 route de Narbonne, 31862 Toulouse Cedex*
‡ *CERT-ONERA, 2 av. Édouard Belin, BP 4025, 31055 Toulouse Cedex*

RÉSUMÉ. *L'expression et le traitement de préférences ou d'incertitudes dans les problèmes de satisfaction de contraintes a fait l'objet de diverses extensions du cadre CSP : CSP possibilistes, CSP probabilistes, CSP additifs... D'un point de vue purement algorithmique, de nombreux problèmes s'exprimant dans ces différents formalismes semblent de même nature et l'on souhaiterait pouvoir définir des algorithmes génériques dans un cadre englobant.*

Nous proposons une généralisation de ces formalismes qui découle d'un compromis entre la généralité (suffisante pour recouvrir l'essentiel des propositions existantes) et la spécificité (suffisante pour que l'étude d'algorithmes ait un sens). De façon plus précise, on note que certains formalismes semblent bien se prêter à l'extension des propriétés et algorithmes traditionnels des CSP alors que d'autres sont plus récalcitrants. Le cadre général proposé permet d'identifier les sources de cette différence et de montrer que certains formalismes définissent des problèmes plus difficiles, tant du point de vue théorique que pratique.

ABSTRACT. *The problem of expressing and handling preferences or uncertainties in constraint satisfaction problems has led to several extensions of the classical CSP framework: possibilistic CSP, probabilistic CSP, partial CSP... From a pure algorithmic view point, numerous problems that can be expressed in these formalisms seem extremely similar and a common abstract framework that covers all of them looks desirable to define generic algorithms and properties.*

We define an algebraic extension of the classical CSP framework that offers a reasonable compromise between generality (in order to cover most existing extensions) and specificity (in order to define significant properties and algorithms). Looking closely to existing extensions, it appears that some of them seems to lead to straightforward extensions of traditional CSP algorithms and properties while other extensions seems much more refractory. The general framework we defined enables us to identify the source of these differences and to show that some extensions define harder problems, both from a theoretical and practical point of view.

MOTS-CLÉS : CSP partiels, traitement de l'incohérence, incertitudes, préférences
KEY WORDS : Partial CSP, overconstrained CSP, uncertainties, preferences.

1 Introduction

La modélisation mathématique d'un problème combinatoire réel, préalable indispensable à sa résolution rationnelle, est souvent rendue délicate par la grande variété de caractéristiques qui doivent être prises en compte :

1. le cœur du problème peut généralement s'exprimer sous la forme de contraintes qui doivent absolument être satisfaites, et qui résultent des propriétés incontournables des objets qui interviennent : non-chevauchement dans le temps ou l'espace, lois physiques...

2. à ces contraintes « physiques » s'ajoutent des souhaits, plus ou moins forts, de voir certaines propriétés (ou contraintes) satisfaites par la solution ; par exemple, le respect de dates de livraisons en ordonnancement ;

3. enfin, il faut prendre en compte le fait que le modèle mathématique construit ne modélise pas parfaitement et complètement le problème qui se posera dans la réalité du fait de la faible précision des données ou de l'impossibilité de prendre en compte certaines d'entre elles. Ce dernier aspect est particulièrement sensible lorsque le système modélisé n'est pas aisément contrôlable (c'est par example le cas de systèmes biologiques).

Le cadre CSP (*Constraint Satisfaction Problems*) offre l'avantage de pouvoir représenter naturellement de larges classes de contraintes dans des domaines discrets et répond donc parfaitement au premier point. Il n'est pas surprenant qu'un nombre important de travaux aient tenté d'apporter des réponses aux problèmes de prise en compte d'incertitudes, de préférences et d'imprécisions sur les données. Nous citerons pour mémoire les travaux de [ROS 76, SHA 81, HER 88, FRE 89, DEC 90, SCH 92, MC 92, FRE 92, BOW 92, BRE 92, FAR 93a, FAR 94a, FAR 94b, RUT 94, GUE 94] qui offrent chacun :

– une extension du formalisme CSP permettant d'évaluer une solution potentielle en terme de coût [SHA 81, FRE 92], de possibilité [ROS 76, FAR 94a], de priorité [SCH 92], de probabilité [FAR 93a] ;

– une panoplie algorithmique pour résoudre les extensions des problèmes traditionnels du cadre CSP : établissement d'un certain niveau de cohérence, recherche de solutions...

Dans toutes ces approches, les contraintes , qu'elles soient impératives ou qu'elles expriment des préférences, sont représentées par des contraintes classiques. De plus :

1. un objet mathématique que nous appellerons *valuation* (habituellement un nombre) est associé à chaque contrainte c ou à chaque n-uplet t définissant la contrainte. Ces valuations expriment l'impact de la violation de c ou du choix de l'affectation partielle t sur la « qualité » de la solution ;

2. ces valuations sont combinées en utilisant un opérateur qui leur donne une sémantique spécifique. Ainsi, dans [SCH 92], les valuations des contraintes violées sont combinées en utilisant un opérateur $\max$, donnant ainsi aux valuations une interprétation en termes de priorité alors que dans [SHA 81], les valuations sont des nombres combinés via l'addition, avec une évidente interprétation en termes de coût.

3. le problème habituellement considéré est de trouver une affectation des variables du problèmes d'impact minimum [1].

Dans ce travail, issu de [SCH 95], plutôt que de choisir un ensemble de valuations précis et un opérateur spécifique pour les combiner, nous remarquons qu'un monoïde ordonné commutatif (un ensemble ordonné avec un opérateur binaire qui satisfait quelques propriétés) est suffisant pour capturer l'essentiel des extensions existantes des CSP. Les valuations sont prises dans l'ensemble du monoïde, combinées via son opérateur et comparées en utilisant l'ordre sur l'ensemble.

Afin de garder une relative simplicité de l'exposé, nous nous limitons au cas où une valuation est associée à chaque contrainte. L'approche plus fine dans laquelle une valuation est attachée à chaque n-uplet de chaque relation n'est pas fondamentalement différente et ne présente un intérêt esthétique d'homogénéité que lorsque les relations sont explicitement manipulées (combinaisons, intersection...).

Notre but est d'utiliser cette structure abstraite pour définir des algorithmes et propriétés générales, pour mettre en lumière différentes relations entre les extensions existantes et pour identifier celles définissant des problèmes plus difficiles, ainsi que la ou les propriétés qui sont à la source de cette difficulté.

Une approche similaire a été suivie, par exemple, dans [MIN 76, GON 90] pour les problèmes de cheminement dans les graphes, ou dans [SHA 91] pour étendre le plus largement possible le cadre d'application d'une classe d'algorithmes de propagation dans les hyper-arbres (utilisé dans les CSP [DEC 90, FRE 92]) ou enfin dans [BIS 95], qui s'appuie sur une structure de dioïde étendue pour étudier les possibles généralisations des algorithmes d'établissement de la k-cohérence.

La section 2 définit ce qu'est un CSP valué, justifie rapidement la structure algébrique utilisée et énumère quelques propriétés simples. La section 3 montre comment différentes extensions existantes peuvent se coder comme des classes de CSP valués. La section 4 donne quelques relations entre ces classes. La section 5 propose une transformation générique des algorithmes de satisfaction classiques de type «*Look-ahead*» (appelés aussi algorithmes prospectifs ou prophylactiques) en algorithmes recherchant une affectation de valuation minimum dans n'importe quel CSP valué. La section 6 présente quelques résultats expérimentaux préliminaires.

1. Les problèmes exprimés en termes de maximisation se ramènent formellement à des problèmes de minimisation via une inversion de l'ordre.

2 Vers les CSP valués

2.1 Notations, définitions

Définition 1 *Un problème de satisfaction de contraintes (ou CSP) $\mathcal{P} = (X, D, C)$ est défini par :*

- *une séquence $X = (x_1 \ldots, x_n)$ de n variables ;*
- *une séquence $D = (d_1 \ldots, d_n)$ de n domaines finis pour les variables de X. Le domaine d_i est associé à la variable x_i.*
- *une séquence $C = (c_1 \ldots, c_e)$ de e contraintes. Chaque contrainte c_i est définie par un couple (v_i, r_i) :*
  - *v_i est une séquence de variables $(x_{i_1} \ldots, x_{i_{n_i}}) \subset X$ sur lesquelles s'applique c_i. On appelle arité de c_i la longueur de la séquence v_i ;*
  - *r_i est une relation, définie par un sous-ensemble du produit cartésien $d_{i_1} \times \cdots \times d_{i_{n_i}}$ des domaines associés aux variables de v_i. Elle représente les n-uplets de valeurs autorisés pour ces variables.*

La notion d'instanciation permet de formaliser cette notion de valeur des variables :

Définition 2 (Instanciation) *Étant donné un CSP $\mathcal{P} = (X, D, C)$, on appelle instanciation A de $Y = \{x_{y_1}, \ldots, x_{y_{|Y|}}\} \subset X$ une application qui associe à chaque variable $x_{y_i} \in Y$ une valeur $\mathrm{A}\,(x_{y_i}) \in d_{y_i}$ (le domaine de x_{y_i}).*

Nous nous permettrons de dénoter les instanciations aussi bien via une application, que via le graphe de cette application (l'ensemble des couples $(x_{y_i}, \mathrm{A}\,(x_{y_i}))$) que nous noterons :

$$\mathrm{A} = \{x_{y_1} \to \mathrm{A}\,(x_{y_1}) \ldots, x_{y_{|Y|}} \to \mathrm{A}\,(x_{y_{|Y|}})\}$$

ou encore par une séquence de valeurs $(\mathrm{A}\,(x_{y_1}) \ldots, \mathrm{A}\,(x_{y_{|Y|}}))$, l'ordre des variables de Y étant implicite. Une instanciation dont le domaine est X est dite complète. Nous nous permettrons enfin d'appliquer une instanciation A à un ensemble ou une séquence v de variables. Le résultat de cette application est alors l'ensemble ou la séquence des images des éléments de v par A .

Il suffit maintenant de définir quand une instanciation satisfait une contrainte pour définir ensuite ce qu'est une solution :

Définition 3 (Satisfaction de contrainte) *Étant donné un CSP $\mathcal{P} = (X, D, C)$, une instanciation A de $Y \subset X$ et une contrainte $c_i = (v_i, r_i)$ de C telle que $v_i \subset Y$, on dit que A satisfait la contrainte c_i (noté $\mathrm{A} \models c_i$) si et seulement si :*

$$\mathrm{A}\,(v_i) \in r_i$$

En d'autres termes, une instanciation A satisfait une contrainte $c_i = (v_i, r_i)$ si et seulement si les valeurs que prennent les variables de v_i dans A apparaissent ensemble dans la relation r_i et sont donc autorisées par r_i.

Définition 4 (Solution(s) d'un CSP) *Une solution* S *de* $\mathcal{P} = (X, D, C)$ *est une instanciation des variables de* X *qui satisfait toutes les contraintes de* C*. On dit alors que l'instanciation* S *satisfait* $\mathcal{P}$ *(noté* $\mathrm{S} \models \mathcal{P}$*). L'ensemble des solutions de* $\mathcal{P}$ *sera noté* $\mathrm{S}_{\mathcal{P}}$.

Un CSP ayant au moins une solution est dit cohérent ou satisfiable ; il est incohérent ou insatisfiable sinon. Le problème de satisfiabilité d'un CSP est un problème NP-complet (il permet d'exprimer directement le problème de k-colorabilité [GAR 79]).

2.2 Un exemple

Considérons l'exemple suivant de problème de «conception gastronomique» :

> «*Vous devez organiser un repas de thèse réunissant des alsaciens et des bourguignons. Afin de satisfaire tous les esprits chauvins de la tablée, il vous faut à tout prix utiliser des produits provenant de ces deux régions. Or le plat principal ne peut être que du coq au vin (de Bourgogne) ou du baekeroffae (cuit au Riesling), les seuls vins disponibles sont un Pinot noir (vin rouge Alsacien) et un Pommard (Bourgogne rouge). Votre bon goût (?) vous empêche d'accompagner le coq au vin d'autre chose que d'un vin rouge bien charpenté, quant au baekeroffae, il faudrait utiliser un vin blanc d'Alsace.*»

Le CSP correspondant est défini par :

Variables : x_1 (vin), x_2 (plat) ;

Domaines : $d_1 = \{\text{Pinot}, \text{Pommard}\}$, $d_2 = \{\text{Coq}, \text{Baekeroffae}\}$;

Contraintes : 3 contraintes.

- **Alsace** : Les alsaciens sont satisfaits dès qu'un produit d'Alsace est utilisé :
  $c_1 = ((x_1, x_2), \{(\text{Pinot}, \text{Baekeroffae}), (\text{Pinot}, \text{Coq}), (\text{Pommard}, \text{Baekeroffae})\})$
- **Bourgogne** : Les bourguignons sont satisfaits dès qu'un produit de Bourgogne est utilisé :
  $c_2 = ((x_1, x_2), \{(\text{Pommard}, \text{Coq}), (\text{Pommard}, \text{Baekeroffae}), (\text{Pinot}, \text{Coq})\})$
- **Mélanges** : Le coq ne s'accommode que d'un vin rouge bien charpenté, le Baekeroffae d'un vin *blanc* d'Alsace : $c_3 = ((x_1, x_2), \{(\text{Pommard}, \text{Coq})\})$

Le CSP ainsi énoncé, incohérent, représente une tentative naïve de traitement d'incertitudes, ou de préférences : il n'est pas certain que votre «bon goût» soit partagé par tous, l'alsacien n'est pas sûr de venir, vous aimeriez certes le satisfaire, mais c'est tout de même le bourguignon qui sera dans le jury...

