Modélisation du Social Golfer en SAT via les contraintes ensemblistes

Frédéric Lardeux[1] **Eric Monfroy**[2]

[1] LERIA - Université d'Angers, Angers, France.
[2] LINA - UMR 6241, Université de Nantes, Nantes, France.
Frederic.Lardeux@univ-angers.fr Eric.Monfroy@univ-nantes.fr

Résumé

Les Problèmes de Satisfaction de Contraintes permettent de modéliser de façon expressive et déclarative les problèmes. De leur côté, les solveurs dédiés au problème de satisfaisabilité (SAT) peuvent gérer de gigantesques instances SAT. Nous présentons donc une technique pour modéliser de façon expressive les problèmes sur les contraintes ensemblistes et pour les encoder automatiquement en SAT. Notre technique est expressive et moins sujette aux erreurs. Nous l'appliquons au probème du Social Golfer en essayant de casser des symmétries du problème.

Abstract

Constraint Satisfaction Problems allow one to expressively model problems. On the other hand, propositional satisfiability problem (SAT) solvers can handle huge SAT instances. We thus present a technique to expressively model set constraint problems and to encode them automatically into SAT instances. Our technique is expressive and less error-prone. We apply it to the Social Golfer Problem and to symmetry breaking of the problem.

1 Introduction

La plupart des problèmes combinatoires peuvent être formulés comme des problèmes de satisfaction de contraintes (CSP) [15]. Un CSP est défini par des variables et des contraintes reliant ces variables. Résoudre un CSP consiste à trouver des affectations des variables qui satisfont les contraintes. Une des plus grandes forces des CSP est l'expressivité : les variables peuvent être de différents types (domaines finis, intervalles, ensembles, ...) et les contraintes également (linéaires ou non, arithmétiques, ensembliste, booléennes, ...). Par ailleurs, le problème de satisfiabilité (SAT) [12] est restreint, en terme d'expressivité, aux variables booléennes et aux formules propositionnelles. Cependant, de nos jours, les solveurs SAT peuvent manipuler des instances SAT gigantesques (plusieurs millions de variables et de clauses). Il est donc intéressant d'encoder les CSP en SAT (e.g., [3, 5]) afin de bénéficier de l'expressivité des CSP et de la puissance de SAT.

Nous nous intéressons ici à l'encodage de contraintes ensemblistes CSP en instances SAT. Dans la communauté de la programmation par contraintes, plusieurs systèmes pour les contraintes ensemblistes ont été réalisés (e.g., bibliothèques [14], intégré à un solveur [1]). Ces travaux ont montré que de nombreux problèmes peuvent être modélisés facilement avec des ensembles.

Coder des problèmes (et des contraintes ensemblistes) directement en SAT est fastidieux (e.g., [16] ou [13]). De plus, optimiser les instances en terme de nombre de variables et clauses conduit rapidement à des modèles très complexes et illisibles dans lesquels se glissent facilement des erreurs. Ainsi, notre approche est basée sur l'encodage automatique des contraintes ensemblistes en instances SAT. Pour ce faire, nous définissons des règles ($\Leftrightarrow_{enc}$) qui encodent les contraintes ensemblistes (telles que l'intersection, l'union, l'appartenance ou le cardinal) dans les clauses et variables SAT correspondantes. L'avantage est que le langage de modélisation (i.e., les contraintes ensemblistes classiques) est déclaratif, expressif, simple et lisible. Nous avons utilisé cette technique pour divers problèmes, et les instances SAT automatiquement générées sont de complexité similaire aux instances générées "à la main". De plus, leur résolution avec un solveur SAT standard (dans notre cas Minisat) est efficace.

Nous illustrons notre approche avec le problème du Social Golfer : q golfeurs jouent chaque semaine durant w semaines, répartis dans g groupes de p golfeurs

$(q = p.g)$; comment programmer le calendrier de ces golfeurs afin qu'aucun d'eux ne joue plus d'une fois dans le même groupe qu'un autre golfeur ?

Des travaux d'encodage tels que [3] et [5] étudient la relation entre la résolution SAT et CSP (en terme de propriétés telles que la consistance) des contraintes sur les domaines finis. Nous sommes nous intéressés par un autre type de contraintes. Pour le problème du Social Golfer (SGP), le travail le plus proche est [16] qui est une révision et une amélioration de [13]. Alors que ces travaux présente l'encodage direct en SAT, nous nous intéressons à un langage de modélisation de plus haut niveau qui est automatiquement transformé en instances SAT. [16] propose également des techniques pour casser les symétries afin d'améliorer le modèle ; quelques une de ces symétries disparaissent naturellement avec les ensembles (e.g., permutations dues au numérotage des joueurs dans un groupe). Les symétries restantes peuvent être facilement cassées dans notre modèle, soit en ajoutant des contraintes ou en raffinant le modèle.

Il est à noter que nous utilisons la contrainte globale de cardinalité de [4] pour encoder la cardinalité des ensembles.

2 Encodage des contraintes ensemblistes

Nous présentons l'encodage de contraintes ensemblistes classiques en CSP (e.g., $\in$, $\cup$, $\cap$, ...) dans des clauses SAT.

