Quantification restreinte : vers une utilisation pratique des QCSP

Marco Benedetti, Arnaud Lallouet, Jeremie Vautard

Quantification restreinte : vers une utilisation pratique des QCSP

Marco Benedetti **Arnaud Lallouet** **Jérémie Vautard**

Laboratoire d'Informatique Fondamentale d'Orléans

{marco.benedetti,arnaud.lallouet,jeremie.vautard}@univ-orleans.fr

Résumé

Dans ce papier, nous introduisons le langage QCSP$^+$ qui étend le cadre des QCSP et permet d'exprimer le concept de *quantification restreinte* via une chaîne de CSP interprétés alternativement de manière conjonctive et disjonctive.

La quantification restreinte apparaît comme une solution confortable aux problèmes de modélisation des QCSP et aide, étonnamment, à réutiliser les techniques de propagation et à élaguer l'espace de recherche.

Nous présentons notre solveur de QCSP$^+$, qui est capable de traiter les contraintes arithmétiques et des contraintes globales, et possède des performances comparables avec l'état de l'art.

Abstract

The QCSP$^+$ language we introduce extends the framework of Quantified Constraint Satisfaction Problems (QCSPs) by enabling us to neatly express *restricted quantifications* via a chain of nested CSPs to be interpreted as alternately conjuncted and disjuncted. Restricted quantifiers turn out to be a convenient solution to the crippling modeling issues we encounter in QCSP and—surprisingly—they help to reuse propagation technology and to prune the search space. Our QCSP$^+$ solver—which also handles arithmetic and global constraints—exhibits state-of-the-art performances.

1 Introduction

Nous étendons le cadre des QCSP grâce à un nouveau langage, appelé QCSP$^+$. La conception de ce langage a été motivé par les difficultés rencontrées lors de la modélisation et de la résolution de problèmes non triviaux par des QCSP. Commençons par décrire les (Q)CSP et en indiquer les points faibles en ce qui concerne la modélisation.

Un CSP est un problème représenté par un ensemble de variables ayant chacune un domaine fini, plus une conjonction de contraintes mentionnant ces variables. Par exemple, si on a $x \in \{1,2,3\}$, $y \in \{3,4\}$, et $z \in \{4,5,6\}$, le CSP

$$x < y \ \wedge \ x + y = z \ \wedge \ z \neq 3x$$

peut être résolu en sélectionnant (si possible) une valeur du domaine de chaque variable telles que les trois contraintes sont satisfaites. Par exemple, $x=1$, $y=4$, $z=5$ est une solution valide. Les CSP permettent de représenter naturellement des problèmes du monde réel, et sont utilisés dans de multiples applications.

Dans ce papier, nous verrons un CSP comme un problème de *décision* dans lequel toutes les variables sont quantifiées *existentiellement* (i.e. l'existence d'une seule affectation de toutes les variables satisfaisant toutes les contraintes suffit à répondre au problème de manière positive) :

$$\exists x{\in}\{1,2,3\}\exists y{\in}\{3,4\}\exists z{\in}\{4,5,6\}.\, x < y \wedge x+y = z \wedge z \neq 3x$$

Le fait, apparemment sans conséquences, d'expliciter des quantificateurs sur les variables nous conduit sur un tout autre terrain : que se passe-t-il si l'on quantifie universellement quelques unes des variables du problème ? Par exemple, que signifie "être vraie" pour la formule suivante ?

$$\exists x{\in}\{1,2,3\}\forall y{\in}\{3,4\}\exists z{\in}\{4,5,6\}.\, x < y \wedge x+y = z \wedge z \neq 3x \quad (1)$$

La manière la plus intuitive de répondre à cette question est d' aborder ce problème comme un *jeu* entre deux joueurs. Un des joueur est associé au quantificateur existentiel, et est appelé joueur-$\exists$. Son adversaire est associé au quantificateur universel, et est appelé joueur-$\forall$. Le but de joueur-$\exists$ est de satisfaire toutes les contraintes, et donc tout le problème. Celui de

joueur-$\forall$ est au contraire de violer au moins une des contraintes, faisant ainsi perdre son adversaire. Les deux joueurs jouent chacun leur tour, un nombre fini et fixé de coups. Ces coups qu'ils exécutent consistent à affecter des valeurs aux variables. Les variables qui sont affectées à chaque coup sont déterminées par le *préfixe* du problème (dans notre exemple, le préfixe est "$\exists x \in \{1,2,3\}\forall y \in \{3,4\}\exists z \in \{4,5,6\}$").

Dans (1), le joueur-$\exists$ joue en premier, et doit choisir une valeur pour x. Ensuite, le joueur-$\forall$ doit choisir une valeur pour y. Enfin, lejoueur-$\exists$ affecte z et le jeu se termine : on évalue alors la satisfaction de l'ensemble des contraintes. On dit qu'un tel CSP *quantifié* (QCSP) est vrai si le joueur-$\exists$ possède une stratégie gagnante, i.e. si il peut arriver à satisfaire toutes les contraintes quoi que fasse son adversaire, et est faux sinon. Par exemple, (1) est vrai, cependant il devient faux si on remplace son préfixe par $\forall x \exists y \forall z$. Contrairement aux CSP purement existentiels, l'ordre des variables devient ici important : en changeant y et z de place dans le préfixe de (1), le joueur existentiel perds le jeu.

Les QCSP sont strictement plus "puissants" que les CSP : il existe une gamme complète de QCSP pour lesquels il n'existe aucune représentation "compacte" sous forme de CSP. (la résolution de QCSP est un problème PSPACE-complet, alors que celle des CSP est NP-complet). Certains de ces problèmes sont naturellement perçus par l'homme comme des jeux (modélisation d'un jeu de plateau) tandis que dans d'autres, l'approche "jeu" est plus difficile à voir (voir Section 2). Au jour d'aujourd'hui, plusieurs algorithmes permettant la résolution des QCSP ont été implémentés.[16][6]

Bien que la force de ce langage apporte beaucoup d'espoirs, nous n'avons trouvé dans la littérature aucune modélisation de problème réel résolu par un solveur de QCSP. Une première explication à cet état de fait est que la résolution de QCSP n'en est qu'a ses balbutiements et que donc, la plupart des propagateurs pour les contraintes quantifiées ne sont pas encore bien définis. Cette limitation actuelle décourage, voire empêche, la production de modèles réalistes.

Cependant, nous avons récemment réalisé un solveur de QCSP possédant des propagateurs pour la plupart des contraintes existantes [4]. Mais malgré cela, la plupart des problèmes "naturels" restent étonnamment difficiles à modéliser.

Pour localiser le problème, revenons à la comparaison entre les QCSP et un jeu opposant $\exists$ et $\forall$: un tel jeu possède des règles qui limite les coups possibles de chaque joueur en fonction des coups déjà joués. Par exemple, beaucoup de jeux interdisent de jouer dans une case déjà occupée par un des joueurs. Dans un QCSP, cela signifie que les joueurs ne peuvent pas affecter n'importe quelle valeur à leurs variables : l'ensemble des mouvements possibles est dynamiquement restreint pendant toute la durée du jeu en fonction des coups déjà joués, pour satisfaire les règles.