Dans tous ces cas, on a abordé le problème en imposant *a priori* comme des contraintes des propriétés dont la satisfaction n'était (peut-être) pas absolument nécessaire. Si le CSP ainsi énoncé possède une solution, alors, cette solution est idéale et l'approche naïve suffit. Sinon, il s'agit de chercher un CSP, qui soit cohérent et «aussi proche que possible» (dans un sens qui dépend de la nature du problème) du CSP exprimé. Ou, de façon équivalente, il s'agit de chercher une instanciation des variables qui minimise (dans un sens qui dépend de la nature du problème) les violations combinées des contraintes du CSP.

2.3 CSP valués

De façon générale, et afin d'exprimer le fait qu'une contrainte peut-être éventuellement violée, on associe à chaque contrainte c une *valuation* prise dans un ensemble de valuations E donné. Si plusieurs contraintes sont violées, la combinaison des valuations qui leur sont associées via une loi de composition interne (notée $\circledast$) définit l'impact cumulé de ces violations. Plus précisément, l'ensemble E est équipé de la structure suivante [2] :

Définition 5 *Une structure de valuation $(E, \succ, \circledast)$ vérifie :*

- *E est un ensemble totalement ordonné par $\succ$, muni d'un élément minimum noté $\bot$ et d'un élément maximum noté $\top$.*
- *E est muni d'une loi de composition interne commutative et associative notée $\circledast$, dite de combinaison, qui vérifie :*
  - *monotonie : $\forall a, b, c \in E$ tels que $a \succcurlyeq c$, on a $(a \circledast b) \succcurlyeq (c \circledast b)$.*
  - *élément neutre : $\forall a \in E, a \circledast \bot = a$;*
  - *élément absorbant [3] : $\forall a \in E, a \circledast \top = \top$;*

Cette structure de monoïde commutatif totalement ordonné [XUO 92] dont l'opérateur est monotone est assez classique dans la communauté de la représentation de l'incertain (pour $E = [0, 1]$, $\circledast$ est une co-norme triangulaire [DUB 82]). Dans la suite, nous supposons que les calculs de $\circledast$ et $\succ$ ont une complexité temporelle dans le pire des cas qui reste polynomiale dans la taille de leurs arguments.

Définition 6 *Un CSP valué (ou VCSP) est défini par un CSP classique (X, D, C), une structure de valuation $S = (E, \circledast, \succ)$ et une application φ de C dans E associant une valuation à chaque contrainte du CSP. On le notera comme un quintuplet (X, D, C, S, φ).*

2. Il serait envisageable de s'appuyer sur d'autres structures algébriques, plus ou moins riches :

- possibilité de remplacer la structure de demi-groupe par une structure de groupe (l'existence d'un opposé permettrait l'expression de contraintes dont la violation est souhaitée) ;
- utilisation d'un dioïde, dont la seconde opération correspondrait à la minimisation utilisée dans la définition de la valuation optimale. Il serait envisageable, comme le fait [MIN 76, GON 90] dans le cas des problèmes de recherche de plus court chemin, de remplacer cet opérateur $\min$ par d'autres opérateurs tels que $+$, $\times$ ou $\max$ et d'étudier les sémantiques des différents problèmes correspondants.

3. En fait, cette propriété découle des autres axiomes : $\bot$ étant élément neutre on a $(\bot \circledast \top) = \top$; $\bot$ étant minimum, on sait que $\forall a \in E, (a \circledast \top) \succcurlyeq (\bot \circledast \top) = \top$; $\top$ étant maximum, $\forall a \in E, (a \circledast \top) \preccurlyeq \top$ et donc $(a \circledast \top) = \top$.

Une instanciation $\mathcal{A}$ des variables X du CSP (X, D, C) étant donnée, on définit l'impact de cette instanciation en combinant les valuations de toutes les contraintes violées par l'instanciation :

Définition 7 (Valuation) *Étant donné un VCSP* $\mathcal{P} = (X, D, C, S, \varphi)$ *et une instanciation* $\mathcal{A}$ *des variables de* X*, on définit la valuation de* $\mathcal{A}$ *dans* $\mathcal{P}$ *par :*

$$\mathcal{V}_{\mathcal{P}}(\mathcal{A}) = \circledast_{\substack{c \in C \\ \mathcal{A} \textit{ viole } c}} [\varphi(c)]$$

Un CSP valué définit une distribution de valuation sur les instanciations complètes. Le calcul de la valuation d'une instanciation est polynomial. Le problème qui nous intéresse est de trouver une instanciation de valuation minimum au sens de $\succ$, dite valuation optimale. Cette valuation optimale sera aussi appelée *valuation du CSP*. Elle fournit une notion graduelle d'incohérence, de la valuation $\perp$ qui correspond à l'absence d'incohérence à la valuation $\top$ pour l'incohérence totale. De façon évidente, la détermination de la valuation d'un CSP valué est NP-difficile.

On peut étendre la notion de valuation à des instanciations partielles :

Définition 8 (Valuation locale) *Étant donné le CSP valué* $\mathcal{P} = (X, D, C, S, \varphi)$ *la valuation locale d'une instanciation quelconque* $\mathcal{A}$ *des variables de* $Y \subset X$ *est définie par :*

$$v_{\mathcal{P}}(\mathcal{A}) = \circledast_{\substack{c_i = (v_i, r_i) \in C \\ v_i \subset Y, \mathcal{A} \textit{ viole } c_i}} [\varphi(c_i)]$$

On a alors la propriété suivante :

Propriété 1 *Étant donné une instanciation* $\mathcal{A}$ *complète, et* $\mathcal{B} \subset \mathcal{A}$*, on a* $v_{\mathcal{P}}(\mathcal{B}) \preccurlyeq v_{\mathcal{P}}(\mathcal{A}) = \mathcal{V}_{\mathcal{P}}(\mathcal{A})$ *(monotonie de* $\circledast$*).*

La valuation locale constitue donc un minorant de la valuation globale, ce qui nous permettra de substituer, pour la recherche d'une instanciation de valuation minimum, un algorithme de type *séparation et évaluation* (ou *Branch and Bound*) à l'algorithme *Backtrack* des CSP classiques.

Par la suite, nous supposons implicitement que $\forall c \in C, \varphi(c) \neq \perp$. Cette hypothèse n'est pas une restriction car les contraintes annotées de $\perp$ n'ont pas d'influence sur la fonction de valuation du fait du rôle d'élément neutre de $\perp$ et il est donc possible de supprimer toute contrainte annotée de $\perp$ sans modifier le sens du VCSP.

2.4 Justification des propriétés

L'ensemble E permet d'exprimer différents niveaux de violations, l'ordre total sur E permettant de les comparer.

La commutativité et l'associativité de $\circledast$ garantit que le résultat de la combinaison de plusieurs valuations ne dépend que de l'ensemble des valuations combinées (et non de l'ordre dans lequel ces combinaisons sont effectuées). La valuation d'une instanciation ne dépend donc que de l'ensemble des contraintes qu'elle viole.

L'élément $\top$ sert à exprimer la violation inacceptable et est donc élément absorbant : il suffit que $\top$ apparaisse dans un ensemble de valuations pour que le résultat de la combinaison de ces valuations soit égal à $\top$. L'élément $\bot$ correspond à une violation sans aucune importance[4]. Son rôle d'élément neutre garantit qu'il n'affecte pas le résultat d'une combinaison[5].

Enfin, si l'on considère deux ensembles de valuations $F \subset G$, la monotonie de $\circledast$ garantit que le résultat de la combinaison des valuations de F ne sera jamais strictement supérieur au résultat de la combinaison des valuations de G. Ainsi, si une instanciation complète $\mathcal{A}'$ viole un sur-ensemble des contraintes violées par l'instanciation complète $\mathcal{A}$, la valuation de $\mathcal{A}'$ ne peut pas être plus faible que la valuation de $\mathcal{A}$.

2.5 Autres propriétés

Nous considérerons par la suite deux propriétés supplémentaires dont nous justifions rapidement l'importance (elle sera plus évidente par la suite) :

– *caractère strict de la monotonie* : $\forall a, b, c \in E$, if $(a \succ c), (b \neq \top)$ alors $(a \circledast b) \succ (c \circledast b)$.

  Alors que la simple monotonie garantit qu'une instanciation $\mathcal{A}$ qui viole un sous-ensemble strict des contraintes violées par une autre instanciation $\mathcal{A}'$ ne peut pas recevoir une valuation plus forte que celle de $\mathcal{A}'$, la monotonie stricte garantit dans ce cas (et si la valuation de $\mathcal{A}$ est différente de $\top$) que la valuation de $\mathcal{A}'$ sera strictement plus forte que celle de $\mathcal{A}$, ce qui est intuitivement raisonnable (cf. section 2.7).

– *idempotence de* $\circledast$: $\forall a \in E, a \circledast a = a$.

  C'est une propriété habituelle des CSP classiques : aucune distinction n'est faite entre une instanciation violant une contrainte et une autre violant plusieurs contraintes. Cette propriété est utilisée, en particulier, dans les algorithmes d'établissement de la k-cohérence. Dans le cas de la cohérence d'arc, la suppression d'une valeur non viable, et donc déjà implicitement interdite, revient à interdire cette valeur une seconde fois (mais explicitement). C'est l'idempotence qui garantit alors l'équivalence des deux CSP. Voir [BIS 95] pour des arguments plus formels.

4. Ces deux éléments peuvent être ajoutés à tout ensemble si ils n'existent pas déjà et leur existence peut être supposée sans perte de généralité.

5. Le choix de s'intéresser à la recherche d'une instanciation de valuation *minimum* (et non maximum) est cohérente avec ces deux dernières propriétés : on cherche à éviter les violations importantes. Cependant, il serait tout aussi possible de définir ce qu'est une structure de valuation de façon à ce que les valuations maximum deviennent intéressantes: il suffit d'inverser l'ordre de $\succ$, $\top$ étant absorbant et minimum, $\bot$ étant maximum et neutre. On cherche à maximiser la satisfaction au lieu de chercher à minimiser la violation. On retrouve la dualité entre normes et co-normes triangulaires [DUB 82].

On notera que :

1. ces deux dernières propriétés sont incompatibles dès lors que E a plus que deux éléments ($\top$ et $\bot$)[6] ;

2. que le seul opérateur $\circledast$ qui vérifie les propriétés imposées par la structure de valuation et l'idempotence est l'opérateur $\max$[7].

2.6 Les CSP possibilistes vus comme CSP valués

Pour donner corps aux définitions précédentes, considérons le cadre des CSP possibilistes défini dans [SCH 92]. Chaque contrainte est annotée avec un réel entre 0 et 1 correspondant à la priorité de la contrainte. Le problème habituellement considéré est la recherche d'une instanciation qui minimise le maximum des priorités des contraintes violées. Il correspond simplement à la structure de valuation suivante :

– l'ensemble des valuations est $E = [0, 1]$ muni de l'ordre traditionnel sur les réels. L'élément maximum est donc 1, l'élément minimum est 0 ;

– l'opérateur $\circledast$ est simplement $\max$, qui est commutatif, associatif et idempotent. On vérifie bien que $\max$ est monotone (mais pas strictement), que 0 est élément neutre et que 1 est absorbant.

Reprenons notre exemple gastronomique pour illustrer ce formalisme. Les CSP possibilistes permettant d'exprimer des préférences ou des priorités entre contraintes, nous allons supposer que la contrainte la moins prioritaire est de satisfaire les Bourguignons (c_2), alors que les deux autres contraintes (c_1 et c_3) ont la même importance.

On annote donc c_1 (Alsace) avec 0.6, c_2 (Bourgogne) avec 0.4 et c_3 avec 0.6. Il faut noter que les valeurs précises des chiffres employés n'ont *aucune* importance. Seul compte le fait que l'annotation de c_1 et de c_3 est supérieure à l'annotation de c_2. On pourrait en fait utiliser deux symboles (faible, fort).

L'instanciation $\mathrm{A}_1 = \{x_1 \to \text{Pommard}, x_2 \to \text{Coq}\}$ viole c_1 et a donc une valuation de 0.6. En fait, on observe que toutes les instanciations ont une valuation de 0.6 car il est impossible de satisfaire simultanément c_1 et c_3. La valuation du CSP est donc de 0.6.

2.7 Relaxations et CSP partiels

Les définitions précédentes peuvent être rapprochées du cadre des «*Partial Constraint Satisfaction Problems*» proposé dans [FRE 89, FRE 92]. En effet, un CSP valué définit lui aussi un treillis de relaxations équipé d'une «distance» (le terme de distance est employé dans [FRE 89] sans référence à la notion traditionnelle de distance).

6. $\bot$ étant élément neutre, on sait que $\forall a \in E, (a \circledast \bot) = a$ et donc pour tout $a, \bot \prec a \prec \top$, la monotonie stricte implique que $(a \circledast a) \succ (a \circledast \bot) = a$ alors que l'idempotence garantit que $(a \circledast a) = a$.

7. Ce résultat est bien connu dans le cadre des co-normes triangulaires. De la monotonie et de l'idempotence, on tire $\forall b \preccurlyeq a$, $(a \circledast \bot) = a \preccurlyeq (a \circledast b) \preccurlyeq a = (a \circledast a)$ et donc $\forall b \preccurlyeq a, a \circledast b = a$.