Univers et Supports. De façon informelle, l'univers est l'ensemble des éléments qui sont utilisés dans le modèle d'un certain problème, et le support $\mathcal{F}$ d'un ensemble F de ce modèle est l'ensemble des éléments potentiellement dans F (i.e., $\mathcal{F}$ est un sur-ensemble de F).

Definition 1 *Soit P un problème, et M un modèle de P dans $\mathcal{L}$, i.e., une transcription de P du langage naturel au langage de contraintes $\mathcal{L}$.*

- *l'univers $\mathcal{U}$ de M est un ensemble fini de constantes ;*
- *le support de l'ensemble F du modèle M est un sous-ensemble de l'univers $\mathcal{U}$; nous le notons $\mathcal{F}$. $\mathcal{F}$ représente les éléments de $\mathcal{U}$ qui sont possiblement éléments de F :*

$$F \subseteq \mathcal{F} \subseteq \mathcal{U} \quad et \quad F \in \mathcal{P}(\mathcal{F})$$

où $\mathcal{P}(\mathcal{F}) = \{A | A \subseteq \mathcal{F}\}$ est l'ensemble des parties de $\mathcal{F}$.

Tout élément de $\mathcal{U} \setminus \mathcal{F}$ ne peut faire partie de F. Les majuscules (e.g., F) désignent des ensembles, et les majuscules calligraphiques (e.g., $\mathcal{F}$) leur support. Lorsqu'il n'y a pas de confusion, nous abrégeons "l'ensemble F du modèle M" par "l'ensemble F". Considérons un modèle M avec l'univers $\mathcal{U}$, et un ensemble F sur $\mathcal{F}$. Pour chaque x de $\mathcal{F}$, nous considérons une variable booléenne $x_\mathcal{F}$ qui est vraie si $x \in F$ et fausse autrement. Nous appelons l'ensemble de ces variables, les variables support de F. Par la suite, nous écrivons $x_\mathcal{F}$ pour $x_\mathcal{F} = true$ et $\neg x_\mathcal{F}$ pour $x_\mathcal{F} = false$.

La règle d'encodage $\Leftrightarrow_{enc}$. Dans la suite, nous considérons 3 ensembles F, G, et H sur leurs supports respectifs $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$ et l'univers $\mathcal{U}$, et pour chaque $x \in \mathcal{U}$ les diverses variables booléennes $x_\mathcal{F}$, $x_\mathcal{G}$, et $x_\mathcal{H}$ comme défini au-dessus. $|G|$ dénote le cardinal de l'ensemble G.

Nous ne forçons pas les supports à être "minimaux" : par exemple, pour la contrainte $F = G$, les ensembles $\mathcal{F} \setminus \mathcal{G}$ et $\mathcal{G} \setminus \mathcal{F}$ peuvent être non vides, alors que $F \setminus G$ et $G \setminus F$ doivent être vides. Ces cas sont considérés par l'encodage. Permettre aux supports de ne pas être minimaux facilite les processus de modélisation. Par contre, l'utilisation de supports réduits limite la taille des instances SAT générées.

Les clauses qui sont générées par la règle d'encodage $\Leftrightarrow_{enc}$ sont de la forme $\forall x \in \mathcal{F},\ \phi(x_\mathcal{F})$ qui réprésente les $|\mathcal{F}|$ formules $\phi(x_\mathcal{F})$ construites pour chaque élément x du support $\mathcal{F}$ de F (x réfère à l'élément du support/univers, et $x_\mathcal{F}$ à la variable representant x pour l'ensemble F). Pour la contrainte d'appartenance, la règle n'est pas quantifiée ; pour les multi-intersection et multi-union, un quantificateur universel additionnel sur i est utilisé pour dénoter un ensemble de règles d'encodage, chacune reliée à un des ensembles $\mathcal{F}_i$.

Le tableau 1 présente nos règles pour les contraintes ensemblistes : d'abord la contrainte, puis son encodage en SAT, et finalement le nombre de clauses générées. Des contraintes telles que $x \notin F$ ou $F \neq G$ sont définies de façon similaire à des contraintes définies dans le tableau 1. De plus, les contraintes peuvent être reliées par les connecteurs $\vee$, $\wedge$, et $\rightarrow$. Dans notre implémentation pour générer les instances SAT, le résultat d'une union doit être stocké dans un ensemble. Ainsi, $H = \bigcup_{i=1}^{n} F_i$ est équivalent à $H = F_1 \cup H_1$, $H_1 = F_2 \cup H_2$, ... La contrainte de multi-union réduit significativement le nombre de variables (pour les ensembles intermédiaires H_i).

3 Encodage SAT pour les modèles à contraintes ensemblistes

Parmi les divers modèles SAT pour le SGP, le plus classique est l'encodage direct (DE) [16] (qui est déjà une révision de [13]). [16] propose également une variante de DE utilisant des variables intermédiaires.