Et il est justement difficile, avec un QCSP, de représenter la satisfaction de ces règles : le préfixe ne peut fournir aucune représentation de tels domaines de variables dynamiques. Dans (1), z a pour domaine $\{4,5,6\}$ quelques soient les valeurs choisies pour x et y. Il nous faut donc inclure les règles du jeu dans les contraintes : il s'agit de représenter le fait qu'un joueur perds immédiatement s'il choisit une valeur prohibée. Ceci est naturellement faisable pour le joueur-$\exists$: il suffit de représenter le respect des règles dans une contrainte additionnelle. Si le joueur-$\exists$ triche, il rends cette contrainte fausse, et donc perd le jeu.

On ne peut cependant pas appliquer de méthode aussi directe pour le joueur-$\forall$: celui-ci perds le jeu dès que toutes les contraintes sont satisfaites, ce qui ne peut être imposé en ajoutant par conjonction une quelconque contrainte additionnelle.

Pour résoudre le respect des règles par le joueur-$\forall$, on pourrait modifier légèrement le formalisme. Une approche basée sur la reformulation a été présentée dans [1] pour le cas des jeux à deux joueurs en QBF mais, pour les raisons décrites en section 4, elle s'avère ne pas être satisfaisante pour notre problème.

Nous présentons donc une solution différente, basée sur une gestion explicite des règles. Comme nous le verrons, le formalisme des QCSP est insuffisant pour formuler cette solution car il n'autorise pas les *disjonctions* entre les ensembles de contraintes. Considérons le problème suivant (dans lequel il n'y a pour l'instant pas de règles) :

$$\forall X_1 \exists Y_1 \forall X_2 \exists Y_2 . C(X_1, X_2, Y_1, Y_2) \quad (2)$$

Les lettres en majuscule dénotent des *ensembles* de variables plutôt qu'une seule variable (les domaines ne sont pas explicités). Supposons que les premiers coups possibles pour le joueur-$\forall$ sont caractérisés par un ensemble de contraintes (un CSP) $L_1^{\forall}(X_1)$, i.e. une affectation des variables de X_1 est un coup autorisé ssi il est une solution de $L_1^{\forall}$ au sens classique des CSP. Ensuite, la réponse du joueur-$\exists$ au second tour est contrainte par un CSP $L_1^{\exists}(X_1, Y_1)$: Une fois les choix du joueur-$\forall$ concernant X_1 connus, une affectation de Y_1 doit être considérée seulement si elle résout $L_1^{\exists}$. De la même façon, on fournit les règles $L_2^{\forall}(X_1, Y_1, X_2)$ et $L_2^{\exists}(X_1, Y_1, X_2, Y_2)$.

Pour capturer ce scénario, nous présentons les *quantificateurs restreints* $\mathbf{q}X[L]$, avec $\mathbf{q} \in \{\exists, \forall\}$: ils quantifient les variables de X, mais seulement sur les affectations de X qui satisfont la *règle* L, qui est basée sur un CSP.

Avec cette notation compacte, notre exemple s'écrit :

$$\forall X_1[L_1^{\forall}(X_1)].\ \exists Y_1[L_1^{\exists}(X_1, Y_1)].\ \forall X_2[L_2^{\forall}(X_1, Y_1, X_2)].$$
$$\exists Y_2[L_2^{\exists}(X_1, Y_1, X_2, Y_2)].\ C(X_1, X_2, Y_1, Y_2)$$

ce qui se lit "*Pour toute affectation de X_1 telle que $L_1^{\forall}$ est satisfaite, il existe une affectation de Y_2 telle que $L_1^{\exists}$ est satisfaite et sur pour toute ...*".

On peut remettre cette formule sous forme prénexe en considérant que "tel que" représente une conjonction quand le joueur-$\exists$ est concerné, et une implication quand il s'agit du joueur-$\forall$. Donc, on demande en fait si :

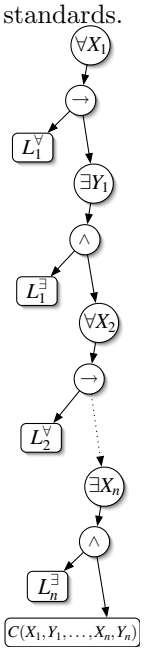
$$\forall X_1(L_1^{\forall} \rightarrow \exists Y_1(L_1^{\exists} \wedge \forall X_2(L_2^{\forall} \rightarrow \exists Y_2(L_2^{\exists} \wedge C)))) \quad (3)$$

ou, en explicitant la disjonction, si

$$\forall X_1 \exists Y_1 \forall X_2 \exists Y_2.(\overline{L_1^{\forall}} \vee (L_1^{\exists} \wedge (\overline{L_2^{\forall}} \vee (L_2^{\exists} \wedge C)))) \quad (4)$$

La formule (4) montre que le moyen le plus naturel pour exprimer la quantification restreinte serait d'étendre le langage de manière à ce qu'il gère les disjonctions. Cependant, passer aux CSP non-conjonctifs en général nous priverait de l'opportunité de réutiliser le grand nombre de propagateurs déjà implémentés dans le cas purement conjonctif. On pourrait retomber dans l'incapacité d'utiliser les outils des CSP sur des problèmes réels, en ayant un joli formalisme et pas d'implémentation efficace sur des problèmes réels.

Notre solution est de concevoir une extension disjonctive *limitée* du formalisme des QCSP qui soit (i) suffisante pour exprimer toutes les disjonctions venant de l' utilisation des quantificateurs (universels) restreints, mais qui soit (ii) toujours capable d'hériter des procédés de raisonnements des solveurs de (Q)CSP standards.



Du point de vue de la logique du premier ordre, la forme syntactique des problème que l' on présente—les QCSP$^+$, généralisant l'exemple (3-4)— est une chaîne alternée de quantifications avec restrictions. Ce langage est intuitivement adéquat pour représenter des problèmes de type jeu-avec-règles (du point de vue de la *modélisation*), (section 2), et il est en même temps possible de le décider en réutilisant la technologie existante (point de vue du *solveur*), (section 3). En effet, cette règle de restriction est un CSP standard, dans lequel les procédures de propagation habituelles peuvent être utilisées. on doit donc concevoir (i) un mécanisme de recherche évaluant la valeur de vérité d'une chaîne alternée de CSP en fonction de la valeur de vérité des feuilles, et (ii) un mécanisme de propagation inter-feuilles qui partage l'information entre les différents CSP de manière correcte.

Notons au final que l'ensemble des scénari de type "jeu-avec-règles" que l'on cible n'est absolument pas un cas marginal. Au contraire, l'utilisation de quantificateurs restreints est tellement naturelle (voir la section 2) qu'on finit par se demander s'il existe des problèmes non-triviaux modélisables avec les QCSP standards ! Le rapport entre notre approche et les contributions récentes sur le terrain de la résolution des QBF est discuté en section 4. Nous présentons l'implémentation d'un solveur complet et des expériences sur des modèles quantifiés en section 5. Dans la section 6, nous résumons nos contributions et nos directions de recherche actuelles.