Considérons la définition suivante de la relaxation d'un CSP valué :

Définition 9 *Étant donné un CSP valué $\mathcal{P} = (X, D, C, S, \varphi)$, on dira qu'un CSP classique $\mathcal{P}' = (X, D, C')$ est une relaxation de $\mathcal{P}$ si et seulement si $C' \subset C$.*

Une relaxation rejette donc un sous-ensemble des contraintes du problème. Les relaxations sont naturellement ordonnées par la relation d'inclusion entre ensembles de contraintes. Les relaxations cohérentes et maximales pour l'inclusion sont donc les CSP classiques qui ne peuvent se rapprocher du CSP original (X, D, C) sans perdre leur propriété de cohérence.

Il est aussi possible d'ordonner totalement les relaxations d'un CSP valué en utilisant la structure de valuation du CSP valué. La valuation d'une relaxation est le résultat de la combinaison des valuations de toutes les contraintes rejetées de cette relaxation.

Définition 10 *Étant donné un CSP valué $\mathcal{P} = (X, D, C, S, \varphi)$ et une relaxation (X, D, C') de $\mathcal{P}$, la valuation de cette relaxation est définie par :*

$$\mathcal{V}_{\mathcal{P}}(X, D, C') = \underset{c \in C - C'}{\circledast} [\varphi(c)]$$

La valuation du sommet du treillis des relaxations, le CSP (X, D, C) est évidemment $\perp$. Les valuations des autres relaxations peuvent être interprétées comme une distance à ce problème idéal (mais éventuellement incohérent).

Les instanciations de X de valuation minimum sont les solutions des relaxations cohérentes les plus proches de ce problème. En effet, à toute instanciation A , on peut associer la relaxation formée par toutes les contraintes satisfaites par A . Cette relaxation est naturellement cohérente, et sa valuation est égale à la valuation de A par définition. Ceci montre que

Propriété 2 *La valuation d'une relaxation cohérente optimale est égale à la valuation d'une instanciation optimale.*

La recherche de la valuation optimale peut donc se faire, au choix, dans l'espace des instanciations ou des relaxations cohérentes.

La monotonie de la loi de combinaison $\circledast$ garantit que l'ordre sur les problèmes défini par cette distribution de valuation sur les relaxations ne sera pas en opposition avec l'ordre de l'inclusion entre relaxations, qu'il semble naturel de respecter :

Propriété 3 *Étant donné un CSP valué (X, D, C, S, φ), et (X, D, C'), (X, D, C''), deux relaxations de $\mathcal{P}$, on a :*

$$C' \subset C'' \Rightarrow \mathcal{V}_{\mathcal{P}}(X, D, C') \succcurlyeq \mathcal{V}_{\mathcal{P}}(X, D, C'')$$

Cependant, il reste possible que deux relaxations (X, D, C') et (X, D, C'') telles que $C' \subsetneq C''$ reçoivent la même valuation. Une relaxation cohérente qui n'est pas maximale au sens de l'inclusion peut donc recevoir une valuation optimale : il y perte d'information par rapport à l'ordre naturel de l'inclusion.

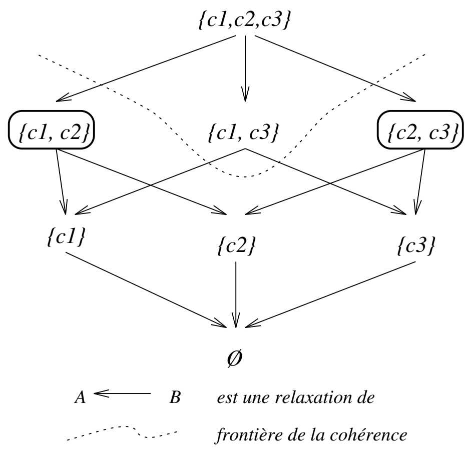

Figure 1. *Le treillis des relaxations et la frontière de la cohérence*

Nous pouvons ici justifier l'intérêt du caractère strict de la monotonie. En effet, si l'opérateur $\circledast$ est strictement monotone, la propriété précédente devient :

Propriété 4 *Étant donné un CSP valué* (X, D, C, S, φ) *et* (X, D, C'), (X, D, C''), *deux relaxations de* $\mathcal{P}$ *de valuation différente de* $\top$, *si* $\circledast$ *est strictement monotone, on a :*

$$C' \subsetneq C'' \Rightarrow \mathcal{V}_{\mathcal{P}}(X, D, C') \succ \mathcal{V}_{\mathcal{P}}(X, D, C'')$$

qui garantit que seules des relaxations maximales cohérentes pourront être optimales. La monotonie stricte est donc une propriété désirable du point de vue de la rationalité des traitements.

2.7.1 Un exemple

La figure 1 donne le treillis des relaxations de notre problème gastronomique. La frontière entre cohérence et incohérence permet d'isoler deux relaxations maximales cohérentes, offrant diverses meilleures «non-solutions» au problème d'origine :

1. Si l'on relaxe c_1, la solution $\mathrm{S}_1 = \{x_1 \to \text{Pommard}, x_2 \to \text{Coq}\}$ apparaît ;
2. Si l'on relaxe c_3, deux autres solutions apparaissent :

$$\mathrm{S}_2 = \{x_1 \to \text{Pommard}, x_2 \to \text{Baekeroffae}\}$$

$$\mathrm{S}_3 = \{x_1 \to \text{Pinot}, x_2 \to \text{Coq}\}$$

Au total, l'ordre induit par l'inclusion entre relaxations définit trois meilleures «non-solutions» du CSP d'origine sur quatre instanciations possibles. Il est clair que si l'on doit choisir parmi ces trois instanciations, il est indispensable de prendre en compte des informations complémentaires de préférence (à qui ne faut il absolument pas déplaire ?), ou de degré d'incertitude (n'y a-t-il pas une petite chance que l'on dispose d'un vin blanc d'Alsace ?) ou de coût (coût des plats, ou coût des répercussions d'un possible mécontentement. . .). C'est le rôle des valuations.

3 Quelques classes

Les exemples qui suivent montrent rapidement que l'essentiel des propositions existantes sont des classes de CSP valués. Les axiomes définissant ce qu'est une structure de valuation restant très généraux, il est certain que d'autres structures que celles citées dans la suite satisfont nos axiomes. Cependant les cadres qui suivent ont déjà été proposés par ailleurs et sont probablement les plus significatifs.

3.1 Les CSP classiques ou $\wedge$*-VCSP*

Les CSP classiques correspondent au cas où :

- E ne contient que les deux éléments $\top$ (habituellement noté *faux*) et $\perp$ (habituellement noté *vrai*) ;
- l'ordre $\succ$ est l'ordre défini par $\top \succ \perp$;
- la loi de combinaison $\circledast$ est la conjonction qui correspond au calcul du maximum dans E (elle sera notée $\wedge$). Elle vérifie les propriétés requises.

Les instanciations ne peuvent recevoir que deux valuations : $\perp$ (solutions) et $\top$ (non solutions). La fonction de combinaison $\wedge$ est stricte et idempotente (l'ensemble à deux éléments $\{\top, \perp\}$ est l'unique cas où les deux propriétés peuvent cohabiter).

3.2 Les CSP possibilistes et flous ou Max-VCSP

Ils ont été rapidement présentés dans la section 2.6. Les instanciations de valuation minimum sont celles qui minimisent l'importance de la contrainte la plus importante parmi celles qui sont violées. Cette classe peut s'interpréter dans le cadre de la théorie des possibilités : si k est attaché à une contrainte, cela signifie que la nécessité que la contrainte soit satisfaite est au moins de k. Une contrainte annotée d'un 1 doit nécessairement être satisfaite. Il n'est pas du tout nécessaire *a priori* de satisfaire une contrainte annotée d'un 0 (mais c'est tout-à-fait possible).

Si l'on affine le modèle VCSP en associant une valuation à chaque n-uplet de chaque relation comme on l'a évoqué dans l'introduction, on aboutit à un modèle équivalent aux CSP flous, à une inversion d'échelle près : un n-uplet valué à ν a un degré d'appartenance de $(1 - \nu)$ à l'ensemble des n-uplets autorisés ou relation floue. L'opérateur $\circledast$ est utilisé pendant la composition de deux relations floues. Le lecteur intéressé pourra se reporter à [ROS 76, DUB 93, FAR 93b, FAR 94a, RUT 94, BIS 95, GUE 94] pour de plus amples détails sur les FCSP ou *Fuzzy CSP*.

Si l'on reprend l'exemple de la section 2.6, on obtient le treillis des relaxations de la figure 2, chaque relaxation étant étiquetée avec sa valuation. Les relaxations cohérentes dont la valuation est *minimum* sont entourées d'un trait épais. Toutes les relaxations cohérentes sont optimales. L'opérateur idempotent $\max$ est, dans ce cas, moins informatif que l'ordre naturel de l'inclusion.

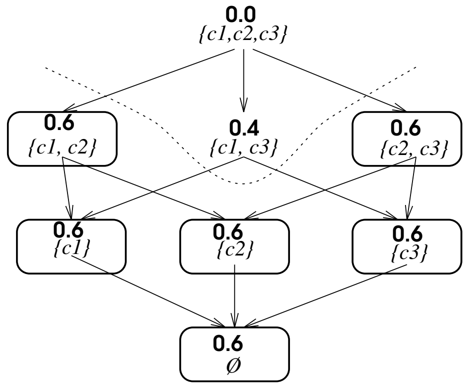

Figure 2. *Le treillis des relaxations dans les CSP possibilistes*

3.3 Les CSP à pénalités ou Σ-VCSP

Ils ont été introduits en tant que classe de « *Partial Constraint Satisfaction Problem* » dans [FRE 92]. Le but est de minimiser le nombre de contraintes violées (ou une somme pondérée). L'approche similaire en logique propositionnelle défini le problème MAX-SAT [GAR 79].

Les CSP à pénalités sont définis par :

- $E = \mathbb{N} \cup \{+\infty\}$;
- l'ordre $\succ$ est l'ordre $>$ (0 est le plus petit élément, $+\infty$ est le plus grand) ;
- l'opérateur $\circledast$ de combinaison est l'addition ;

L'opérateur $\circledast$ est monotone strict et non idempotent. Dans la version restreinte de [FRE 92], la valuation associée à une contrainte est toujours égale à 1 (il est donc impossible d'exprimer des contraintes dures).

L'exemple habituel peut être immédiatement « équipé » de pénalités : supposons que vous estimiez que votre bon goût est très important et que sa violation entraînerait une douleur esthétique de 100 (sur une échelle qui reste à normaliser). Vous estimez de la même façon que l'invité alsacien est très chauvin et pourrait subir une douleur esthétique de 50, alors que l'invité bourguignon n'est pas très sensible et pourrait subir une douleur esthétique de 30.

Il suffit d'annoter la contrainte c_1 avec 50, la contrainte c_2 avec 30 et la contrainte c_3 avec 100. Le treillis des relaxations du CSP à pénalités ainsi défini est illustré à la figure 3. La monotonie stricte garantit que les meilleures instanciations définissent une relaxation maximale cohérente. Ici, il n'y qu'une seule instanciation de valuation minimum :

$$\mathrm{A} = \{x_1 \to \text{Pommard}, x_2 \to \text{Coq}\}$$

de valuation 50 (la contrainte c_1 est la seule violée et la relaxation associée est donc $(X, D, \{c_2, c_3\})$).

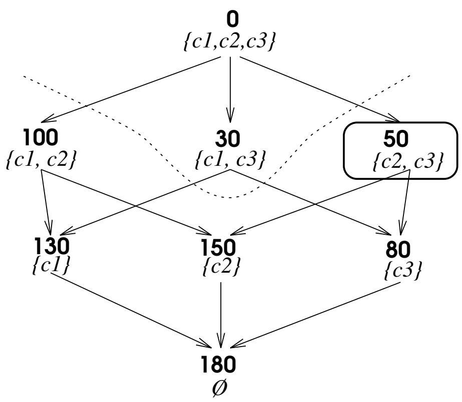

Figure 3. *Le treillis des relaxations dans les CSP à pénalités*

3.4 L'approche lexicographique

Il s'agit d'une amélioration du schéma possibiliste permettant d'exprimer des priorités tout en utilisant un opérateur monotone strict (proposé dans le cadre des logiques non-monotones dans [BEN 93] et [LEH 92] et étendu aux CSP dans [FAR 93b]). Pour cela, au lieu de juger la qualité d'une instanciation uniquement en fonction de la valuation de la contrainte violée la plus importante, on prend en compte le nombre de contraintes violées à chacun des niveaux d'importance exprimés (et pas seulement au niveau le plus important).

– l'ensemble E est formé de l'ensemble, noté $[0, 1]^*$, des multi-ensembles[8] de nombres entre 0 et 1, complété avec un élément $\top$ spécifique. Le multi-ensemble contenant deux occurrences de a est noté $\{a^2\}$;

– l'ordre $\succ$ utilisé est l'ordre lexicographique induit sur les multi-ensembles par l'ordre $\succ$ utilisé dans le cas possibiliste (le plus petit élément $\perp$ est le multi-ensemble vide) et complété de façon à ce que $\top$ soit le plus grand élément ;

– la fonction $\circledast$ est l'union de multi-ensembles étendue de façon à traiter $\top$ comme un élément absorbant.