TABLE 1 – Encodage des contraintes ensemblistes en SAT

$x \in F$	$\Leftrightarrow_{enc}$	$x \in \mathcal{F},\ x_{\mathcal{F}}$	1 clause unitaire
		$x \notin \mathcal{F},\ false$	1 clause vide
$F = G$	$\Leftrightarrow_{enc}$	$\forall x \in \mathcal{F} \cap \mathcal{G},\ x_{\mathcal{F}} \leftrightarrow x_{\mathcal{G}}$	$2.\lvert\mathcal{F} \cap \mathcal{G}\rvert$ clauses binaires
		$\forall x \in \mathcal{F} \setminus \mathcal{G},\ \neg x_{\mathcal{F}}$	$\lvert\mathcal{F} \setminus \mathcal{G}\rvert$ clauses unitaires
		$\forall x \in \mathcal{G} \setminus \mathcal{F},\ \neg x_{\mathcal{G}}$	$\lvert\mathcal{G} \setminus \mathcal{F}\rvert$ clauses unitaires
$F \cap G = H$	$\Leftrightarrow_{enc}$	$\forall x \in \mathcal{F} \cap \mathcal{G} \cap \mathcal{H},\ x_{\mathcal{F}} \wedge x_{\mathcal{G}} \leftrightarrow x_{\mathcal{H}}$	$\lvert\mathcal{F} \cap \mathcal{G} \cap \mathcal{H}\rvert$ clauses ternaires $+ 2.\lvert\mathcal{F} \cap \mathcal{G} \cap \mathcal{H}\rvert$ clauses binaires
		$\forall x \in (\mathcal{F} \cap \mathcal{G}) \setminus \mathcal{H},\ \neg x_{\mathcal{F}} \vee \neg x_{\mathcal{G}}$	$\lvert(\mathcal{F} \cap \mathcal{G}) \setminus \mathcal{H}\rvert$ clauses binaires
		$\forall x \in \mathcal{H} \setminus (\mathcal{F} \cap \mathcal{G}),\ \neg x_{\mathcal{H}}$	$\lvert\mathcal{H} \setminus (\mathcal{F} \cap \mathcal{G})\rvert$ clauses unitaires
$F \cup G = H$	$\Leftrightarrow_{enc}$	$\forall x \in \mathcal{F} \cap \mathcal{G} \cap \mathcal{H},\ x_{\mathcal{F}} \vee x_{\mathcal{G}} \leftrightarrow x_{\mathcal{H}}$	$\lvert\mathcal{F} \cap \mathcal{G} \cap \mathcal{H}\rvert$ clauses ternaires $+ 2.\lvert\mathcal{F} \cap \mathcal{G} \cap \mathcal{H}\rvert$ clauses binaires
		$\forall x \in (\mathcal{F} \cap \mathcal{H}) \setminus \mathcal{G},\ x_{\mathcal{F}} \leftrightarrow x_{\mathcal{H}}$	$2.\lvert(\mathcal{F} \cap \mathcal{H}) \setminus \mathcal{G}\rvert$ clauses binaires
		$\forall x \in (\mathcal{G} \cap \mathcal{H}) \setminus \mathcal{F},\ x_{\mathcal{G}} \leftrightarrow x_{\mathcal{H}}$	$2.\lvert(\mathcal{G} \cap \mathcal{H}) \setminus \mathcal{F}\rvert$ clauses binaires
		$\forall x \in \mathcal{H} \setminus (\mathcal{F} \cup \mathcal{G}),\ \neg x_{\mathcal{H}}$	$\lvert\mathcal{H} \setminus (\mathcal{F} \cup \mathcal{G})\rvert$ clauses unitaires
		$\forall x \in \mathcal{F} \setminus \mathcal{H},\ \neg x_{\mathcal{F}}$	$\lvert\mathcal{F} \setminus \mathcal{H}\rvert$ clauses unitaires
		$\forall x \in \mathcal{G} \setminus \mathcal{H},\ \neg x_{\mathcal{G}}$	$\lvert\mathcal{G} \setminus \mathcal{H}\rvert$ clauses unitaires
$F \subseteq G$	$\Leftrightarrow_{enc}$	$\forall x \in \mathcal{F} \cap \mathcal{G},\ x_{\mathcal{F}} \rightarrow x_{\mathcal{G}}$	$\lvert\mathcal{F} \cap \mathcal{G}\rvert$ clauses binaires
		$\forall x \in \mathcal{F} \setminus \mathcal{G},\ \neg x_{\mathcal{F}}$	$\lvert\mathcal{F} \setminus \mathcal{G}\rvert$ clauses unitaires