2 Motivations et exemples

Les concepts impliqués dans la modélisation des problèmes communs suivants sont naturellement capturés par une conjonction de contraintes. Cependant, les questions que nous posons requièrent des quantificateurs universels et des disjonctions du style des QCSP$^+$.

Problème 1 *Supposons un ordre (partiel) $\preceq$ sur les solutions d'un CSP P donné. Un CSP n'étant pas suffisant pour caractériser de manière compacte la* meilleure *solution de P par rapport à $\preceq$, les QCSP$^+$ donnent au problème une formulation élégante en une alternation : $\exists X[P(X)].\ \forall Y[P(Y)].\ Y \preceq X$.*

Dans l'exemple suivant [13], les solutions du CSP représentent des ensembles, et la relation usuelle $\subseteq$ donne la relation de préférence.

Exemple 1 (Strategic companies) *On a une collection C de* compagnies*, un ensemble G de* produits *("goods") et une relation $Prod \subseteq C \times G$ spécifiant les produits fabriqués par chaque compagnie. Les compagnies peuvent participer financièrement aux autres, et un sous-ensemble $C' \subseteq C$ qui détient plus de 50% du capital de $c \in C$ est appelé un ensemble* contrôlant *c. Une compagnie peut avoir plusieurs ensembles la contrôlant. Ils sont tous représentés par la relation $Contr \subseteq 2^C \times C$.*

*Un ensemble de compagnies $S \subseteq C$ est dit "*production-preserving*" —écrit $PP(S)$— si il (i) couvre tous les produits de G, i.e. si en cumulant les produits g tels que $\langle c, g\rangle \in Prod$ et $c \in S$, on obtient G ; et (ii) est clos par relation de contrôle, i.e. pour tout $\langle C', c\rangle \in Contr$, si $C' \subseteq S$, alors $c \in S$.*

On note ensemble stratégique *tout ensemble production-preserving minimal (i.e. tout sous-ensemble strict de cet ensemble n'est pas PP).*

Une compagnie $x \in C$ n'est pas stratégique *(intuitivement : sa vente n'a pas d'impact sur l'ensemble des biens produits ou sur les compagnies contrôlées) si il n'appartient à aucun ensemble stratégique, i.e. si*

$$\forall S[S \subseteq C \wedge \mathsf{PP}(S) \wedge x \in S].\ \exists S'[\mathsf{PP}(S') \wedge x \notin S'].\ S' \subset S$$

Les relations *Prod* et *Contr* définies ci-dessus peuvent aisément être exprimées en logique propositionnelle. On adopte cette restriction pour avoir une comparaison directe avec les solveurs booléens (section 5). Des modèles plus réalistes —explicitant les *quantités* de produits, les *capacités* de production et les *pourcentages* de participations— sont toujours bien exprimés par des CSP, mais sortent du cadre naturel de la logique propositionnelle.

Problème 2 (Stratégie de jeu) *Soit un ensemble de variables X_i décrivant l' état au tour i d'un système évoluant au gré des coups alternés de deux opposants $p_\forall$ et $p_\exists$, sélectionnés parmi un ensemble fini de possibilités. Un* jeu *est défini par un 4-uplet $\langle \mathsf{PA}, \mathsf{SSA}, \mathsf{G}, \mathsf{I} \rangle$ de CSP :*

Un coup M ("Move") dans un état X est possible seulement quand l' axiome de précondition $\mathsf{PA}(X, M)$ *est satisfait et conduit à un nouvel état X' défini par l'* axiome de succession d'état $\mathsf{SSA}(X, X', M)$. *La condition initiale et les conditions de victoire sont reconnues respectivement par $\mathsf{I}(X)$ et $\mathsf{G}(P, X)$*

Considérons que $\overline{\forall}$ dénote $\exists$ et vice-versa. Le premier joueur $p_\exists$ gagne le jeu en au plus k tours si $\exists X_0[I(X_0)].\mathsf{WS}(\exists, 1)$, avec

$$\mathsf{WS}(\mathsf{q}, i) := \mathsf{q}X_i, M_i[\ \mathsf{PA}(X_{i-1}, M_i) \wedge \overline{G}(p_{\overline{q}}, X_{i-1}) \wedge \mathsf{SSA}(X_{i-1}, X_i, M_i)].\mathsf{WS}(\overline{\mathsf{q}}, i{+}1)$$

for $i < 2k$, and $\mathsf{WS}(\mathsf{q}, i) := G(p_{\overline{q}}, X_{i-1})$ for $i = 2k$.

L'exemple suivant montre que les QCSP$^+$ permettent d'échapper à la complexité de la formalisation des jeux que l'on trouve par exemple dans [14].

Exemple 2 (Puissance n×m−k) *Soit $B = (b_{ij}) \in \{0, p_\forall, p_\exists\}^{n \times m}$ une matrice de variables représentant l'état du plateau d'un jeu de Puissance-4 généralisé (joué sur une matrice $n \times m$ où on essaye d'aligner k pions). On modélise l'existence d'une stratégie gagnante pour le premier joueur en posant $\mathsf{PA}(B, x) := b_{nx} \neq 0$, où $x \in [1..m]$ est la colonne du coup courant, et en utilisant*

$$\wedge_{i \in [1..n] j \in [1..m]} (b_{ij} \neq b'_{ij}) \leftrightarrow (j = x \wedge b_{ij} = 0 \wedge b'_{ij} = P \wedge b_{i-1\,j} \neq 0)$$

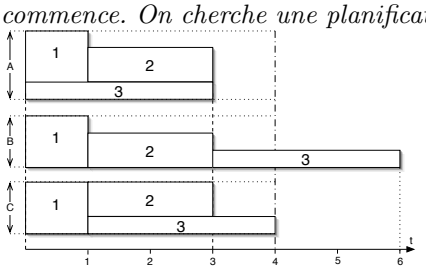
comme $\mathsf{SSA}(B, B', x)$, où $P \in \{p_\forall, p_\exists\}$ est le joueur actuel. La condition initiale $\mathsf{I}(B)$ est $\wedge_{ij} b_{ij} = 0$. Le CSP $\overline{\mathsf{G}}$ est une conjonction de contraintes excluant tout alignement.