Habituellement, ce sont des singletons qui sont associés à chaque contrainte. Mais l'utilisation de multi-ensembles plus «complexes» comme annotation a un sens : l'annotation d'une contrainte avec le multi-ensemble $\{0.6^2\}$ exprime que cette contrainte est aussi importante que deux contraintes annotées avec $\{0, 6\}$, infiniment plus importante que toute contrainte dont la valuation ne contient que des flottants strictement plus petits et infiniment moins importante que toute contrainte dont la valuation contient des flottants strictement plus grands.

8. Un ensemble dans lequel on peut répéter plusieurs fois le même élément.

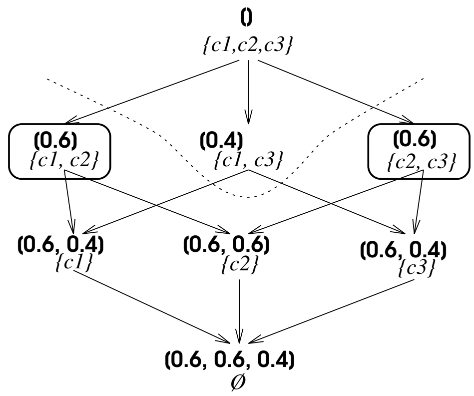

Figure 4. *Le treillis des relaxations dans les CSP lexicographiques*

À partir de l'exemple équipé avec les priorités utilisées dans le cadre possibiliste, on peut immédiatement définir un CSP lexicographique : chaque priorité est simplement transformée en un multi-ensemble ne contenant que cette priorité.

La Figure 4 illustre le treillis des relaxations du CSP ainsi défini avec les valuations de chacune des relaxations. Ici, on a une monotonie stricte et les deux meilleures relaxations sont les deux relaxations maximales cohérentes qui ont toutes deux une valuation de $\{0.6\}$.

3.5 Les CSP probabilistes ou Π*-VCSP*

La résolution prévisionnelle d'un problème réel se fait fréquemment en négligeant une partie des contraintes qui sont, *a priori* d'existence incertaine. Il est possible de prendre en compte la probabilité d'existence de contraintes incertaines dans la formulation initiale du problème comme le propose [FAR 93a, FAR 95]. L'hypothèse faite est qu'il y a indépendance entre les événements qui causent l'apparition des contraintes incertaines. La classe de VCSP obtenue est définie par :

- l'ensemble E est l'intervalle $[0, 1]$;
- l'ordre $\succ$ est défini par la relation habituelle $>$ (le plus grand élément est 1) ;
- l'opérateur $\circledast$ est définie par $x \circledast y = 1 - (1 - x).(1 - y) = x + y - x.y$. L'élément neutre est donc 0, l'élément absorbant est 1 ;

Chaque contrainte est simplement annotée avec sa probabilité d'existence p. Si deux contraintes classiques de probabilités d'existence respectives p_1 et p_2 sont violées par une instanciation, la probabilité que l'instanciation viole au moins une contrainte parmi ces deux contraintes est de $1 - (1 - p_1).(1 - p_2)$ (du fait de l'hypothèse d'indépendance des événements), d'où la définition de l'opérateur.

Reprenons encore une fois notre exemple gastronomique pour illustrer le formalisme des CSP probabilistes. La présence de bourguignons et d'alsaciens chauvins est

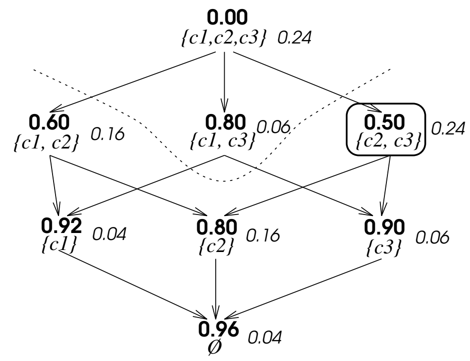

Figure 5. *Le treillis des relaxations et probabilités*

incertaine avec une probabilité de 0.8 et de 0.5 respectivement. Qui plus est, le «bon goût» traditionnel étant de plus en plus souvent remis en cause, il n'est même plus certain (probabilité 0.6) que la contrainte c_3 existe vraiment. Le treillis des relaxations est illustré dans la Figure 5, chaque relaxation étant étiquetée avec :

– en gras : la valuation de la relaxation qui correspond à la probabilité qu'une instanciation solution de cette relaxation *ne soit pas solution* du problème réel ;

– en italique, à droite, la probabilité que le problème réel soit effectivement égal à la relaxation en question.

L'instanciation de valuation minimum est $\{x_1 \rightarrow \text{Pommard}, x_2 \rightarrow \text{Coq}\}$ dont la probabilité de ne pas être solution est de 0.5. Il s'agit donc de la solution la plus «robuste»[9].

4 Relations entres classes

Chacune des classes présentées dans la section précédente a fait l'objet de développements algorithmiques spécifiques. Si la grande majorité des algorithmes et propriétés issues du cadre CSP classique a pu être adaptée au cadre flou/possibiliste (cf. [FAR 94a]), cela semble beaucoup plus délicat pour les autres formalismes, pour lesquels aucune extension de l'arc-cohérence, par exemple, n'a pu être définie jusqu'ici (cf. [BIS 95] sur le sujet). Cette différence de difficulté est confirmée ici.

9. Cette notion de «robustesse» peut-être étendue à un ensemble de solutions : un ensemble de solutions minimisant la probabilité de ne pas contenir une solution est obtenu en prenant une solution dans chacune des deux relaxations maximales cohérentes, ici l'ensemble :

$$\{\{x_1 \rightarrow \text{Pommard}, x_2 \rightarrow \text{Coq}\}, \{x_1 \rightarrow \text{Pommard}, x_2 \rightarrow \text{Baekeroffae}\}\}$$

qui a une probabilité de ne pas contenir une solution de 0.3 : c'est la probabilité de non cohérence du CSP. Cet ensemble de solutions montre en particulier qu'il est possible de choisir déjà la valeur d'une des variables (ici un Pommard comme vin) sans que cela n'affecte la probabilité de trouver une solution du problème réel (cf. [FAR 93a, FAR 95]).

4.1 Raffinement et équivalence

Afin de comparer les classes précédentes, qui s'appuient sur des structures de valuation différentes, nous introduisons les notions suivantes :

Définition 11 (Raffinement) *Un CSP valué $\mathcal{P}' = (X, D, C', S', \varphi')$ est un* raffinement *du CSP valué $\mathcal{P} = (X, D, C, S, \varphi)$ si, pour toute paire d'instanciations* A *et* B *de X telle que $\mathcal{V}_{\mathcal{P}}(\mathrm{A}) \succ \mathcal{V}_{\mathcal{P}}(\mathrm{B})$, alors $\mathcal{V}_{\mathcal{P}'}(\mathrm{A}) \succ \mathcal{V}_{\mathcal{P}'}(\mathrm{B})$.*

Le point essentiel est que si $\mathcal{P}'$ est un *raffinement* de $\mathcal{P}$, alors l'ensemble des instanciations optimales de $\mathcal{P}'$ est inclus dans l'ensemble des instanciations optimales de $\mathcal{P}$. Le problème de trouver une instanciation optimale de $\mathcal{P}$ se réduit au même problème dans $\mathcal{P}'$.

Définition 12 (Équivalence) *Deux CSP valués sont dits équivalents lorsque chacun d'entre eux est un raffinement de l'autre.*

Pour deux CSP équivalents, les valuations induisent le même ordre sur les instanciations complètes et les ensembles d'instanciations optimales sont donc identiques : le problème de recherche d'une instanciation optimale est équivalent dans les deux cas.

Afin de pouvoir comparer la difficulté de résolution dans deux structures de valuation distinctes, on introduit la notion suivante :

Définition 13 (Raffinement polynomial) *Étant donné deux structures de valuations S et S', on appelle raffinement polynomial de S en S' une fonction Φ qui :*

- *à tout VCSP $\mathcal{P} = (X, D, C, S, \varphi)$ associe le VCSP $\mathcal{P}' = (X, D, C, S', \varphi')$ tel que $\mathcal{P}'$ est un raffinement de $\mathcal{P}$;*
- *est calculable en temps polynomial.*

La notion de raffinement polynomial s'inspire donc de la notion de « transformation polynomiale » (ou « *many-one reduction* ») traditionnelle en théorie de la complexité (cf. [GAR 79, PAP 94]). Elle est adaptée aux problèmes d'optimisation dans les CSP valués plus qu'aux problèmes de décision. Cette nouvelle définition est motivée par le fait que tous nos problèmes d'optimisation définissent des problèmes de décision (Existe-t-il une instanciation de valuation inférieure à k ?) qui sont tous NP-complets et donc de complexité identique. Comme on va le voir, la notion de raffinement polynomial permet de mettre en lumière des différences entre les différentes classes de CSP valués [10].

---

10. Depuis peu, nous avons pris connaissance d'une démarche similaire en théorie de la complexité, s'appliquant dans le cadre général des problèmes d'optimisation. Ces travaux ont mené à la définition des classes OptP et s'appuient sur la notion de réduction métrique, très proche de la notion de raffinement polynomial (cf. [KRE 88, KRE 92]). Les raffinements polynomiaux qui sont présentés dans la suite permettent très facilement de construire des réductions métriques et d'utiliser les résultats de Krentel. On montre ainsi que le problème des CSP à pénalités est, dans le cas général, OptP$(O(n))$-complet, et donc dans la même classe que le voyageur de commerce. Si les valuations sont toutes égales à 1, comme le considère [FRE 92], le problème est seulement OptP$(O(\log(n)))$-complet, et dans la même classe que le problème de recherche d'une clique de cardinalité maximum.

4.2 *Raffinement polynomiaux entre classes*

L'ensemble des classes de CSP valués précédentes, présenté dans le tableau cidessous, peut être partitionné suivant la monotonie stricte ou l'idempotence de l'opérateur $\circledast$. Les CSP classiques et les CSP possibilistes forment une première classe, les CSP à pénalités, lexicographiques et probabilistes forment l'autre classe. Cette partition se retrouve au niveau des liens de raffinement polynomiaux.

Classe	E	$\circledast$	$\bot$	$\top$	$\succ$	Prop.
$\wedge$-VCSP	$\{t, f\}$	$\wedge = \max$	t	f	$f \succ t$	idem.+ m. strict.
Max-VCSP.	$[0, 1]$	$\max$	0	1	$>$	idem.
Σ-VCSP	$\overline{\mathbb{N}}$	$+$	0	$+\infty$	$>$	m. strict.
Π-VCSP	$[0, 1]$	$x + y - xy$	0	1	$>$	m. strict.
Lex-VCSP	$[0, 1]^* \cup \{\top\}$	$\cup$	$\varnothing$	$\top$	lex.	m. strict.

Classes avec un opérateur idempotent : un CSP classique n'étant qu'une forme spécifique de CSP possibiliste où seules les valuations $\top$ et $\bot$ sont utilisées, l'identité définit un raffinement polynomial entre $\wedge$-VCSP et Max-VCSP.

Classes avec un opérateur non idempotent : On considérera à part les CSP probabilistes, faisant intervenir des valuations réelles. On peut cependant noter que si l'on étend le cadre des CSP à pénalités en autorisant des pénalités dans $\mathbb{R}^+$ et non seulement dans $\mathbb{N}$, il est possible de transformer simplement un CSP probabiliste en un CSP à pénalités qui lui est équivalent : une contrainte avec une probabilité d'existence de $\varphi(c)$ sera traduite en une contrainte identique, avec une pénalité associée de $-\log(1 - \varphi(c))$. Réciproquement, un CSP à pénalités peut être traduit en CSP probabiliste : une contrainte avec une pénalité associée égale à $\varphi(c)$ sera traduite en la même contrainte avec une pénalité associée égale à $(1 - e^{-\varphi(c)})$.

Un raffinement polynomial simple existe entre CSP à pénalités et CSP lexicographiques : on choisit une fois pour toutes un élément $\alpha \in [0, 1[$, une pénalité de k finie est transformée en un multi-ensemble $\{\alpha^k\}$ contenant k occurrences d'α. La valuation $+\infty$ est transformée en l'élément désigné $\top$ de la structure de valuation des CSP lexicographiques. Le CSP lexicographique, obtenu en temps polynomial, est trivialement équivalent au CSP à pénalités original.

On peut également exhiber un raffinement polynomial des CSP lexicographiques vers les CSP à pénalités. Soit $\alpha_1, \ldots, \alpha_k$ les éléments de $]0, 1[$ qui apparaissent dans les différentes valuations du CSP lexicographique, dans un ordre croissant. Soit n_i le nombre total d'occurrences de α_i dans toutes les annotations du CSP lexicographique. On définit une fonction f traduisant chaque élément d'un multi-ensemble en une pénalité. La priorité la plus basse α_1 est traduite en $f(\alpha_1) = 1$. Inductivement, la priorité α_i est traduite en $f(\alpha_i) = f(\alpha_{i-1}) \times (n_{i-1} + 1)$[11]. Une valuation lexico-

11. De cette façon, la pénalité $f(\alpha_i)$ associée à la priorité α_i est strictement plus grande que la somme de toutes les pénalités $f(\alpha_j), j < i$. Cette propriété est trivialement satisfaite pour $i = 1$ et inductivement vérifiée.

graphique est simplement traduite en la somme des pénalités $f(\alpha_i)$ pour chaque α_i dans la valuation. La valuation $\top$ est transformée en $+\infty$. Les contraintes sont inchangées. Toutes les opérations utilisées (multiplication, addition) sont polynomiales dans la taille de leurs opérandes et la taille des opérandes reste polynomiale : si k est le nombre de niveaux de priorités utilisés dans le CSP lexicographique et ℓ le nombre maximum d'occurrences d'une priorité, alors la pénalité la plus large est en $O(\ell^k)$, un nombre dont la longueur est, pour un codage raisonnable, en $O(k.\log(\ell))$ alors que l'ensemble des valuations lexicographiques d'origine utilisaient au moins un espace en $O(k + \log(\ell))$. La transformation est donc polynomiale et on a vu que le CSP obtenu est équivalent au CSP d'origine.