$H = F \setminus G$	$\Leftrightarrow_{enc}$	$\forall x \in \mathcal{F} \cap \mathcal{G} \cap \mathcal{H},\ x_{\mathcal{F}} \wedge \neg x_{\mathcal{G}} \leftrightarrow x_{\mathcal{H}}$	$\lvert\mathcal{F} \cap \mathcal{G} \cap \mathcal{H}\rvert$ clauses ternaires $+ 2.\lvert\mathcal{F} \cap \mathcal{G} \cap \mathcal{H}\rvert$ clauses binaires
		$\forall x \in \mathcal{F} \setminus (\mathcal{G} \cup \mathcal{H}),\ \neg x_{\mathcal{F}}$	$\lvert\mathcal{F} \setminus (\mathcal{G} \cup \mathcal{H})\rvert$ clauses ternaires
		$\forall x \in \mathcal{H} \setminus \mathcal{F},\ \neg x_{\mathcal{H}}$	$\lvert\mathcal{H} \setminus \mathcal{F}\rvert$ clauses unitaires
		$\forall x \in (\mathcal{F} \cap \mathcal{H}) \setminus \mathcal{G},\ x_{\mathcal{F}} \leftrightarrow x_{\mathcal{H}}$	$2.\lvert(\mathcal{F} \cap \mathcal{H}) \setminus \mathcal{G}\rvert$ clauses binaires
		$\forall x \in (\mathcal{F} \cap \mathcal{G}) \setminus \mathcal{H},\ x_{\mathcal{F}} \rightarrow x_{\mathcal{G}}$	$\lvert(\mathcal{F} \cap \mathcal{G}) \setminus \mathcal{H}\rvert$ clauses binaires
$H = \bigcup_{i=1}^{n} F_i$	$\Leftrightarrow_{enc}$	$\forall I, J \in \mathcal{P}(N), I \neq \emptyset, I \cup J = N,$ $\forall x \in \mathcal{H} \cap (\bigcap_{i \in I} \mathcal{F}_i) \setminus (\bigcup_{j \in J} \mathcal{F}_j),$ $\bigvee_{i \in I} x_{\mathcal{F}_i} \leftrightarrow x_{\mathcal{H}}$	$\sum_{\substack{I,J \in \mathcal{P}(N),\\ I \neq \emptyset,\\ I \cup J = N}} \left(\lvert\mathcal{H} \cap (\bigcap_{i \in I} \mathcal{F}_i) \setminus (\bigcup_{j \in J} \mathcal{F}_j)\rvert.(\lvert I\rvert + 1)\right)$ clauses binaires et $\sum_{\substack{I,J \in \mathcal{P}(N),\\ I \neq \emptyset,\\ I \cup J = N}} \left(\lvert\mathcal{H} \cap (\bigcap_{i \in I} \mathcal{F}_i) \setminus (\bigcup_{j \in J} \mathcal{F}_j)\rvert\right)$ clauses de tailles $(\lvert I\rvert + 1)$
		$\forall x \in \mathcal{H} \setminus (\bigcup_{i=1}^{n} \mathcal{F}_i),\ \neg x_{\mathcal{H}}$	$\lvert\mathcal{H} \setminus (\bigcup_{i=1}^{n} \mathcal{F}_i)\rvert$ clauses unitaires
		$\forall i \in [1..n],\ \forall x \in \mathcal{F}_i \setminus \mathcal{H},\ \neg x_{\mathcal{F}_i}$	$\sum_{i=1}^{n} \lvert(\mathcal{F}_i \setminus \mathcal{H}\rvert$ clauses unitaires
$H = \bigcap_{i=1}^{n} F_i$	$\Leftrightarrow_{enc}$	$\forall x \in \mathcal{H} \cap (\bigcap_{i=1}^{n} \mathcal{F}_i),\ \bigwedge_{i=1}^{n} x_{\mathcal{F}_i} \leftrightarrow x_{\mathcal{H}}$	$2.\lvert\mathcal{H} \cap (\bigcap_{i=1}^{n} \mathcal{F}_i)\rvert$ clauses de taille $(\lvert I\rvert + 1)$
		$\forall x \in \bigcap_{i=1}^{n} \mathcal{F}_i \setminus \mathcal{H},\ \bigvee_{i=1}^{n} (\neg x_{\mathcal{F}_i})$	$\lvert\bigcap_{i=1}^{n} \mathcal{F}_i \setminus \mathcal{H}\rvert$ clauses n-aire
		$\forall x \in \mathcal{H} \setminus (\bigcap_{i=1}^{n} \mathcal{F}_i),\ \neg x_{\mathcal{H}}$	$\lvert\mathcal{H} \setminus (\bigcap_{i=1}^{n} \mathcal{F}_i)\rvert$ clauses unitaires
$\lvert G\rvert = k$	$\Leftrightarrow_{enc}$	[4]	$n + \sum_{i=1}^{n} 2u_i^n(\lfloor \frac{u_i^n}{2} \rfloor + 1)(\lceil \frac{u_i^n}{2} \rceil + 1) - (\frac{u_i^n}{2} + 1)$ *clauses et* $\sum_{i=1}^{n} u_i^n$ *variables with* $u_n^n = 1, u_1^n = n$ *and* $u_i^n = u_{2i-1}^n + 2u_{2i}^n + u_{2i+1}^n$.