Problème 3 (Conformant Scheduling) *Considérons n tâches de durée $\delta_1 \ldots \delta_n$ requérant une quantité de ressource $r_1 \ldots r_n$, et sujette aux contraintes d'ordonnancement $O \subseteq [1..n] \times [1..n]$, où $\langle i, j \rangle \in O$ signifie que la tâche i doit se terminer avant que la tâche j ne démarre.*

On veut ordonnancer les tâches tel que : (i) on finit avant le temps T, (ii) on respecte O, et (iii) la consommation de ressource ne dépasse à aucun instant une capacité fixée R. La plupart des solveurs de CSP fournissent la contrainte globale $\mathsf{cumulative}(R, t_1, \delta_1, r_1, \ldots, t_n, \delta_n, r_n)$ pour vérifier cette dernière condition. On a un degré d'incertitude sur la demande réelle en ressource de chaque tâche : un environnement hostile peut avoir un impact sur elles, selon certaines contraintes $\mathsf{EM}(r_1, \ldots, r_n)$. Est-il possible d' obtenir une planification des tâches robuste par rapport à l'adversaire ?

$$\exists t_1, \ldots, t_n[\wedge_{\langle i,j \rangle \in O} t_i + \delta_i \leq t_j].\ \forall r_1, \ldots, r_n[\mathsf{EM}(r_1, \ldots, r_n)].\ \mathsf{cumulative}(R, t_1, \delta_1, r_1, \ldots, t_n, \delta_n, r_n) \wedge max(t_i + \delta_i) \leq T$$

Exemple 3 *Considérons trois tâches devant être accomplies avant le temps $T = 4$ et sans jamais dépasser la capacité $R = 5$, avec $\delta_1 = 1$, $\delta_2 = 2$, $\delta_3 = 3$, $r_1 = 3$, $r_2 = 2$, and $r_3 = 1$. La tâche 1 doit finir avant que la 2 ne commence. On cherche une planification inattaquable,*



sachant que l'environnement peut ajouter une unité de coût à deux tâches au maximum. L'image ci-contre montre que la planification optimale (A) est sujette à des attaques critiques, la planification linéaire (B) dépasse T, et la solution (C) de notre instance QCSP$^+$ est totalement satisfaisante.

3 Formalisation et décision des QCSP$^+$

Soit V un ensemble de variables et $D = (D_v)_{v \in V}$ l'ensemble de leurs domaines. Soit $W \subseteq V$. On note D^W l'ensemble $\Pi_{v \in W} D_v$ des tuples sur W. La projection d'un tuple t (ou d'un ensemble de tuples T) sur une variable v (ou un ensemble de variables W) est notée $t|_v$ (ou $T|_W$). Une *contrainte* est un couple $c = (W, T)$ avec $W \subseteq V$ et $T \subseteq D^W$ (W et T sont notées $var(c)$ et $sol(c)$). Un *CSP* est un ensemble de contraintes. Pour un CSP C, on note $var(C) = \bigcup_{c \in C} var(c)$ ses variables et $sol(C) = \bowtie_{c \in C} sol(c)$ ses solutions, avec — pour tout$W, U \subseteq V$, $A \subseteq D^W$, et $B \subseteq D^U$ —, on a $A \bowtie B = \{t \in D^{W \cup U} \mid t|_W \in A \wedge t|_U \in B\}$.

On étend la notion de domaine à un ensemble de variables comme suit : on appelle *domaine* sur $W \subseteq V$ toute fonction R de domaine W qui, à chaque $v \in W$, associe un sous-ensemble $R(v) \subseteq D_v$.

Définition 1 (Quantificateur restreint) *Un* quantificateur restreint *est un 4-uple* $\mathcal{Q} = (\mathsf{q}, W, R, C)$ *où* $\mathsf{q} \in \{\exists, \forall\}$, *$W$ est un ensemble de variables, R est un domaine sur W, et C un CSP.*

Définition 2 (QCSP$^+$) *Un QCSP$^+$ doté de variables libres F est un couple $(\mathcal{QS}, G)$, où la* structure de quantification *$\mathcal{QS}$ est une séquence finie (éventuellement vide) $[\mathcal{Q}_1, \ldots, \mathcal{Q}_n]$ de quantificateurs restreints $\mathcal{Q}_i = (\mathsf{q}_i, V_i, R_i, C_i)$ pour lesquels on a toujours :*
- $\forall i, j \in [1..n], V_i \cap F = \emptyset$ *and* $i \neq j \Rightarrow V_i \cap V_j = \emptyset$
- $var(C_i) \subseteq W_i \cup F$ *avec* $W_i = \bigcup_{j \leq i} V_j$

et G est un CSP appelé but, *tel que $var(G) \subseteq W_n \cup F$. Si $F = \emptyset$, on dit que le QCSP$^+$ est* clos.

Comme dans les exemples présentés, on note un tel QCSP$^+$ de manière compacte en l'écrivant :

$$\mathsf{q}_1 V_1 \in R_1\ [C_1].\ \ldots\ .\ \mathsf{q}_n V_n \in R_n\ [C_n].G$$

Il existe seulement deux QCSP$^+$ clos ayant une structure de quantification vide. : $([\,], \top)$ et $([\,], \bot)$, où $\top$ est un CSP vide (trivialement vrai), et où $\bot$ désigne tout CSP possédant une contrainte vide (trivialement faux).

Soit un QCSP$^+$ $P = ([(\mathsf{q}_1, V_1, R_1, C_1), \ldots, (\mathsf{q}_n, V_n, R_n, C_n)], G)$ donné, ayant un ensemble de variables libres F. L'affectation d'une valeur $a \in D_v$ à une variable $v \in F$, s'écrit $P[v=a]$ et est définie par $P[v=a] = ([(\mathsf{q}_1, V_1, R_1, C_1[v=a]), \ldots, (\mathsf{q}_n, V_n, R_n, C_n[v=a])], G[v=a])$. L'affectation dans un CSP est obtenue en reprenant les affectations dans chaque contrainte : soit une contrainte $c = (W, T)$, on a $c[v=a] = c$ si $v \notin W$. Sinon, on a $c' = c[v=a] = (W', T')$ avec $W' = W \setminus \{v\}$ et $T' = \{t' \in D^{W'} \mid \exists t \in D^W, t|_{W'} = t' \wedge t|_v = a\}$.

On étend naturellement cette notion aux affectations de tuples $[W = t]$, avec $W \subseteq V$, $t \in D^W$.

Pour tout $W \subseteq V$, $t \in D^W$, et tout domaine R sur W, on écrit $t \in_W R$ pour $\forall v \in W\ t|_v \in R(v)$.

Définition 3 (Evaluation d'un QCSP$^+$) *Tout QCSP$^+$ clos $(\mathcal{QS}, G)$ est évalué à une valeur de vérité ($\in \{$*true,false*$\}$) comme suit.*

Cas de base $\mathsf{eval}(([\,], \top)) = \mathsf{true}$ *et* $\mathsf{eval}(([\,], \bot)) = \mathsf{false}$.

Récurrence *Soit $\mathcal{QS} = [(\mathsf{q}, W, R, C) \mid \mathcal{QS}']$. On a $\mathsf{eval}((\mathcal{QS}, G)) = \mathsf{true}$ ssi : (i) $\mathsf{q} = \forall$ et pour tout $t \in_W R$ tel que $t \in sol(C)$, on a $\mathsf{eval}((\mathcal{QS}', G)[W = t]) = \mathsf{true}$, ou (ii) $\mathsf{q} = \exists$ et il existe $t \in_W R$ tel que $t \in sol(C)$ et $\mathsf{eval}((\mathcal{QS}', G)[W = t]) = \mathsf{true}$.*

La procédure de décision que nous avons implémentée est calquée sur cette définition de eval, et est basée sur une recherche en profondeur classique et sur l'arbre et/ou associée à la structure de quantification.