Les deux cadres CSP lexicographiques et CSP à pénalités sont donc, dans un sens précis, équivalents : tout problème d'optimisation dans l'un des cadres peut se ramener en temps polynomial à un problème d'optimisation dans l'autre cadre.

Liens entre idempotents et non idempotents : enfin, il existe un raffinement polynomial simple entre CSP possibilistes et CSP lexicographiques. Le CSP lexicographique est simplement obtenu en annotant chaque contrainte avec le multi-ensemble qui ne contient que la priorité originale de la contrainte dans le CSP possibiliste.

Pour conclure, il faut noter que l'intérêt des « raffinement polynomiaux » précédents est double :

1. ils montrent que les classes de CSP valués précédentes ayant un opérateur non idempotent sont au moins aussi difficiles à résoudre que toute classe de CSP valués ayant un opérateur idempotent. La traduction de ces raffinements polynomiaux en «réductions métriques» [KRE 88] situe cet argument dans un cadre théorique plus fourni ;

2. toutes les transformations utilisées sont suffisamment simples pour que pratiquement, tout algorithme adapté à l'un des cadres probabilistes, pénalités ou lexicographiques puisse être utilisé pour trouver une instanciation optimale des CSP valués exprimés dans un autre de ces cadres ou même pour résoudre des CSP classiques ou possibilistes ;

On notera que cette distinction entre opérateur idempotent et opérateur non idempotent est également visible au niveau des classes polynomiales traditionnelles des CSP classiques : pour mémoire, la recherche d'une instanciation optimale dans des CSP valués ayant des domaines de cardinalité 2 est polynomiale quand $\circledast$ est idempotent, mais NP-difficile (MAX2SAT) pour les CSP à pénalités (et donc les CSP lexicographiques et probabilistes). Une des seules classes polynomiales qui semble s'étendre directement aux CSP valués est la classe des CSP structurés en hyper-arbre (cf. [DEC 90, SHA 91, FRE 92]).

5 Étendre les algorithmes

Dans la suite, on se limite aux CSP valués binaires (dans lesquels toutes les contraintes pèsent sur deux variables).

5.1 Établissement de la cohérence locale

Dans le cadre des CSP classiques, le problème de satisfiabilité définissant un problème NP-complet, un ensemble de propriétés de cohérence locale et d'algorithmes permettant de les établir ont été définis. En particulier, la notion de k-cohérence forte [FRE 82] définit une série de propriétés que l'on peut établir en temps polynomial. À chacune d'entre elles correspond une relaxation de qualité croissante de la notion de satisfiabilité (un CSP satisfiable ne sera pas k-incohérent).

La cohérence d'arc est certainement le niveau de cohérence locale le plus employé. Elle a été étendue aux CSP possibilistes/flous dans [ROS 76, SNO 90, SCH 92, FAR 94a]. La reformulation de la propriété dans le cadre CSP valué est alors :

Définition 14 *Un CSP valué sera dit arc-cohérent ssi (1) il existe, pour toute variable, une valeur qui définit une instanciation de valuation locale inférieure à* $\top$*, et (2) toute instanciation d'une variable peut-être étendue sur n'importe quelle autre variable en une instanciation de deux variables ayant la même valuation locale.*

Dans le cadre des CSP possibilistes/flous, des algorithmes polynomiaux qui établissent cette propriété en fournissant un CSP arc-cohérent, définissant la même distribution de valuation, ont été exhibés [ROS 76, SNO 90, SCH 92, FAR 94a]. Un minorant sur la valuation du CSP est alors facilement calculable.

Naturellement, cette définition pourrait également être utilisée dans le cas de VCSP non idempotents, mais encore faut-il pouvoir définir l'algorithme correspondant qui devra, en temps polynomial, construire un CSP arc-cohérent et définissant la même distribution de valuation. Jusqu'ici aucun algorithme de ce type n'a pu être proposé. [BIS 95] ont montré que la généralisation « naïve » des algorithmes d'établissement de la k-cohérence obtenue en utilisant le $\min$ et $\circledast$ respectivement pour la projection et la combinaison de relations ne définissent pas des algorithmes corrects dans le cas d'opérateurs non idempotents : la distribution de valuations obtenue est différente et l'algorithme peut ne pas se terminer. Il ne faut cependant pas surestimer ce résultat : le problème de généralisation des propriétés et algorithmes d'établissement de la k-cohérence reste ouvert si l'on s'autorise des modifications plus importantes de l'algorithme, des propriétés ou simplement de la notion d'équivalence de deux CSP valués (voir par exemple la définition 12).

5.2 Recherche d'instanciations optimales

À la suite des travaux de [SHA 81, SCH 92, FRE 92, DUB 93], nous considérons la généralisation des algorithmes de satisfaction de type «*look-ahead*», aussi appelés algorithmes prospectifs ou algorithmes prophylactiques. Ces algorithmes combinent une recherche arborescente de type *Backtrack* avec l'établissement d'un certain niveau de cohérence locale à chaque nœud. Quelques instances possibles ont été considérées dans [NAD 89] : algorithmes *Backtrack*, *Forward-checking*, *Really Full Look Ahead*... Nous considérons dans cette section qu'un tel algorithme est entièrement défini par le type de cohérence locale vérifié à chaque nœud de la recherche arborescente : *check-backward*, *check-forward*, arc-cohérence complète ou plus...

Dans un algorithme prospectif, une instanciation partielle, initialement vide, est étendue de façon systématique jusqu'à ce qu'une solution (une instanciation complète et cohérente) soit trouvée ou que la propriété de cohérence locale utilisée soit violée : un retour-arrière est alors effectué. La généralisation d'un tel algorithme au cadre des CSP valués s'appuie sur la transformation de la recherche arborescente de type *Backtrack* en une recherche de type «séparation/évaluation» en profondeur d'abord (*Depth First Branch and Bound*). Cette classe d'algorithme, comme l'algorithme *Backtrack*, étend systématiquement une instanciation partielle (initialement vide) jusqu'à ce que (1) une instanciation complète soit obtenue et une nouvelle «meilleure» solution soit trouvée ou (2) un minorant (*lower bound*) de l'ensemble des valuations de toutes les instanciations complètes contenant l'instanciation courante dépasse la valuation de la meilleure solution trouvée jusqu'ici : un retour-arrière est effectué. Le minorant utilisé définit l'algorithme. Nous nous proposons de dériver de façon systématique un minorant à partir de n'importe quelle propriété de cohérence locale.

Dans les CSP classiques, vus comme des CSP valués, la propriété de cohérence locale utilisée fournit directement l'évaluation : ainsi, dans l'algorithme *Really Full Look-Ahead*, l'apparition d'un domaine vide lors de l'établissement de la cohérence d'arc garantit que la valuation de toute extension de l'instanciation courante sera supérieure (et donc égale) à $\top$. Cependant, aucune généralisation des algorithmes d'établissement de la k-cohérence forte n'est disponible dans le cas d'un opérateur non idempotent [BIS 95]. Nous allons donc nous appuyer sur les notions de cohérences locales du cadre CSP *classique* et sur la notion de relaxation pour définir une famille de minorants :

Propriété 5 *Étant donné une propriété de cohérence locale L, un minorant de la valuation d'un CSP valué donné $\mathcal{P}$ est défini par la valuation d'une relaxation optimale de $\mathcal{P}$ parmi celles vérifiant la propriété L.*

En effet, la valuation du CSP valué est égale à la valuation d'une relaxation optimale cohérente et l'ensemble des relaxations cohérentes est par définition inclus dans l'ensemble des relaxations satisfaisant la propriété de cohérence locale.

Cette classe de minorants vérifie deux propriétés intéressantes :

– elle garantit que l'algorithme généralisé, appliqué à un CSP classique vu comme un CSP valué, se comportera comme l'algorithme original sur le CSP classique (en effet, un CSP classique vu comme un CSP valué n'a qu'une relaxation de valuation inférieure à $\top$: lui-même) ;

– une propriété de cohérence locale plus forte définit un meilleur minorant, conduisant à une recherche avec moins de nœuds mais éventuellement plus de calculs à chaque nœud.

5.2.1 Extension de l'algorithme Backtrack

Dans l'algorithme *Backtrack* simple, la propriété de cohérence locale utilisée est la cohérence de l'instanciation partielle courante. Le minorant que l'on peut dériver de cette propriété, dite de *check-backward*, est donc la valuation d'une relaxation optimale dans laquelle l'instanciation courante ne viole aucune contrainte.

La relaxation obtenue par le rejet des contraintes directement violées par l'instanciation courante respecte cette propriété et est optimale du fait de la monotonie de la loi de combinaison $\circledast$. Le minorant est donc défini par :

$$\underset{\substack{c \in C \\ \mathcal{A} \text{ viole } c}}{\circledast} \varphi(c)$$

Dans cette définition, une contrainte $C = (v, r)$ est dite violée par l'instanciation $\mathcal{A}$ ssi toutes les variables de la contrainte sont instanciées par $\mathcal{A}$ et que $\mathcal{A}(v) \notin r$. Ce minorant est calculable en temps polynomial. On retrouve le minorant fourni par la «valuation locale» déjà mis en évidence dans la définition 8.

En pratique, ce minorant peut être mis-à-jour incrémentalement, lorsqu'une nouvelle variable $x_i \in X$ reçoit une valeur : le minorant associé au nœud courant est obtenu en combinant par $\circledast$ le minorant associé au nœud père aux valuations de toutes les contraintes violées du fait de l'instanciation de x_i.

L'algorithme ainsi défini recouvre toutes les classes d'algorithmes *Branch and Bound* définis pour les CSP possibilistes, les CSP à pénalités...[SCH 92, FRE 92, FAR 94a, RUT 94]. Il est décrit plus précisément comme l'algorithme 1, formé de deux procédures mutuellement récursives. V désigne l'ensemble des variables non instanciées (initialement égal à X), $\mathcal{A}$ désigne l'instanciation courante (initialement vide), α représente la valuation de la meilleure solution trouvée jusqu'ici (initialement égale à $\top$), ℓ contient la valuation locale de l'instanciation courante $\mathcal{A}$ (qui sert ici aussi de minorant, initialement égal à $\perp$).

```
BB-Min(V, A, ℓ, α)
si V = ∅ alors
|   mémoriser la solution
|   retourner ℓ
sinon
|   Soit x_i ∈ V, une variable non instanciée
|   pour tout ν ∈ d_i et tant que (ℓ ≺ α) faire
|   └ α ← min(α, BB-⊛(V − {x_i}, A ∪ {x_i → ν}, ℓ, α, x_i))
└   retourner α

BB-⊛(V, A, ℓ, α, x)
L ← liste des contraintes c = (v, r) ∈ C / (v ∩ V) = ∅, x ∈ v
tant que (L ≠ ∅) et (ℓ ≺ α) faire
|   Choisir et supprimer c de L
└   si A viole c alors ℓ ← (ℓ ⊛ φ(c))
si ℓ ≺ α alors retourner BB-Min(V, A, ℓ, α)
sinon retourner ℓ
```

Algorithme 1: *L'algorithme* Branch and Bound *pour les CSP valués binaires*

Dans le cas possibiliste, une petite sophistication pourrait être apportée, consistant à ne pas tester les contraintes dont la valuation est plus petite que le minorant v courant (leur violation ne peut pas affecter ce minorant).

5.3 Forward-checking

L'algorithme *Forward-checking* utilise une forme extrêmement limitée de cohérence d'arc comme propriété de cohérence locale : il s'agit de pouvoir étendre l'instanciation A courante sur chacune des variables non instanciées en une affectation ne violant aucune contrainte.

Le minorant induit par cette propriété, dite de *check-forward*, est donc la valuation d'une relaxation optimale la vérifiant. Une telle relaxation doit rejeter :

1. toutes les contraintes violées par A elle-même ;

2. pour chaque variable non instanciée x_i, un des ensembles $C(x_i, \nu)$ de contraintes qui sont violées si x_i est instanciée avec la valeur ν de son domaine.

La loi de combinaison $\circledast$ étant monotone, une relaxation optimale est obtenue en rejetant, pour chaque variable x_i non instanciée, un ensemble $C(x_i, \nu)$ dont les valuations combinées sont minimum pour cette variable.

La valuation de cette relaxation est calculable en temps polynomial puisqu'elle est définie par la combinaison :

1. des valuations des contraintes directement violées par A (qui n'est autre que le minorant utilisé dans l'algorithme *Branch and Bound* précédent) ;

2. des valuations des contraintes de tous les ensembles $C(x_i, \nu)$ sélectionnés. Ce calcul nécessite grossièrement $O(end)$ vérifications de contraintes et applications de $\circledast$ (e est le nombre de contraintes) ; toutes les valuations minimums peuvent être obtenues avec moins de (dn) comparaisons et combinées avec moins de n applications de $\circledast$.