Nous l ?appellerons TME. Nous proposons un modèle pour le SGP à base de contraintes ensemblistes, indépendantes des solveurs. Ces contraintes sont ensuite encodées en SAT grâce à nos règles $\Leftrightarrow_{enc}$.

Modèle ensembliste Une instance du problème est donnée par un triplet $g-p-w$: p joueurs par groupe, g groupes par semaine, et w semaines. L'univers est l'ensemble des golfeurs $\mathcal{P} = \{p_1, \ldots, p_q\}$ avec $q = g.p$ le nombre total de golfeurs. Nous avons besoin de $w.g$ variables ensemblistes pour modéliser les groupes $G_{1,1}$, ..., $G_{w,g}$. L'ensemble $G_{i,j}$ représente le groupe j de la semaine i et est sur le support $\mathcal{G}_{i,j} = \mathcal{P}$. Chaque $G_{i,j}$ contiendra p joueurs de $\mathcal{P}$. Les supports sont "minimaux" : ils ne peuvent pas être réduits sans perdre de solutions (symétriques). Voici les contraintes du problème du Social Golfer.

— p joueurs par groupe chaque semaine

$$\forall i \in [1..w], \forall j \in [1..g], \quad |G_{i,j}| = p \tag{1}$$

— Chaque golfeur joue chaque semaine

$$\forall i \in [1..w] \quad \bigcup_{j=1..g} G_{i,j} = \mathcal{P} \tag{2}$$

— Un golfeur ne joue pas dans deux groupes la même semaine

$$\forall i \in [1..w] \quad \bigcap_{j=1..g} G_{i,j} = \emptyset \tag{3}$$

Les contraintes (3) ne sont pas nécessaires car elles sont impliquées par les contraintes (1) et (2).

— Deux joueurs ne peuvent pas jouer deux fois dans le même groupe

$$\begin{aligned} &\forall w_1, w_2 \in [1..w], p_i, p_j \in \mathcal{P}, \\ &g_1, g_2 \in [1..g], w_1 > w_2 \ \wedge \ i > j \ \wedge \\ &p_i \in G_{w_1,g_1} \ \wedge p_j \in G_{w_1,g_1} \\ &\wedge \ p_i \in G_{w_2,g_2} \ \rightarrow \ p_j \notin G_{w_2,g_2} \end{aligned} \tag{4}$$

Une autre formulation peut être donnée en utilisant les contraintes de cardinalité :

$$\begin{aligned} &\forall w_1, w_2 \in [1..w], g_1, g_2 \in [1..g], \\ &w_1 > w_2 \ \wedge \ g_1 \geq g_2 \ \wedge \\ &|G_{w_1,g_1} \cap G_{w_2,g_2}| \leq 1 \end{aligned} \tag{5}$$

SCE : encodage des contraintes ensemblistes A partir du modèle ci-dessus, nos règles $\Leftrightarrow_{enc}$ d'encodage vont automatiquement générer l'instance SAT correspondante (voir section 2). Le nombre de clauses générées est :

— Les contraintes (1) génèrent $w.g.w.(g.p + \sum_{i=1}^{g.p}[2u_i^{g.p}(\lfloor \frac{u_i^{g.p}}{2} \rfloor + 1)(\lceil \frac{u_i^{g.p}}{2} \rceil + 1) - (\frac{u_i^{g.p}}{2} + 1)])$ clauses avec $u_{g.p}^{g.p} = 1, u_1^{g.p} = g.p$ et $u_i^{g.p} = u_{2i-1}^{g.p} + 2u_{2i}^{g.p} + u_{2i+1}^{g.p}$. La complexité de la formule générée par les contraintes (1) est $\mathcal{O}(w^2.g^3.p^2)$.

— Les contraintes (2) génèrent $w.g.p$ clauses.

— Les contraintes (4) génèrent $w.(w-1).g.(g+1).q.(q-1)/2)$ clauses $(\mathcal{O}(w^2.g^4.p^2))$ et les contraintes (5) génèrent $w.((w-1)/2).g.((g+1)/2).3.q.(q + \sum_{i=1}^{q}[2u_i^q(\lfloor \frac{u_i^q}{2} \rfloor + 1)(\lceil \frac{u_i^q}{2} \rceil + 1) - (\frac{u_i^q}{2} + 1)])$ clauses $(\mathcal{O}(w^2.g^5.p^3))$.

Complexité des instances SAT Puisque la complexité des contraintes (4) est $\mathcal{O}(w^2.g^4.p^2)$ alors que la complexité des contraintes (5) est $\mathcal{O}(w^2.g^5.p^3)$, nous nous concentrerons sur la formulation avec implications (contraintes (4)). La complexité des instances SAT générées par l'encodage du modèle par contraintes ensemblistes (SCE) constitué des contraintes (1), (2), et (4) est $\mathcal{O}(w^2.g^4.p^2)$.

Post-traitement par propagation unitaire La propagation unitaire est un procédé simple pour éliminer des clauses unitaires (des clauses avec uniquement des littéraux faux et un littéral libre) en donnant la valeur vrai aux littéraux libres. Cette valuation peut donner de nouvelles clauses unitaires, et le procédé est alors itéré jusqu'à atteindre un point fixe. Les algorithmes de propagation unitaire peuvent significativement réduire : 1) la taille des instances, 2) le nombre de variables et 3) le temps de résolution. Il est à noter que l'encodage que nous avons choisi pour la contrainte de cardinalité génère de nombreuses clauses unitaires qui disparaissent avec la propagation unitaire.

4 Suppression de symétries pour le SGP

Retirer des solutions symétriques facilite le travail d'un solveur complet. Le problème du Social Golfer possède de nombreuses symétries : la position d'un joueur dans un groupe n'est pas pertinente ; les groupes d'une semaine peuvent être renumérotés ; les semaines peuvent être permutées, ... La suppression de symétries consiste à éliminer les symétries par l'ajout de nouvelles contraintes ou la modification du modèle. [13] propose des clauses pour retirer les symétries entre les joueurs, pour ordonner les groupes d'une semaine, et pour ordonner les semaines. Cependant, ces clauses deviennent de plus en plus complexes et des erreurs peuvent facilement s'introduire. En fait, [16] corrige les clauses pour la suppression de symétries de [13] (erreurs d'indices principalement dans les nombreux opérateurs $\bigvee$ et $\bigwedge$). Nous appelons cet encodage TME^{SB}. D'autres symétries peuvent être supprimées

([11] ou [10]). Toutes les symétries peuvent être supprimées [7], mais au coût d'un nombre super exponentielle de contraintes qui ne peut pas être considéré en pratique.