Comme dans les solveurs de CSP, il est crucial de posséder une méthode d'inférence en avant (appelée *propagation*) qui soit appliquée à chaque noeud de la recherche, afin d'élaguer l'espace de recherche. Dans le cas des QCSP$^+$, cette propagation est particulière, mais est néanmoins construite à partir de la propagation classique des CSP.

On modélise la propagation dans le cadre des CSP comme une fonction paramétrique $\mathsf{prop}_P(\cdot)$ —où le paramètre P est un CSP — dont le domaine et le codomaine sont les familles de domaines sur $var(P)$. Le résultat $R' = \mathsf{prop}_P(R)$ d'une propagation sur R doit *contracter* le domaine de chaque variable (pour ainsi réduire le facteur de branchement de la recherche), tout en préservant les solutions de P. Cela signifie que pour toute variable $v \in var(P)$, on a $R'(v) \subseteq R(v)$, mais, en même temps, pour toute solution $t \in sol(P)$, si $t|_v \in R(v)$ alors $t|_v \in R'(v)$.

La propagation est calculée, par itération chaotique, comme le plus grand point fixe des opérateurs de réduction de domaine associés à chaque contrainte $c \in P$, où l'opérateur c élague les valeurs qui échouent au *test de consistance locale* de c [2].

Soit R_1 un domaine sur W_1 et R_2 un domaine sur W_2 ($W_1 \cap W_2 = \emptyset$). On définit $R = R_1 \sqcup R_2$ par $R(v) = R_1(v)$ if $v \in W_1$ et $R(v) = R_2(v)$ si $v \in W_2$. On note les projection des domaines sur les sous-domaines avec $|$. Par exemple, on a $R|_{W_1} = R_1$ et $R|_{W_2} = R_2$.

La propagation dans les QCSP$^+$ est définie de la manière suivante :

Définition 4 (Propagation en cascade) *Soit un schéma de propagation* prop *donné, un QCSP$^+$ P, et un domaine R^F pour les variables libres F de P, la* cascade*(P), obtenue par* propagation en cascade *de P est définie comme suit :*

Cas de base. *On a* $\mathsf{cascade}(([\,], G)) = ([\,], \bot)$ *si, pour tout $v \in F$, on a $R'(v) = \emptyset$, avec $R' = \mathsf{prop}_G(R^F)$. Sinon,* $\mathsf{cascade}(([\,], G)) = ([\,], G)$.

Cas inductif. *Soit $P = ([(\mathsf{q}, W, R, C) | \mathcal{QS}], G)$ et $R' = \mathsf{prop}_C(R^F \sqcup R)$. Si il existe $v \in F \cup W$ tel que $R'(v) = \emptyset$, alors* $\mathsf{cascade}(P) = ([\,], \top)$ *si* $\mathsf{q} = \forall$, *et* $\mathsf{cascade}(P) = ([\,], \bot)$ *si* $\mathsf{q} = \exists$. *Sinon (si il n'y a pas de domaine vide dans R'), on a* $\mathsf{cascade}(P) = ([(\mathsf{q}, W, R'|_W, C) | \mathcal{QS}'], G')$ *où* $(\mathcal{QS}', G') = \mathsf{cascade}((\mathcal{QS}, G))$ *est obtenu en utilisant le nouveau domaine R' pour les variables libres $F \cup W$ de $(\mathcal{QS}, G)$.*

Intuitivement, la propagation au niveau de chaque quantificateur restreint (q, W, R, C) est la source d'in-

formation "faisant foi" en ce qui concerne les nouveaux domaines de W. Inversement, pour les variables de $var(C) \setminus W$, on utilise une version locale temporaire des domaines où tous les élagages réalisés par les CSP dominants sont cumulés. Ce résultat parcourt la chaîne des propagations, et disparaît ensuite.

La structure alternée des QCSP$^+$ dote la propagation en cascade d'un avantage intrinsèque sur l' arc-consistance quantifiée usuelle des QCSP : il transforme le test de consistance globale pour les valeurs des domaines universels en une propriété *locale*, décidable au niveau des seules contraintes du CSP représentant la règle du quantificateur restreint.

De cette façon, on peut retirer des valeurs de tous les domaines, qu'ils soient universels ou existentiels, symétriquement, en réutilisant simplement les propagateurs non quantifiés des CSP.

On qualifie de *préservant la validité* toute application $\mathsf{op} : QCSP^+ \rightarrow QCSP^+$ telle que $\forall P \in QCSP^+$, $\mathsf{eval}(P) = \mathsf{eval}(\mathsf{op}(P))$, et on dit que op est un *contracteur* si $\forall P = (\mathcal{QS}, G) \in QCSP^+$, $P' = (\mathcal{QS}', G') = \mathsf{op}(P)$ on a : (i) $|\mathcal{QS}'| \leq |\mathcal{QS}|$, (ii) $var(P') \subseteq var(P)$, et (iii) pour tout $(q, W, R, C) \in \mathcal{QS}$, $(q', W', R', C') \in \mathcal{QS}'$, if $v \in W \cap W'$ alors $R'(v) \subseteq R(v)$.

Théorème 1 *La propagation en cascade est un contracteur préservant la validité pour les QCSP$^+$.*

Cette propriété garantit la correction et la complétude de la procédure de décision, qui réalise la propagation en cascade à chaque noeud de la recherche. Pour plus de détails, voir [3].

Jeux non-blocables.

Dans les QCSP$^+$, il est possible, pendant la résolution, que le CSP d'un quantificateur restreint devienne inconsistant. Du point de vue "jeu" du problème, cela se traduit par le fait que le joueur concerné perds la partie en cours de jeu. Or, il existe de nombreux jeux dans lesquels ce problème ne s'applique pas, si bien qu'il est toujours possible de jouer en suivant les règles, quels que soient les coups joués précédemment.

Plus formellement, de tels problèmes respectent la propriété suivante :

Définition 5 *Un QCSP$^+$ Q est* non-blocable *si, et seulement si :*

1. *il s'agit de* $([], \top)$ *ou de* $([], \bot)$*, ou bien*
2. *Q est de la forme* $([(q, W, R, C) \mid \mathcal{QS}'], G)$*, C admet au moins une solution dans R, et, pour toute solution s de C, le QCSP$^+$ $\mathcal{Q}' = \mathcal{Q}[s \rightarrow W]$ est non-blocable.*

Dans un tel cas, la détection soit de l'inconsistance, soit de la trivialité des conditions de victoire du joueur existentiel (dans notre formalisation, ces conditions de victoire sont représentées par le "but" : le CSP G), permet de décider immédiatement le problème : intuitivement, si le but devient inconsistant, le joueur-$\forall$ n'a qu'à respecter les règles (et il peut toujours le faire), ainsi, soit le joueur-$\exists$ triche, ce qui lui fait perdre la partie, soit il respecte les règles, et la partie se termine avec une défaite pour lui, étant donné qu'on sait que les conditions de victoire étaient inconsistantes. A l'inverse, si le but devient trivial, il suffit au joueur-$\exists$ de respecter les règles pour s'assurer la victoire.