Dans la pratique, le minorant peut être maintenu incrémentalement durant la recherche arborescente. On associe à chaque valeur de chaque variable non instanciée une valuation notée $B(\nu, x_i)$, définie comme la combinaison par $\circledast$ des valuations de toutes les contraintes qui seraient violées si la valeur ν était affectée à la variable x_i étant donné l'instanciation courante. Initialement, tous les B sont égaux à $\perp$. Quand l'instanciation courante A est étendue en $A' = A \cup \{x_j = \mu\}$, les B sont mis-à-jour comme suit :

$$B(\nu, x_i) \leftarrow B(\nu, x_i) \circledast \Big(\circledast_{\substack{c=(v,r)\in C, v=\{x_i,x_j\} \\ A\cup\{x_i=\nu\} \text{ viole } c}} \big[\varphi(c)\big] \Big) \tag{1}$$

Le minorant est alors obtenu en combinant par $\circledast$ les valuations de toutes les contraintes violées par l'instanciation courante et tous les $B(\nu, x_i)$ minimum pour chaque variable non instanciée. Lors de l'extension de l'instanciation A en $A' = A \cup \{x_j = \mu\}$, la combinaison des valuations de toutes les contraintes violées par A' est simplement obtenue en combinant la valuation résultant de la combinaison des valuations des contraintes violées par A (calculée au nœud père) avec $B(\mu, x_j)$.

Comme pour le *Forward-Checking* traditionnel, il faudra sauvegarder les $B(\nu, x_i)$ avant chaque affectation pour pouvoir les restaurer lors du retour-arrière

On remarquera que la satisfaction de la condition suffisante pour effacer une valeur ν de la variable x_i peut provenir d'une augmentation des B, mais aussi d'une baisse de α. On peut choisir, par exemple, de vérifier cette condition suffisante (1) avant chaque test de cohérence et (2) après ce test si il y a eu mise-à-jour du B.

Il faut noter que toutes ces «optimisations» permettent uniquement de diminuer le nombre de vérifications de contraintes effectuées par la suite mais ne modifient pas le nombre de nœuds explorés par l'algorithme. De plus, il est nécessaire de sauvegarder (et de restaurer) les domaines en plus des $B(\nu, x_i)$..

L'algorithme ainsi défini recouvre les différentes extensions de l'algorithme *ForwardChecking* proposées pour les CSP possibilistes, flous ou à pénalités [SCH 92, FRE 92, FAR 94a].

L'algorithme correspondant est schématiquement présenté comme l'algorithme 2. Les paramètres utilisés ont la même signification que dans l'algorithme précédent. Le minorant amélioré est maintenu dans un paramètre supplémentaire M, initialement égal à $\perp$. La fonction Push-Domains (resp. Pop-Domains()) sauvegarde (resp. restaure) les $B(\nu, x_i)$ et les domaines des variables non instanciées. La fonction M-à-jour-B met à jour les $B(\nu, x_i)$ associées aux valeurs des variables non instanciées selon le mécanisme décrit dans l'équation 1 et calcule la nouvelle valeur du minorant.

Dans le cas possibiliste il serait possible de ne pas « propager » les contraintes dont la valuation est plus petite que v, que le minorant amélioré ou que le $B(\nu, x_i)$ considéré.

5.4 Étendre l'algorithme Really Full Look-Ahead

Dans la pratique, l'application de l'algorithme précédent à de larges classes de CSP ayant un opérateur strictement monotone donne des résultats assez décevants. Le comportement observé est caractéristique d'une évaluation de médiocre qualité et il est donc naturel de tenter d'augmenter la qualité du minorant en passant à un niveau de cohérence supérieur : l'arc-cohérence complète utilisée dans l'algorithme *Really Full Look-Ahead*.

Dans ce cas le minorant est la valuation de la relaxation optimale parmi celles où l'instanciation courante n'entraîne pas l'apparition de domaine vide lors de l'établissement de l'arc-cohérence (nous dirons dans ce cas que le CSP est arc-incohérent).

Considérons une structure de valuation S avec un opérateur strictement monotone et tel que pour tout $a, b \in E, a, b \prec \top$, on a $(a \circledast b \prec \top)$ (cette propriété supplémentaire est satisfaite par toutes les structures de valuations présentées jusqu'ici). On considère le problème de décision associé au calcul du minorant :

Soit MAX-AC-BCSP le problème de décision défini par :

Instance : Un CSP binaire valué défini dans une telle structure S et une valuation ℓ ;

Question : existe-t-il une relaxation du CSP valué de valuation inférieure à ℓ et qui ne soit pas arc-incohérente ?

FC-Min($V, \mathrm{A}\ , \ell, \alpha, M$)
si $V = \varnothing$ **alors**
  mémoriser solution
  retourner ℓ
sinon
  Soit $x_i \in V$, une variable non instanciée
  pour *tout* $\nu \in d_i$ *et tant que* $(\alpha \succ M)$ **faire**
    $\alpha \leftarrow \min(\alpha,$ FC-$\circledast(V - \{x_i\}, \mathrm{A}\ \cup \{x_i \rightarrow \nu\}, \ell, \alpha, x_i))$
  retourner α

FC-$\circledast$($V, \mathrm{A}\ , \ell, \alpha, x$)
Push-Domains
$\ell \leftarrow \ell \circledast B(\nu, x_i)$
Soit $M \leftarrow$ M-à-jour-B($V, \mathrm{A}\ , \ell, x$)
si $(M \prec \alpha)$ **alors**
  Soit $temp \leftarrow$ FC-Min($V, \mathrm{A}\ , \ell, \alpha, M$)
  Pop-Domains
  retourner $temp$
sinon Pop-Domains
retourner ℓ

M-à-jour-B($V, \mathrm{A}\ , \ell, x$)
Soit L la liste des contraintes $c = (v, r) \in C\ /\ (v \cap V) \neq \varnothing, x \in v$
tant que $(L \neq \varnothing)$ **faire**
  Choisir et supprimer $c = (v, r)$ de L
  Soit $\{x_j\} = v - \{x\}$ **pour** *tout* $\mu \in d_j$ **faire**
    si $(\mathrm{A}\ \cup \{x_j \rightarrow \mu\}$ *viole c)* **alors**
      $B(\mu, x_j) \leftarrow B(\mu, x_j) \circledast \varphi(c)$

retourner $\left[\ell \circledast [\circledast_{x_l \in V} \left(\min_{\omega \in d_l} B(\omega, x_l)\right)]\right]$

Algorithme 2: *L'algorithme* Forward checking *adapté aux CSP valués binaires*

Théorème 1 *Le problème* MAX-AC-BCSP *est* NP*-complet dans le sens fort.*

La preuve est donnée dans l'annexe A. Il est donc aussi difficile (en terme de classe de complexité) de déterminer le nombre minimum de contraintes qu'il faut enlever pour restaurer l'arc-cohérence que de déterminer le nombre minimum de contraintes qu'il faut enlever pour restaurer la cohérence ! Ce résultat rend l'utilisation d'un minorant de ce type peu intéressante et complète le résultat de [BIS 95] sur la difficulté d'étendre les notions de cohérence locale.

On notera que le même problème est polynomial dans le cas d'un opérateur idempotent (le minorant est fourni par établissement de l'arc-cohérence, définie dans ce cas, cf. [ROS 76, SNO 90, SCH 92, FAR 94a]). Un problème simple dans le cas d'un opérateur idempotent devient difficile dans le cas d'un opérateur strictement monotone.

5.5 Autres stratégies d'exploration

Une exploration du type « Meilleur d'abord » ne semble pas adaptée aux CSP du fait de la très grande taille que pourrait atteindre la liste des sommets explorés. Ceci a été confirmé dans une étude expérimentale rapide que nous avions menée sur les CSP possibilistes et par une étude similaire mené par [FRE 92] sur les CSP à pénalités. Des algorithmes de type *Depth First Iterative Deepening* permettraient d'éviter cette complexité spatiale, au prix d'une complexité temporelle légèrement plus forte (mais asymptotiquement équivalente [KOR 85]) et semblent bien adaptés au cas possibiliste ou lexicographique. Dans le cas possibiliste, une autre stratégie, moins complexe en terme de nombre de sommets développés, a été proposé dans [COO 94], mais au prix d'une complexité spatiale exponentielle. Naturellement, si une preuve d'optimalité n'est pas désirée ou impossible pour des raisons de taille, les algorithmes de type « recherche locale » sont probablement plus pertinents.

5.6 Heuristiques

Dans le cadre d'un *Depth First Branch and Bound*, il faut simultanément tenter d'abaisser le plus vite possible le majorant α (*i.e.*, trouver rapidement une très bonne solution), de faire croître le plus rapidement possible le minorant pour pouvoir élaguer et enfin, de limiter l'espace de recherche en choisissant, par exemple, les plus petits domaines en premier.

Dans le cas du *Forward-Checking*, on peut espérer que le choix d'une valeur dont la valuation associée $B(\nu, x_i)$ est minimum permet de se diriger plus rapidement vers une bonne solution.

Pour ce qui est de l'ordre des variables, les heuristiques traditionnelles des CSP classiques (plus petit domaine, plus grand degré, largeur maximum) sont utilisables. [WAL 93] constatent qu'une heuristique s'appuyant sur la largeur et séparant les ex-æco par les domaines donnent de très bon résultats. De nouvelles heuristiques sont expérimentées dans la section 6.

Pour ce qui est de l'ordre des contraintes, il semble judicieux de tirer parti de la dureté et des valuations associées en choisissant de tester en premier les contraintes les plus dures et de valuation maximale afin de faire croître la borne plus rapidement.

6 Quelques expérimentations

Ce chapitre présente les résultats de quelques évaluations préliminaires des algorithmes développés dans le cadre des CSP valués. Seul l'algorithme *Forward-checking* a été considéré et testé sur des problèmes générés aléatoirement. Les quelques tests que nous avons pu effectuer sur le simple *Branch and Bound* montrent une inefficacité importante et notre propos n'est pas tant de comparer les algorithmes que de comparer la difficulté de résolution des problèmes. Le nombre de vérifications de contraintes, le temps de résolution et le nombre d'états explorés sont mesurés. Seule la première mesure est reportée ici car elle est bien corrélée avec le temps d'exécution. Dans tous les cas, le problème considéré est celui de la recherche d'une solution *optimale* avec preuve d'optimalité.

6.1 Évolution de la difficulté

Du fait de leurs relations et de leur intérêt, les cadres classiques, possibilistes et lexicographiques ont été étudiés, en utilisant des CSP aléatoires valués [12]. Dans le cas classique, toute contrainte est considérée comme inviolable.

Nous avons considéré des CSP aléatoires connexes, ayant 16 variables et des domaines de taille 9. Le but étant d'évaluer les difficultés de résolution, une « tranche » à taux de satisfiabilité fixe des contraintes (60 %) a été examinée dans l'espace traditionnel (taux de satisfiabilité, densité du graphe) tel qu'il est décrit dans [HUB 92]. Le nombre de contraintes varie de 15 (CSP arborescent) à 120 (graphe complet) par pas de 7 contraintes. À chaque point, 50 CSP ont été générés et résolus. Les valuations des contraintes sont tirées aléatoirement, avec une distribution uniforme, dans l'ensemble $\{0.25, 0.50, 0.75, 1\}$. Ces valuations sont ignorées dans le cadre classique, traitées comme un multi-ensemble de cardinal 1 dans le cas lexicographique. Les instanciations peuvent avoir deux valuations différentes dans le cas classique, cinq dans le cas possibiliste. Dans le cas lexicographique, si e est le nombre de contraintes, et en supposant que les valuations sont réparties de façon équitable, de l'ordre de $(e/4)^4$ valuations sont possibles. Seul l'algorithme de *Forward Checking* utilisant l'heuristique de choix de valeur (minimum des $B(\nu, x_i)$) et l'heuristique de choix de variable, dynamique, consistant à choisir la variable minimisant le quotient domaine sur degré, a été utilisée. La Figure 6 indique l'évolution des moyennes du nombre de tests de contraintes effectuées dans les trois cas. La probabilité de satisfiabilité dans le cas classique observée sur l'échantillon est indiquée sur la figure en pointillés.

Les conclusions sont multiples : tant que l'on se situe dans une zone où les CSP sont cohérents au sens classique, le surcoût des CSP possibilistes et même lexicogra-

12. Cela permet de constater expérimentalement les différences de complexité résultant d'un opérateur strictement monotone en conservant partiellement la sémantique : les instanciations optimales dans le cas lexicographique sont aussi des instanciations optimales dans le cas possibiliste.