Avec notre langage nous pouvons supprimer les symétries en ajoutant de nouvelles contraintes au modèle initial ou en raffinant ce modèle par modification de supports et contraintes ; Notre modèle étant différent de celui de [16], nous n'avons pas les mêmes symétries. Nous essayons cependant de supprimer les similaires. Le premier groupe de suppression de symétries ($SB1$) consiste à remplir la première semaine : les p premiers joueurs dans le 1er groupe, les p suivants dans le second, et ainsi de suite. Afin de complèter $SB1$, nous considérons le groupe $SB2$: les p premiers joueurs (qui ont déjà joué ensemble dans le 1er groupe de la 1ère semaine grâce à $SB1$) sont répartis dans différents groupes chaque semaine. Quand p est plus grand que g il est clair que le problème n'a pas de solution. Par la suite nous considérons donc que $g \geq p$.

Suppression de symétries pour le modèle ensembliste par ajout de contraintes Pour $SB1$, il suffit d'ajouter les contraintes suivantes au modèle SCE.

$$\forall i \in [1..p.g], p_i \in G_{1,i \ div \ (p+1)} \tag{6}$$

Pour le 2ème groupe $SB2$ de suppression de symétries, les contraintes nécessaires sont également simples :

$$\forall i \in [2..w], \forall j \in [1..p], p_j \in G_{i,j} \tag{7}$$

Ces contraintes augmentent le nombre de clauses des instances SAT générées, mais toutes ces nouvelles clauses sont unitaires et vont donc être utilisées (puis disparaître) par propagation unitaire. L'encodage SAT de ce modèle ensembliste avec suppression de symétries par ajout de contraintes est nommé SCE^{SBC} et consiste en les contraintes (1), (2), (4), (6) et (7).

Suppression de symétries par modification du modèle. Bien que cela soit plus ennuyeux, cela réduit les supports des ensembles et les cardinalités, et par conséquent la taille des instances SAT générées. Pour $SB1$ l'unique modification consiste à modifier le support des groupes de la 1ère semaine et à fixer ses groupes :

$$\begin{aligned} \forall i \in [1..g], \mathcal{G}_{1,i} = \{p_{1+(i-1).g}, \dots p_{p+(i-1).g}\} \\ \text{and} \ \ \forall i \in [1..g], G_{1,i} = \mathcal{G}_{1,i} \end{aligned} \tag{8}$$

Les autres ensembles, variables, et contraintes restent inchangés. Pour $SB2$, nous changeons les variables de groupes. Plutôt que les $G_{i,j}$, nous considérons les ensembles $G'_{1,1}, \dots, G'_{w,g}$ tels que :

- pour la 1ère semaine $G_{i,j} = G'_{i,j}$;
- pour les semaines suivantes $G_{i,j} = G'_{i,j} \cup \{p_j\}$ si $j \leq p$, $G_{i,j} = G'_{i,j}$ sinon.

Le support des $G'_{1,i}$ (i.e., les groupes de la 1ère semaine) sont définis comme avec $SB1$. Puisque les p premiers joueurs sont répartis sur les p premiers groupes de chaque semaine, les supports des autres groupes peuvent être réduits à $\mathcal{P}' = \{p_{p+1}, \dots, p_q\}$, et $\forall i \in [2..w], \forall j \in [1..g], \mathcal{G}_{i,j} = \mathcal{P}'$

P joueurs par groupe chaque semaine. Les contraintes (1) doivent être remplacées par les contraintes (9)–(11).

$$\forall i \in [1..g], \ \ |G'_{1,i}| = p \tag{9}$$

$$\forall j \in [2..w], \forall i \in [1..p], \ \ |G'_{j,i}| = p - 1 \tag{10}$$

$$\forall j \in [2..w], \forall i \in [p+1..g], \ \ |G'_{j,i}| = p \tag{11}$$

Chaque joueur joue chaque semaine. Les contraintes (12) remplacent les contraintes (2).

$$\forall j \in [2..w] \ \bigcup_{i=1..g} G_{j,i} = \mathcal{P}' \tag{12}$$

Deux joueurs ne peuvent pas jouer deux fois dans le même groupe. Les contraintes (4) sont remplacées par les contraintes (13)–(17). Puisque 2 groupes $G_{i,j}$ avec $j \leq p$ et $i > 1$ ont le joueur p_j en commun, les groupes correspondants $G'_{i,j}$ (dont les supports ne contiennent pas les p_l, $l \leq p$) ne peuvent pas avoir d'autres joueurs p_k en commun :

$$\begin{aligned} \forall w_1, w_2 \in [2..w], \ p_i \in \mathcal{P}, \ g_1 \in [1..p], \ w_1 > w_2, \\ p_i \in G'_{w_1,g_1} \ \rightarrow \ p_i \notin G'_{w_2,g_1} \end{aligned} \tag{13}$$