Si on a cette propriété, on peut modifier la propagation en cascade de manière à tenir compte de ce fait : D'une part, l'inconsistance ou la trivialité du goal entraîne le même résultat pour le problème entier. D'autre part, le retrait d'une valeur v du domaine d'une variable libre X d'un CSP C_i du problème implique que cette valeur peut être retirée du CSP associé au quantificateur restreint où X est liée. En effet, le fait que v soit retirée par propagation du domaine d X dans C_i implique, soit que celui-ci deviendra inconsistant si on affecte la valeur v à X, soit que le CSP d'un quantificateur extérieur à Ci l'est déjà, ou le deviendra après cette opération. Etant donné que le problème est non-blocable, par contradiction, v ne peut être affectée à X, ce qui implique son inconsistance dans le CSP où X est liée.

La propagation dans le cas des problèmes non-blocables devient alors :

Définition 6 *Soit un schéma de propagation* prop *donné, un QCSP$^+$ P non-blocable, et un domaine R^F pour les variables libres F de P, la* $\mathsf{cascade}(P)$*, obtenue par* propagation en cascade *de P est définie comme suit :*

Cas unique. *Soit* $P = ([(q_1, W_1, R_1, C_1); \ldots; (q_n, W_n, R_n, C_n)], G)$ *et* $R' = \mathsf{prop}_{C_1 \cup \ldots \cup C_n}(R^F \sqcup R_1 \sqcup \ldots \sqcup R_n)$*. Si* $G = \bot$ *alors* $\mathsf{cascade}(P) = ([], \bot)$*, sinon si* $G = \top$ *alors* $\mathsf{cascade}(P) = ([], \top)$*, sinon (si il n'y a pas de domaine vide dans R'), on a* $\mathsf{cascade}(P) = ([(q_1, W_1, R'|_{W_1}, C_1); \ldots; (q_n, W_n, R'|_{W_n}, C_n)], G')$

Cette propriété est malheureusement aussi difficile à calculer que le problème lui-même. Il est donc nécessaire, dans la pratique, de connaître au préalable cette propriété du problème si on veut pouvoir jouir de l'accélération qu'elle peut produire dans la résolution.

4 Discussions et travaux à ce sujet

L'existence de ce "problème des règles du joueur universel" dans les langages quantifiés conjonctifs a été récemment identifié dans la communauté QBF [1]. Il a été discuté d'une version statique de cette question dans le cas des encodages en QBF de variables de

QCSP multivaluées dont l'application booléenne nécessite d'éviter les combinaisons illégales [15].

Ce problème a un impact considérablement plus grand sur les QCSP que sur les QBF : ce dernier ne prend pas du tout en compte le point de vue de la *modélisation* étant donné qu'on ne demande pas à un humain de lire et de comprendre une spécification en QBF. De telles QBF sont obtenus via une *compilation* assistée par ordinateur, qui part d'un langage de plus haut niveau et finit par un processus appelé *CNF-isation*. Cela consiste à projeter le sens qu'une formule structurée dans une Forme Normale Conjonctive en reliant par conjonction les clauses locales de chaque sous-formules en utilisant des variables auxiliaires [21].

On peut, dans le cas des QBF, contourner le problème des règles en modélisant sous forme d'une matrice conjonctive le sens complet de, par exemple (4). Cette astuce a été adoptée depuis toujours —souvent de manière tacite— dans les modèles QBF. C'est pourquoi il existe une pléthore d'instances de problèmes réels [18].

Une technique d'encodage des QBF plus structurée, appliquée aux jeux à deux joueurs, a été proposée dans [1], où des *variables d'index* sont créées pour relier dans un QBF l'encodage en SAT des préconditions, effets, etc. [19].

Cependant, il est montré que cette approche peut conduire les solveurs à explorer des espaces de recherche artificiellement élargis, un effet qui peut être partiellement contré en utilisant des encodages réglés de manière à obtenir une synergie entre la recherche top-down et les mécanismes d'inférences booléens, ou en fournissant au solveur les informations sur le rôle particulier des variables d'index. Plus récemment, les représentations DNF pour les contraintes booléennes ont été adoptées pour réduire l'espace de recherche [22, 24].

Ces approches utilisent des détails complexes, ou ne présentent pas explicitement un langage ou une sémantique. Elles requièrent des solveurs basés sur la recherche, ou exploitent des concepts purement booléennes (e.g. la DNF), ou encore se focalisent sur des jeux (pour les humains) plutôt que sur le problème plus général de modéliser des concepts via les contraintes quantifiées.

Inversement, la quantification restreinte est une solution plutôt élégante et générale, inspirée par des formes de quantification analogues que l'on trouve dans d'autres langages sous l'appellation *qualifiée* (dans les logiques de description), *bornée* (en Programmation Logique), ou encore *restreinte* (Théorie de la logique et des sémantiques).

Les quantificateurs restreints —qui nous permettent de réutiliser la technologie de propagation et ne s'appliquent pas uniquement à une résolution basée sur la recherche — peuvent être entièrement reportés dans le cas des QBF, où ils produisent le langage QBF$^+$, que l'on peut définir comme étant la *restriction* des QCSP$^+$ aux variables booléennes et aux contraintes clausales. Les QBF$^+$ seront étudiés dans un papier futur, mais nous en présentons une première application dans la section suivante.

Les techniques inspirées des QBF qui ont été mentionnées ci-dessus peuvent être interprétées comme des contournements au problème de la quantification restreinte en QBF. Tandis que les premières expériences suggèrent (cf. section suivante) que les quantificateurs restreints peuvent être tout autant avantageux en QBF, leur introduction dans les QCSP sont supportés par des arguments incontestables, des obstacles majeurs nous empêchant d'adapter les autres solutions :

D'une part, le modeleur devrait avoir la responsabilité de capturer la sémantique de la structure alternante dans (4), ce qui est une menace à l'un des principes fondamentaux des (Q)CSP : les modèles doivent être lisibles et écrivables par l'homme.

D'autre part, les contraintes quantifiées ne peuvent pas subir de manipulations syntaxiques comme les clauses. Pour illustrer ce problème, considérons le QCSP$^+$ suivant : $\exists X[C_1(X)]\forall Y[C_2(X,Y)].C_3(X,Y)$. Une manière d'obtenir une version QCSP de cette formule serait de définir une (unique) contrainte sur $X \cup Y$, qui serait traitée par une extension quantifiée de GAC [20]. Bien que très coûteux (en temps ou en espace, selon la technique employée), ceci est faisable pour des petite contraintes table. Cependant, aucune solution viable n'existe si les C_{1-3} sont, (comme c'est le cas d'habitude) des ensembles de contraintes dont certaines peuvent être arithmétiques, voire globales.