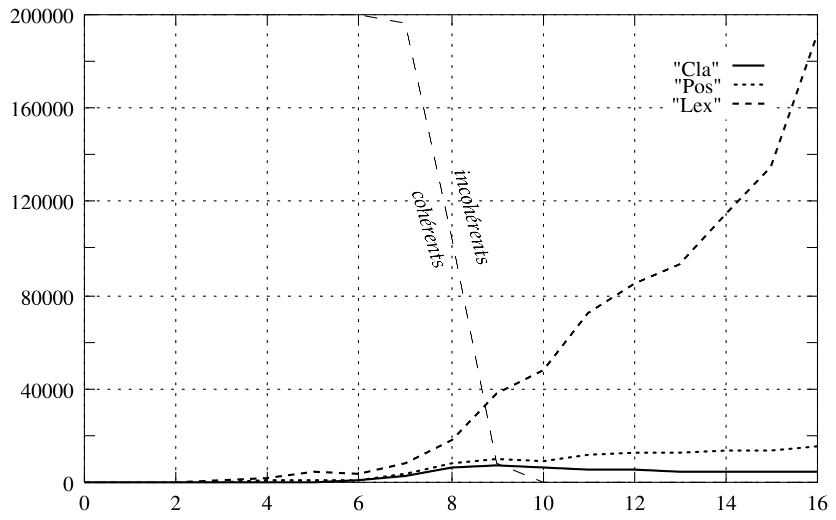

Figure 6. *Moyenne des nombres de vérifications de contraintes dans les cadres classique, possibiliste et lexicographique*

phiques reste « limité », mais c'est aussi le cas où ces extensions sont inutiles. Dans le cas de CSP incohérents, les CSP possibilistes ont un coût aussi important que le pire coût des CSP classiques. Il est probable que ce coût diminue, comme dans le cas classique, quand l'incohérence n'est plus traitée par le cadre possibiliste (toutes les instanciations ayant une valuation de 1), mais ce point n'a pas été atteint dans l'expérimentation.

Le coût des CSP lexicographiques est croissant avec la « dureté » du CSP et le nombre de vérification de contraintes est, dans le pire des cas, de plusieurs ordres de grandeurs supérieur à celui des CSP classiques et possibilistes : la rationalité de traitement semble coûter très cher.

6.2 Heuristiques et évaluations

Nous avons considéré différentes heuristiques de choix de variable, dans le cas de CSP à pénalités, toutes les contraintes ayant une valuation de 1 (cf. [FRE 92]). Seul l'algorithme *Forward-Checking* a été utilisé. Il est plus efficace en général que le simple *Branch and Bound* et une heuristique de choix de valeur (choix de la valeur ayant un $B(\nu, x_i)$ minimum) est disponible.

Considérant les résultats obtenus sur les CSP lexicographiques, proches des CSP à pénalités, nous avons choisi, pour des raisons de temps de résolution, de tester ces heuristiques en un point de l'espace des CSP classiques aléatoires correspondant à des CSP incohérents, mais proches de la frontière avec la cohérence et de mêmes tailles que précédemment. Les contraintes ont un taux de satisfiabilité de 60 %, 73 contraintes sont présentes et 50 CSP aléatoires sont traités avec chacune des heuristiques de choix de variable, une première fois sans l'heuristique de choix de valeur, une seconde fois avec. La moyenne et l'écart-type des résultats obtenus pour chacune d'entre elles sont donnés dans la Figure 7. Les heuristiques de choix de variable considérées sont :

1. **Aléatoire** : l'ordre est choisi aléatoirement afin de juger des effets positifs ou négatifs des autres heuristiques ;

2. **Min. des** $B(\nu, x_i)$: choix d'une variable maximisant le minimum, sur son domaine, des $B(\nu, x_i)$. Le but est de faire augmenter la borne le plus possible dans le pire des cas. L'idée semblait intéressante mais est finalement assez désastreuse ;

3. **Domaine** : choix du plus petit domaine [13]. Les résultats sont de meilleure qualité et dus uniquement à la « propagation » (les domaines sont initialement tous de même taille) ;

4. **Largeur conjonctive** : choix d'une variable de largeur maximum (connectée au plus grand nombre de variables déjà instanciées). En cas d'égalité, la variable de plus petit domaine est choisie (la plus efficace des heuristiques proposées dans [WAL 93]).

5. **Largeur/domaine** : choix d'une variable maximisant le rapport largeur sur domaine. La première variable choisie est une variable de degré maximum, toutes les largeurs étant nulles initialement. Elle est légèrement plus efficace que la précédente sur notre échantillon ;

6. **Largeur** : choix d'une variable de largeur maximum. La première variable choisie est une variable de degré maximum, toutes les largeurs étant nulles initialement. Étrangement, elle est plus efficace que les deux précédentes.

7. **Degré** : choix d'une variable de degré maximum. Le degré, dans des CSP où les contraintes ont toutes la même dureté et les même valuations, semble un excellent guide ;

8. **Largeur+degré** : choix d'une variable maximisant la somme de la largeur et du degré ;

9. **Largeur+degré+domaine** : choix d'une variable maximisant le quotient de la somme de la largeur et du degré par le cardinal du domaine ;

10. **Domaine/degré** : choix d'une variable minimisant le rapport du cardinal du domaine par le degré. Une heuristique méconnue, qui donne aussi d'excellents résultats dans le cas classique.

13. La taille d'un domaine est définie comme le nombre de valeurs ayant un $B(\nu, x_i)$ associé tel que sa combinaison avec le minorant courant ne permet pas de dépasser la valuation de la meilleure solution connue.

La richesse et la quantité des informations permet de considérer un grand nombre d'heuristiques. Il n'est pas certain que ce classement se généralise à d'autres zones de l'espace des CSP aléatoires, mais il apparaît que degré et taille des domaine sont des paramètres essentiels. Dans le cas de problèmes réels, il sera possible (et conseillé) d'exploiter en plus la variabilité des taux de satisfiabilité des contraintes et de leur valuation.

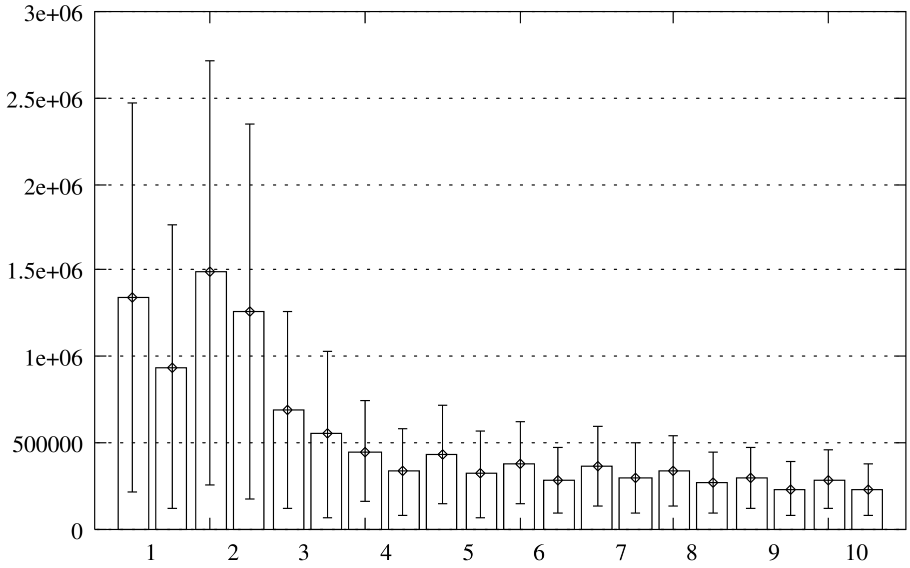

Figure 7. *Moyennes et écart-types des nombres de vérifications de contraintes pour chaque heuristique (sans et avec l'heuristique de choix de valeurs)*

7 Conclusion

Potentiellement, le cadre des CSP valués a un domaine d'application très vaste, permettant de représenter et de traiter des problèmes de satisfaction et des problèmes d'optimisation sous contraintes (affectation de fréquences à des liens de communication radio, affectation de prises de vues à des satellites…). Sa disponibilité dans un langage de programmation par contraintes permettrait à l'utilisateur d'exprimer plus simplement des préférences ou incertitudes. Dans le cas où un problème est déjà modélisé sous forme de CSP classique, la prise en compte de contraintes valuées permet de se diriger, via l'heuristique de choix de valeurs, vers des solutions plus satisfaisantes, avec un coût algorithmique ajustable à volonté.

Plus généralement, la définition d'un cadre algébrique commun permet de distinguer différentes classes de CSP partiels (au sens de Freuder), tant du point de vue de la rationalité des traitements que de la complexité des problèmes. S'il est considéré comme suffisant, le cadre possibiliste permet de conserver des temps de résolution du même ordre que ceux obtenus dans le cas classique et c'est alors un choix idéal, d'autant plus que nombre de notions et d'algorithmes traditionnels, telles que l'établissement de l'arc-cohérence, s'étendent sans difficulté. Dans les autres cas, les temps de résolution observés dépassent très largement les temps de résolution observés dans le cadre classique. La possibilité d'utiliser ces algorithmes comme des algorithmes «*anytime*», fournissant des instanciations de qualité croissante avec le temps permet d'étendre légèrement leur domaine d'application. Un travail important de sophistication des algorithmes et des heuristiques reste cependant à faire.

Il passe par une amélioration de la qualité de l'évaluation des instanciations partielles, mais tous les résultats obtenus ici et dans [BIS 95] montrent que les notions de cohérence locale ne se prêtent pas aisément à la définition de bonnes évaluations. L'utilisation de la classe polynomiale des VCSP arborescents est certainement une direction à explorer.

A Preuve de NP-complétude de MAX-AC-BCSP

Notons tout d'abord que MAX-AC-BCSP appartient à NP car le problème du calcul de la fermeture arc-cohérente d'un CSP binaire peut être résolu en temps polynomial [MAC 85].

Pour montrer la forte NP-complétude de MAX-AC-BCSP, nous exhibons une transformation polynomiale à partir de MAX-2SAT, défini de la façon suivante :

Instance : un ensemble de variables propositionnelles $V = \{v_1, \ldots, v_n\}$, un ensemble Φ de m 2-clauses sur V et un entier $k \leq m$;

Question : existe-t-il une interprétation des variables de V qui satisfasse au moins k clauses de Φ.

Le problème ainsi défini est NP-complet dans le sens fort [GAR 76]. Étant donné une instance de MAX-2SAT, nous construisons une instance de MAX-AC-BCSP comme suit :

L'ensemble X des variables du CSP contient $n + 2n.(m + 1)$ variables :

1. les n premières variables $x_1, \ldots, x_n$ correspondent aux n variables propositionnelles de V et ont un domaine associé $d_i = \{t, f\}$, de cardinalité 2 ;
2. les $2n.(m + 1)$ variables suivantes ont un domaine dont le cardinal est égal à 1, contenant l'unique valeur $\bigstar$. Cet ensemble de variables est construit à partir de n ensembles X_i, $1 \leq i \leq n$, de $2(m + 1)$ variables $X_i = \{x^t_{i,1}, \ldots, x^t_{i,m+1}, x^f_{i,1}, \ldots, x^f_{i,m+1}\}$, $1 \leq i \leq n$. Chaque ensemble X_i de variables est associé avec la variable x_i définie précédemment.

L'ensemble C des contraintes est constitué à partir de 3 ensembles :

1. C_u résulte de la traduction directe des m clauses de Φ. Il y a donc une contrainte entre deux variables x_i et x_j, $1 \leq i, j \leq n$ dès lors que les deux variables propositionnelles v_i et v_j correspondantes apparaissent dans une même clause $\varphi \in \Phi$. Les paires de valeurs autorisées par la contrainte sont celles qui satisfont la clause (*i.e.*, pour une clause $a \vee b$, les paires autorisées sont (t, t), (t, f) et (f, t)) ;
2. C^t contient, pour chaque variable x_i, $1 \leq i \leq n$ et pour chaque variable $x^t_{i,j}$, $1 \leq j \leq m + 1$, une contrainte autorisant uniquement la paire $(t, \bigstar)$;

3. C^f contient, pour chaque variable x_i, $1 \leq i \leq n$ et pour chaque variable $x^f_{i,j}$, $1 \leq j \leq m+1$, une contrainte autorisant uniquement la paire $(f, \bigstar)$;

Soit un total de $2n.(m+1) + m$ contraintes. Chaque contrainte est annotée avec la même valuation α, différente de $\top$ et $\perp$. La Figure 8 donne la micro-structure du CSP résultant de l'application de la transformation à l'instance de MAX-2SAT définie par $k = 3$, $\Phi = \{v_1 \vee v_2, v_2 \vee v_3, v_1 \vee v_3\}$.

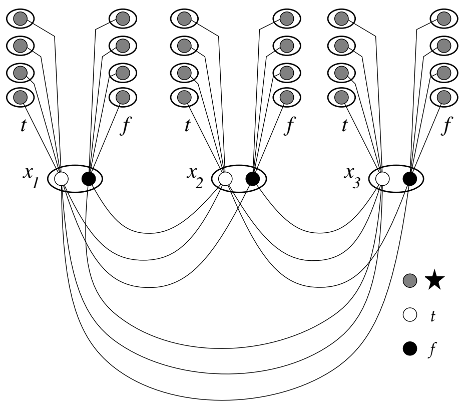

Figure 8. *Exemple d'une transformation*

Le CSP contient $O(m^2)$ contraintes, $O(m^2)$ variables et des domaines de taille $O(1)$ et la complexité de la transformation est donc trivialement polynomiale. Il reste à montrer que l'existence d'une interprétation qui satisfait plus de k clauses de Φ est équivalente à l'existence d'une relaxation ayant une fermeture arc-cohérente non vide et une valuation inférieure à $\ell = (\alpha \circledast \cdots \circledast \alpha)$ avec $n.(m+1) + (m-k)$ occurrences de α.