La relation entre les autres paires de groupes ne change pas.
Entre un groupe de la 1ère semaine (sauf le 1er groupe) et les groupes des autres semaines :

$$\begin{aligned} \forall w_1 \in [2..w], \ p_i, p_j \in \mathcal{P}, \ g_1 \in [2..g], g_2 \in [1..g], \\ i > j, p_i \in G'_{1,g_1} \ \wedge \ p_j \in G'_{1,g_1} \ \wedge \\ p_i \in G'_{w_1,g_2} \ \rightarrow \ p_j \notin G'_{w_1,g_2} \end{aligned} \tag{14}$$

Entre deux groupes (sauf de la 1ère semaine) de même numéro avec un indice supérieur à p :

$$\begin{aligned} \forall w_1, w_2 \in [2..w], \ p_i, p_j \in \mathcal{P}, \ g_1 \in [p+1..g], \\ w_1 > w_2, \ i > j, p_i \in G'_{w_1,g_1} \wedge p_j \in G'_{w_1,g_1} \wedge \\ p_i \in G'_{w_2,g_1} \ \rightarrow \ p_j \notin G'_{w_2,g_1} \end{aligned} \tag{15}$$

Entre deux groupes (sauf de la 1ère semaine) de numéros différents :

$$\begin{aligned} \forall w_1, w_2 \in [2..w], \ p_i, p_j \in \mathcal{P}, \ g_1, g_2 \in [1..g], \\ w_1 > w_2, g_1 \neq g_2, \ i > j, p_i \in G'_{w_1,g_1} \ \wedge \\ p_j \in G'_{w_1,g_1} \ \wedge \ p_i \in G'_{w_2,g_2} \ \rightarrow \ p_j \notin G'_{w_2,g_2} \end{aligned} \tag{16}$$

L'encodage SAT du modèle ensembliste avec suppression de symétries par modification de modèle est nommé SCE^{SBM} et est composé des contraintes (8)–(17).

5 Comparaisons de modèles

Le tableau 2 résume les codages (décrits auparavant) que nous comparons dans la prochaine section. NOM_{UP} dénote le codage NOM après propagation unitaire.

Expressivité. Les variables de notre modèle ensembliste sont bien plus simples : 2 indices au lieu de 4, ceci les rendant plus lisibles. En fait, nous n'avons pas à numéroter les positions dans un groupe, et nous n'avons pas à ajouter un index pour représenter le numéro du joueur. La seconde différence est la simplicité et la lisibilité des contraintes. De fait, les contraintes ensemblistes sont plus expressives que de pures clauses SAT. Ensuite, l'encodage en SAT est fait par les règles $\Leftrightarrow_{enc}$. L'avantage est double : 1) les contraintes sont lisibles, expressives, faciles à modifier, et le modèle qui en résulte est donc plus compréhensible ; 2) moins de fautes peuvent s'introduire car le procédé de modélisation est beaucoup plus simple et léger. De plus, le modèle ensembliste est indépendant des solveurs : le même modèle (exceptée l'exacte syntaxe) peut être résolu pas un solveur CSP sur les ensembles ou par un solveur SAT après encodage par les règles $\Leftrightarrow_{enc}$.

En résumé, en terme d'expressivité, lisibilité, sensibilité aux erreurs, et indépendance aux solveurs, notre modèle ensembliste est supérieur à un encodage direct tel que DE ou TME. La suppression de symétries est également plus simple dans notre modèle.

Structure du modèle Afin de comparer les encodages, nous avons généré des instances du SGP avec : l'encodage direct DE, celui de Triska-Musliu [16] (TME), et notre encodage de modèle ensembliste avec propagation unitaire (SCE_{UP}) et sans (SCE). Dans le tableau 3, chaque instance est définie par le triplet (groupes, joueurs par groupe, semaines) et pour chaque encodage, le nombre de clauses et variables (générées). L'encodage le plus petit pour chaque instance est en gras. On ne peut pas comparer les encodages uniquement par leur taille. Cependant, les très grosses instances ne sont pas résolvables, due aux limites de mémoires. Il est donc primordial de générer des instances les plus petites possibles.

L'encodage direct (DE) est clairement inutilisable quand le nombre de joueurs ou de groupes grossit : le nombre de clauses explose immédiatement. Avec l'introduction de variables intermédiaires, le nombre de clauses est moins important pour TME, mais le nombre de variables grossit tout de même. SCE produit plus de variables mais moins de clauses. Comme supposé, SCE_{UP} produit l'encodage le plus intéressant en terme de variables et clauses : en fait, SCE génère de nombreuses clauses unitaires ou binaires (section 3) qui disparaissent après propagation unitaire.

Impact de la suppression de symétries Nous avons appliqué les 2 groupes de suppression de symétries présentés en section 4 ; les résultats sont présentés dans le tableau 3. Pour TME, l'introduction de contraintes supplémentaires augmente le nombre de clauses (environ 10% de clauses supplémentaires), et le nombre de variables ne change pas. Il est à noter que la propagation unitaire est inutile pour les instances TME et TME^{SB}, car ces instances ne possèdent pas de clauses unitaires. Pour SCE, la suppression de symétries par ajout de contraintes accroit de façon négligeable le nombre de contraintes (voir SCE^{SBC}). La suppression de symétrie par modification du modèle (SCE^{SBM}) réduit significativement la taille des instances SAT générées : de 20 à 60% de variables en moins et de 40 à 60% de clauses en moins. Cette réduction significative est due à la réduction des supports et aux contraintes de cardinalité.