Une autre option serait d'utiliser la version réifiée de chaque contrainte [12]. La version réifiée de $C(X)$ est une contrainte toujours consistante $C'(X,y)$, où $y \in \{0,1\}$, qui est satisfaite par $C'(X,1)$ si $C(X)$ est consistante, et par $C'(X,0)$ sinon. On peut donc écrire $\exists X.\forall Y.\exists a,b.C_1(X) \wedge C'_2(X,Y,a) \wedge C'_3(X,Y,b) \wedge (\neg a \vee b)$ (une version généralisée d'une telle réécriture montre que les QCSP$^+$ sont toujours dans PSPACE).

Dans la pratique, on doit faire face à des complications imparables, étant donné que (i) il n'existe aucune implémentation de la version réifiée de beaucoup de contraintes, et (ii) le passage à la version réifiée peut être catastrophique du point de vue du solveur, étant donné que les contraintes réifiées ne peuvent élaguer de valeurs des domaines des variables tant qu'elles ne connaissent pas la valeur de leur dernier paramètre.

Récemment, Christian Bessière et Guillaume Verger ont introduit dans [9] un formalisme appelé Strategic

CSP permettant de prendre en compte le "problème des règles du joueur universel". Il s'agit d'étendre les QCSP en y ajoutant un ensemble X de variables d'état, non quantifiées, destinées à restreindre en temps voulu les domaines des variables quantifiées Y du problème : lors de la résolution d'un SCSP dont la première variable $y_1$est quantifiée universellement, on ne retient pour cette variable que les valeurs consistantes par rapport au CSP constitué des variables dans $X \cup \{y_1\}$ et dont le réseau de contraintes est la restriction du réseau de contraintes du SCSP initial à ces seules variables.

Si on prend le point de vue du jeu de plateau à deux joueurs, ces variables d'état servent intuitivement à décrire le plateau. Dans les exemples de modélisation donnés dans [9], elles sont utilisées pour représenter la grille 3×3 du jeu du morpion, ou la grille du Puissance-4 (chaque case de la grille de jeu est représentée par une variable d'état dont la valeur détermine le joueur la possédant). Lors de la résolution du SCSP, leurs domaines sont réduits au fur et à mesure que les contraintes les liant deviennent actives, c'est-à-dire, pour chaque contrainte, au moment où la dernière variable quantifiée que cette contrainte concernée est instanciée.

Cependant, ce genre de modélisation ne s'applique pas à des jeux où il est possible de modifier une case du plateau après qu'elle a été jouée. Par exemple, dans le jeu d'othello où il est possible, en les encadrant, de reprendre des pions à l'adversaire, il est nécessaire d'avoir une représentation du plateau après chaque coup, multipliant d'autant les variables nécessaires. Dès lors, la représentation de tels jeux est plus naturelle en utilisant les QCSP$^+$, dans lesquels les variables représentant l'état du jeu sont présentes dans chaque CSP de chaque quantificateur restreint, les règles de transitions de l'état n à l'état $n+1$ étant représentées sous forme de contraintes dans le CSP du $(n+1)$-ième quantificateur restreint (qui a connaissance des variables représentant le n-ième état du jeu). De plus, le fait de n'avoir qu'un seul ensemble de contraintes, l'activation de chacune d'entre elles étant fonction des variables la concernant, rend la modélisation de problèmes réels en SCSP moins aisée qu'en QCSP$^+$.

5 Implantation, modèles et expériences

Nous avons implanté notre procédure de décision pour les QCSP$^+$ dans un système construit à partir du solveur *GeCode* [23]. Notre solveur, appelé QeCode, accepte une grande variété de contraintes en entrée [1], et est disponible publiquement. A présent, aucun autre système de résolution de contraintes quantifiées n'accepte de langage de contraintes aussi expressif que QeCode, et il n'existe pas de jeu de tests disponibles pour ces problèmes. Nous avons proposé une suite de tests initiale en réalisant des générateurs pour les problèmes de la Section 2. Les exemples 1 et 2 ne nécessitent pas de variables non-booléennes, ce qui fait qu'ils ont une traduction directe en QBF ou QBF$^+$. Nos générateurs peuvent produire cette formulation propositionnelle et ceci nous permet de tester notre système contre les solveurs QBF. Il faut toutefois remarquer que cette comparaison n'est que d'importance relative et qu'elle est biaisée en faveur des solveurs QBF car ces problèmes ne permettent pas de mettre en oeuvre les points forts de QeCode (variables non-booléennes, propagateurs complexes), tandis que ses faiblesses sont apparentes (par exemples, ses structures de données ne sont pas spécialisées pour le raisonnement booléen).

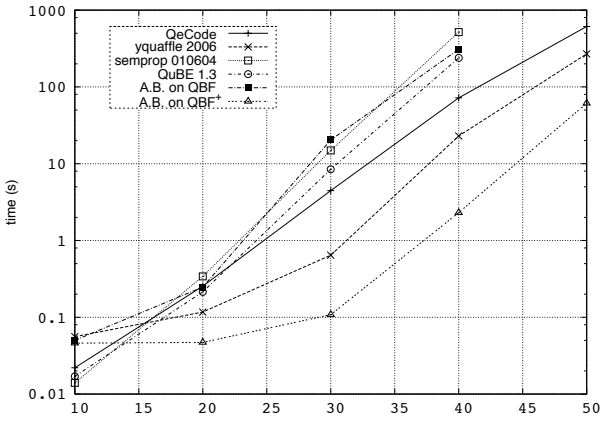


FIG. 1 – Comparaison du temps de résolution sur des problèmes de "Strategic Company" (Exemple 1, Section 2). L'abscisse donne le nombre total de compagnies du problème. QeCode est comparé avec des solveurs QBF de l'état de l'art, et avec la formulation QBF$^+$.

Les résultats sont décrits en Figure 1 et 2. Pour le problèmes des "strategic companies", il apparaît que seulement un solveur QBF parmi les six solveurs "état de l'art" que nous avons testé [2] est plus rapide que QeCode.

Ce résultat est impressionnant si on considère un travail récent[17] qui estime à deux ou trois ordres de magnitude le gain que les solveur de (Q)CSP peuvent atteindre en utilisant les structures de données légères des solveurs SAT/QBF (que GeCode, et donc QeCode, n'utilisent pas).

Pour vérifier cette hypothèse du gain, nous avons modifié la procédure de décision décrite dans [8] de

[1] En fait, toutes celles proposées par (la dernière version de) *GeCode*.

[2] Nous avons aussi testé Quantor [10] et sKizzo [7] mais ils ne sont pas compétitifs sur cette famille de problèmes.