Notons que l'instance (X, D, C) obtenue n'est pas arc-cohérente : les valeurs t des variables x_i, $1 \leq i \leq n$ n'ont pas de support sur chacune des $m+1$ contraintes de C^f connectant x_i à $x^f_{i,j}$, $1 \leq j \leq m+1$; de même, les valeurs f des variables x_i, $1 \leq i \leq n$ n'ont pas de support sur chacune des $m+1$ contraintes de C^t connectant x_i à $x^t_{i,j}$, $1 \leq j \leq m+1$. De ce fait, un CSP (X, D, C') , $C' \subset C$, ayant une fermeture arc-cohérente non vide ne contiendra jamais simultanément des contraintes de C^t et de C^f pesant toutes les deux sur une même variable x_i, $1 \leq i \leq n$.

Si l'on considère une interprétation ω des variables de V satisfaisant plus de k clauses parmi les m clauses de Φ, la relaxation (X, D, C') correspondante, avec $|C'| = n.(m+1) + k$ est obtenue :

1. en sélectionnant dans C_u les k contraintes qui correspondent aux k clauses satisfaites par ω ;
2. si ω affecte la valeur t à une variable v_i, $1 \leq i \leq n$, on sélectionne aussi les $m+1$ contraintes de C^t pesant sur la variable x_i ;
3. si ω affecte la valeur f à une variable v_i, $1 \leq i \leq n$, on sélectionne aussi les $m+1$ contraintes de C^f pesant sur la variable x_i.

Comme $|C'| = n.(m+1) + k$, il y a précisément $[2n(m+1) + m] - [n.(m+1) + k] = n(m+1) + (m-k)$ contraintes rejetées, toutes annotées avec une valuation égale à α. La relaxation a donc bien une valuation de $\ell = (\alpha \circledast \cdots \circledast \alpha)$ avec $n.(m+1)+(m-k)$ occurrences de α. La fermeture arc-cohérente du CSP ainsi défini existe et est obtenue en supprimant du domaine d_i de x_i, $1 \leq i \leq n$, la valeur t si v_i est affectée à f (car cette valeur n'a pas de support sur les contraintes de C^f, qui ont été sélectionnées), la valeur f sinon (elle n'a pas de support sur les contraintes de C^t, qui ont été sélectionnées). Le CSP ainsi défini est bien arc-cohérent : tous les domaines ont une taille de 1 et ces valeurs satisfont les contraintes sélectionnées dans C_u, C^t et C^f.

Réciproquement, supposons qu'il existe une relaxation (X, D, C'), $C' \subset C$, $|C'| \geq n.(m+1)+k$ ayant une fermeture arc-cohérente. Montrons d'abord que pour tout x_i, $1 \leq i \leq n$, au moins une des contraintes parmi les $2(m+1)$ contraintes de C^t et C^f pesant sur x_i appartient à C'. Supposons que pour une variable x_j, $1 \leq j \leq n$, aucune des contraintes de C^t et C^f et pesant sur x_j n'appartient à C'. Étant donné que, comme il l'a été dit précédemment, un maximum de $m+1$ contraintes pesant sur x_i, $1 \leq i \leq n$, peuvent être sélectionnées dans C^t et C^f sans perdre l'existence d'une fermeture arc-cohérente, un maximum de $(n-1).(m+1)$ contraintes de C' proviennent de C^t et C^f. Comme il est impossible que C' contienne plus de m contraintes provenant de C_u et que $(n-1).(m+1)+m < n.(m+1)+k$ car $k > 0$, on aboutit à une contradiction.

Puisqu'au plus $n.(m+1)$ contraintes de C^t et C^f apparaissent dans C', il y a au moins k contraintes de C_u qui sont dans C' pour atteindre $n.(m+1)+k$. Chaque variable étant impliquée dans au moins une des contraintes de C^t et C^f, les variables de la fermeture arc-cohérente du CSP ont précisément une valeur dans leur domaine et ces valeurs satisfont nécessairement les contraintes de C_u (la fermeture étant arc-cohérente). L'affectation ω des variables de V construite à partir des valeurs des x_i dans cette fermeture satisfait donc plus de k clauses de Φ. $\square$.

Références

[BEN 93] BENFERHAT S., CAYROL C., DUBOIS D., LANG J. et PRADE H., Inconsistency management and prioritized syntax-based entailment. *In : Proc. of the 13th IJCAI*, éd. par BAJCSY R. p. 640–645, Chambéry, France, 1993.

[BIS 95] BISTARELLI S., MONTANARI U. et ROSSI F., Constraint solving over semirings. *In : Proc. of the 14th IJCAI*, Montréal, Canada, août 1995.

[BOW 92] BOWEN J., LAI R. et BAHLER D., Fuzzy semantics and fuzzy constraint networks. *In : Proc. of the first IEEE conference on Fuzzy Systems*, p. 1009–1016, San Francisco, 1992.

[BRE 92] BREWKA G., GUESGEN H. W. et HERTZBERG J., *Constraint Relaxation and Non-monotonic Reasoning*, Rapport technique nTR-92-002, International Computer Science Institute, 1992.

[COO 94] COOPER M. C., On the complexity of fuzzy constraint satisfaction problems, 1994. École d'été sur les problèmes de satisfaction de contraintes.

[DEC 90] DECHTER R., DECHTER A. et PEARL J., Optimization in constraint networks. *In : Influence Diagrams, Belief Nets and Decision Analysis*, éd. par

OLIVER R. et SMITH J., chap. 18, p. 411–425, John Wiley & Sons Ltd., 1990.

[DUB 82] DUBOIS D. et PRADE H., A class of fuzzy measures based on triangular norms. a general framework for the combination of uncertain information. *Int. Journal of Intelligent Systems*, vol. 8, n1, 1982, p. 43–61.

[DUB 93] DUBOIS D., FARGIER H. et PRADE H., Propagation and satisfaction of fuzzy constraints. *In : Fuzzy Sets, Neural Networks and Soft Computing*, éd. par R.R. Y. et L.A. Z., p. 166–187, Kluwer Acad., 1993.

[FAR 93a] FARGIER H. et LANG J., Uncertainty in constraint satisfaction problems: a probabilistic approach. *In : Proc. of ECSQARU '93, LNCS 747*, p. 97–104, Grenade, Spain, novembre 1993.

[FAR 93b] FARGIER H., LANG J. et SCHIEX T., Selecting preferred solutions in Fuzzy Constraint Satisfaction Problems. *In : Proc. of the* 1^{st} *European Congress on Fuzzy and Intelligent Technologies*.

[FAR 94a] FARGIER H., *Problèmes de satisfaction de contraintes flexibles et application à l'ordonnancement de production*, Toulouse, France, Thèse de doctorat, Université Paul Sabatier – Institut de Recherche en Informatique de Toulouse, juin 1994.

[FAR 94b] FARGIER H., LANG J., MARTIN-CLOUAIRE R. et RELLIER J.-P., Uncertainty and flexibility in constraint satisfaction: A case study. *In : Proceedings ECAI'94 Workshop on Constraint Satisfaction Issues raised by Practical Applications*, éd. par SCHIEX T. et BESSIÈRE C., Amsterdam, août 1994.

[FAR 95] FARGIER H., LANG J., MARTIN-CLOUAIRE R. et SCHIEX T., A constraint satisfaction framework for decision under uncertainty. *In : Proc. of the* 11^{th} *Int. Conf. on Uncertainty in Artificial Intelligence*, Montréal, Canada, août 1995.

[FRE 82] FREUDER E. C., A sufficient condition for backtrack-free search. *Journal of the ACM*, vol. 29, n1, 1982, p. 24–32.

[FRE 89] FREUDER E. C., Partial constraint satisfaction. *In : Proc. of the* 11^{th} *IJCAI*, p. 278–283, Detroit, MI, 1989.

[FRE 92] FREUDER E. et WALLACE R., Partial constraint satisfaction. *Artificial Intelligence*, vol. 58, décembre 1992, p. 21–70.

[GAR 76] GAREY M., JOHNSON D. et STOCKMEYER L., Some simplified NP-complete graph problems. *Theoretical Computer Science*, vol. 1, 1976, p. 237–267.

[GAR 79] GAREY M. R. et JOHNSON D. S., *Computers and Intractability : A Guide to the Theory of NP-completeness*, New York, W.H. Freeman and Company, 1979. ISBN: 0-7167-1044-7, 0-7167-1045-5.

[GON 90] GONDRAN M. et MINOUX M., *Graphes et algorithmes*, chap. «Les algèbres de chemin», p. 69–105, Eyrolles, 1990, 2nd. édition. ISSN 0399-4198.

[GUE 94] GUESGEN H. W., *A Formal Framework for Weak Constraint Satsifaction Based on Fuzzy Sets*, Rapport technique nTR-94-026, Berkeley, California, International Computer Science Institute, juin 1994.

[HER 88] HERTZBERG J., GUESGEN H., VOSS A., FIDELAK M. et VOSS H., Relaxing constraint networks to resolve inconsistencies. *In : GWAI'88*, p. 61–65, Eringerfeld, Germany, 1988.

[HUB 92] HUBBE P. D. et FREUDER E. C., An efficient cross-product representation of the constraint satisfaction problem search space. *In : Proc. of AAAI-92*, p. 421–427, San Jose, CA, 1992.

[KOR 85] KORF R. E., Depth first iterative deepening : An optimal admissible tree search. *Artificial Intelligence*, vol. 27, 1985, p. 97–109.

[KRE 88] KRENTEL M., The complexity of optimization problems. *Journal of Computer and System Sciences*, vol. 36, 1988, p. 490–509.

[KRE 92] KRENTEL M. W., Generalizations of optp to the polynomial hierarchy. *Theoretical Computer Science*, vol. 97, 1992, p. 183–198.

[LEH 92] LEHMANN D., *Another Perspective on Default Reasoning*, Rapport de recherche n92-12, Israel, Leibniz Center for Research in Computer Science. Hebrew University of Jerusalem, juillet 1992.

[MAC 85] MACKWORTH A. K. et FREUDER E. C., The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, vol. 25, 1985, p. 65–74.

[MC 92] MARTIN-CLOUAIRE R., Dealing with soft constraints in a constraint satisfaction problem. *In : Proc. of the International Conference on Information Processing of Uncertainty in Knowledge based Systems*, p. 37–40, Palma de Mallorca, juillet 1992.

[MIN 76] MINOUX M., Structures algébriques généralisées des problèmes de cheminement dans les graphes. *Recherche opérationnelle*, vol. 10, n6, 1976, p. 33–62.

[NAD 89] NADEL B. A., Constraint satisfaction algorithms. *Comput. Intell.*, vol. 5, n4, novembre 1989, p. 188–224.

[PAP 94] PAPADIMITRIOU C. M., *Computational Complexity*, Addison-Wesley Publishing Company, 1994. ISBN 0-201-53082-1.

[ROS 76] ROSENFELD A., HUMMEL R. et ZUCKER S., Scene labeling by relaxation operations. *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 6, n 6, 1976, p. 173–184.

[RUT 94] RUTTKAY Z., Fuzzy constraint satisfaction. *In : Proc. FUZZ-IEEE'94*, Orlando, Florida, 1994.

[SCH 92] SCHIEX T., Possibilistic constraint satisfaction problems or "How to handle soft constraints ?". *In : Proc. of the 8^{th} Int. Conf. on Uncertainty in Artificial Intelligence*, Stanford, CA, juillet 1992.

[SCH 95] SCHIEX T., FARGIER H. et VERFAILLIE G., Valued constraint satisfaction problems: hard and easy problems. *In : Proc. of the 14^{th} IJCAI*, Montréal, Canada, août 1995.

[SHA 81] SHAPIRO L. et HARALICK R., Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, 1981, p. 504–519.

[SHA 91] SHAFER G., *An Axiomatic Study of Computation in Hypertrees*, Working paper n232, Lawrence, University of Kansas, School of Business, 1991.

[SNO 90] SNOW P. et FREUDER E., Improved relaxation and search methods for approximate constraint satisfaction with a maximin criterion. *In : Proc. of the 8^{th} biennal conf. of the canadian society for comput. studies of intelligence*, p. 227–230.

[WAL 93] WALLACE R. J. et FREUDER E. C., Conjunctive width heuristics for maximal constraint satisfaction. *In : Proc. of AAAI-93*, Washington, DC, 1993.

[XUO 92] XUONG N. H., *Mathématiques discrètes et informatique*, Paris, France, Masson (Logique Mathématique Informatique), 1992. ISBN : 2-225-82621-8.

Thomas Schiex *a été ingénieur de recherches au sein du Département d'Automatique du CERT-ONERA de 1991 à fin 1994. Depuis 1994, il est chargé de recherches à l'INRA. Il s'intéresse particulièrement à la résolution de problèmes de satisfaction et d'optimisation sous contraintes dans des domaines discrets, issus plus particulièrement de la biologie.*

Hélène Fargier *est docteur en Intelligence Artificielle. Sa thèse, effectuée à l'Institut de Recherche en Informatique de Toulouse et intitulée* « Problèmes de satisfaction de contraintes flexibles et application à l'ordonnancement de production » *est essentiellement consacrée au cadre des CSP flous et montre en particulier comment la majorité des propriétés et algorithmes classiques s'étendent à ce cadre.*

Gérard Verfaillie *est depuis 1986 ingénieur de recherches au sein du Département d'Informatique du CERT-ONERA. Il s'y intéresse au raisonnement à base de contraintes pour résoudre ou aider à résoudre des problèmes de planification, de conception ou de diagnostic, avec application au domaine spatial (conception et gestion de satellites d'observation…).*