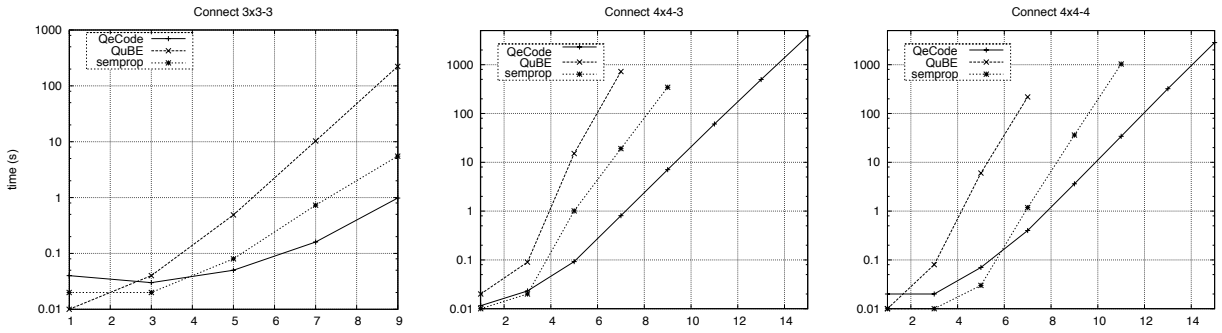


FIG. 2 – Comparaison sur quelques jeu "Puissance $n * m$-k" (Section 2, Exemple 2). L'abscisse donne la profondeur de l'analyse (nombre de coups).

manière à résoudre des problèmes QBF^{+3} : Les résultats, —étiquetés "A.B. on QBF$^+$" dans la figure 1— confirment l'amélioration de un/deux ordres de magnitude sur le meilleur solveur QBF et sur le solveur de base, respectivement.

L'existence de stratégies dans les jeux de plateaux est un problème combinatoire avec un grand nombre d'alternance de quantificateurs et est reconnu difficile pour les algorithmes de raisonnement généralistes. Pour le Puissance-n×m-k, les résultats sont assez favorables pour QeCode (Figure 2, seuls les deux meilleurs solveurs de QBF sont représentés). Nos modélisations n'incluent pas de détection de symétries ni de prédicats auxiliaires.

6 Conclusion et Perspectives

Des obstacles majeurs rendent complexe la modélisation de problèmes issus du monde réel par des langages comme QBF ou QCSP en forme purement conjonctive. La cause principale est la discipline de programmation induite par le quantificateur universel. Nous proposons de surmonter cette difficulté en introduisant le langage QCSP$^+$ qui autorise des quantifications restreintes. Grâce à une intégration contrôlée de la disjonction, ce langage enrichi facilite la modélisation et en même temps est prévu pour faciliter la réutilisation des technologies classiques de résolution et de propagation : le solveur QeCode est implanté au dessus de *GeCode*. Par ailleurs, QeCode (qui permet également de résoudre le cas particulier des QCSP classiques) se trouve être le premier solveur de QCSP qui accepte une très grande variété de contraintes, y compris des contraintes globales, et un des premiers à être diffusé publiquement.

Nous avons réalisé des générateurs multi-langages afin de constituer une première suite d'instances utilisant la quantification restreinte. Les comparaisons avec les solveurs QBF sur certains modèles booléens sont plutôt favorables. Toutefois, notre principale contribution est ailleurs : les QCSP$^+$ offrent la possibilité auparavant absente d'écrire et de résoudre des modèles réels basés sur les contraintes quantifiées.

Nous travaillons actuellement à la modélisation d'applications en QCSP$^+$, et nous recherchons des procédures de décision pour les QCSP$^+$ et QBF$^+$ qui ne sont pas basées sur la recherche.

[3] L'implémentation actuelle est limitée à des préfixes $\forall\exists$, mais le cas général vient également des résultats de la section 3.

Références

[1] C. Ansótegui, C. Gomes, and B. Selman. The Achilles' Heel of QBF. In *Proc. of AAAI*, 2005.

[2] K.R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221(1-2) :179–210, 1999.

[3] M. Benedetti, A. Lallouet, and J. Vautard. Problem Modeling and Solving in QCSP made Practical. Tech. Rep. 11-06, LIFO, Univ. Orleans, 2006.

[4] M. Benedetti, A. Lallouet, and J. Vautard. Reusing CSP propagators for QCSPs. In *Proc. of the Annual ERCIM Workshop on Constraint Solving and Constraint Logic Programming (CSCLP-06)*, Portugal, 2006.

[5] M. Benedetti, A. Lallouet, and J. Vautard. QeCode's web page, `www.univ-orleans.fr/lifo/members/vautard/qecode`. 2006.

[6] G. verger, and C. Bessiere. Blocksolve : une approche bottom-up des QCSP. In *Proc. of JFPC'06*. 2006.

[7] M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proc. of LPAR'04*. 2005.

[8] M. Benedetti. Abstract Branching for Quantified Formulas. In *Proc. of AAAI*, 2006.

[9] C. Bessiere and G. Verger. Strategic constraint satisfaction problems. In *Proc. of CP'06 Workshop on Modelling and Reformulation*, Nantes, France, 2006.

[10] A. Biere. Resolve and Expand. In *Proc. of SAT*, 2004.

[11] L. Bordeaux and E. Monfroy. Beyond NP : Arc-consistency for quantified constraints. In *Proc. of CP*, 2002.

[12] L. Bordeaux. Boolean and Interval Propagation for Quantified Constraints. In *Proc. of Workshop on Quantification in Constraint Programming*, Barcelona, Spain, 2005.

[13] M. Cadoli, T. Eiter, and G. Gottlob. Default logic as a query language. *IEEE Trans. on Knowledge and Data Engineering*, 9(3) :448–463, 1997.

[14] I. P. Gent and A. G. Rowley. Encoding Connect-4 using Quantified Boolean Formulae. *Proc. of 2nd Int. Workshop on Modelling and Reformulating CSPs, Ireland*, 2003.

[15] I. P. Gent, P. Nightingale, and A. Rowley. Encoding quantified CSPs as Quantified Boolean Formulae. In *Proc. of ECAI*, 2004.

[16] I. P. Gent, P. Nightingale, and K. Stergiou. QCSP-Solve : A solver for quantified constraint satisfaction problems. In *Proc. of IJCAI*, 2005.

[17] I. P. Gent, C. Jefferson, and I. Miguel. Minion : Lean, Fast Constraint Solving. *Proc. of ECAI*, 2006.

[18] E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formula library, `www.qbflib.org`, 2001.

[19] H. A. Kautz and B. Selman. Planning as satisfiability. In *Proc. of ECAI*, 1992.

[20] P. Nightingale. Consistency for Quantified Constraint Satisfaction Problems. In *Proc. of CP*, 2005.

[21] D. A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, pages 293–304, 1986.

[22] A. Sabharwal, C. Ansotegui, C. P. Gomes, J. W. Hart, and B. Selman. QBF Modeling : Exploiting Player Symmetry for Simplicity and Efficiency. *Proc. of SAT*, 2006.

[23] C. Schulte and G. Tack. Views and iterators for generic constraint implementations. In *Proc. of CP*, 2005.

[24] L. Zhang. Solving QBF with Combined Conjunctive and Disjunctive Normal Form. In *Proc. of AAAI*, 2006.