Sans propagation unitaire, les instances de SCE^{SBM} sont toujours les plus petites en terme de nombre de clauses. La propagation unitaire n'a aucun impact sur TME. Cependant son impact est significatif sur SCE, SCE^{SBM} et SCE^{SBC}. Pour SCE, la propagation unitaire divise le nombre de variables par un facteur de 6 à 25 : ceci est principalement dû aux variables des contraintes de cardinalité. Le nombre de clauses est réduit d'environ 10%. Pour SCE^{SBC}, la propagation unitaire réduit encore plus le nombre de variables (jusqu'à 30 fois moins de variables). Le nombre de clauses est lui réduit de 30 à 60%. Pour SCE^{SBM}, la propagation unitaire est moins spectaculaire : en fait, le modèle intiale est lui-même réduit. Cependant, le nombre de variables est divisé par un facteur de 5 à 15. Le nombre de clauses est réduit d'environ 10%.

En résumé, la propagation unitaire est plus bénéfique à SCE^{SBC} ; cependant, $\text{SCE}^{\text{SBM}}_{\text{UP}}$ donne toujours les meilleures instances en terme de nombre de variables et clauses.

6 Analyse expérimentale

Nous comparons maintenant l'efficacité des encodages en terme de temps de résolution, et pour cela, nous utilisons le solveur MiniSat [9]. SatELite [8] est un pré-traitement de Minisat qui réduit énormément

TABLE 2 – Liste des encodages

Nom	Description	Contraintes
DE	Encodage direct	[16]
TME	encodage Triska-Musliu	[16]
TME^{SB}	TME avec suppression de symétries	[16]
SCE	encodage SAT du modèle ensembliste	(1), (2), (4)
SCE^{SBC}	SCE avec suppression de symétries par ajout de contraintes	(1), (2), (4),(6), (7)
SCE^{SBM}	SCE avec suppression de symétries par modification de modèle	(8)–(17)
NAME_{UP}	encodage après propagation unitaire	

TABLE 3 – Taille des instances générées avec différents encodages.

	DE		TME		TME_*^{SB}		SCE		SCE^{SBM}		SCE^{SBC}		SCE_{UP}		$\text{SCE}_{\text{UP}}^{\text{SBM}}$		$\text{SCE}_{\text{UP}}^{\text{SBC}}$	
Prob.	var	cl. $\times 10^6$	var	cl. $\times 10^3$	var	cl. $\times 10^3$	var	cl. $\times 10^3$	var	cl. $\times 10^3$	var	cl. $\times 10^3$	var	cl. $\times 10^3$	var	cl. $\times 10^3$	var	cl. $\times 10^3$
5-3-6	1 350	3	1 800	60	1 800	71	8 625	50	5 702	21	8 625	50	1 410	44	**860**	**18**	980	23
5-3-7	1 575	4	2 100	79	2 100	92	11 110	68	7 734	30	11 110	68	1 645	60	**1 032**	**26**	1 176	34
8-4-4	4 096	48	5 120	323	5 120	390	24 224	235	14 192	96	24 224	235	3 840	205	**2 376**	**78**	2 580	92
8-4-5	5 120	81	6 400	483	6 400	567	34 752	373	22 476	173	34 752	373	4 800	335	**3 168**	**149**	3 440	176
8-4-6	6 144	121	7 680	675	7 680	776	47 072	543	32 552	273	47 072	543	5 760	498	**3 960**	**243**	4 300	288
8-4-7	7 168	170	8 960	898	8 960	1 016	61 184	744	44 420	396	61 184	744	6 720	692	**4 752**	**361**	5 160	427
8-4-8	8 192	227	10 240	1 153	10 240	1 288	77 088	978	58 080	542	77 088	978	7 680	918	**5 544**	**500**	6 020	593
8-4-9	9 216	292	11 520	1 441	11 520	1 592	94 784	1 243	73 532	711	94 784	1 243	8 640	1 175	**6 336**	**663**	6 880	786
8-4-10	10 240	365	12 800	1 759	12 800	1 928	114 272	1 540	90 776	902	114 272	1 540	9 600	1 465	**7 128**	**848**	7 740	1 006
9-4-6	7 776	196	9 720	1 048	9 720	1 191	117 324	858	46 344	448	117 324	858	7 344	793	**5 620**	**472**	5 620	472
9-4-7	9 072	274	11 340	1 401	11 340	1 568	157 284	1 180	63 368	652	157 284	1 180	8 568	1 104	**6 008**	**562**	6 744	701
9-4-8	10 368	366	12 960	1 805	12 960	1 996	203 076	1 553	82 984	895	203 076	1 553	9 792	1 465	**7 024**	**783**	7 868	975
9-4-9	11 664	470	14 580	2 261	14 580	2 261	254 700	1 976	105 192	1 176	254 700	1 977	11 016	1 878	**8 040**	**1 040**	8 992	1 294
9-4-10	12 960	588	16 200	2 767	16 200	3 006	312 156	2 451	129 992	1 496	312 156	2 451	12 240	2 342	**9 056**	**1 334**	10 116	1 658

le nombre de clauses (e.g., en utilisant de la détection de subsomptions) et variables (e.g., en éliminant les littéraux purs). Ce pré-traitement a un coût, mais il améliore généralement le temps global de résolution. Il peut aussi être désactivé. Le tableau 4 représente les temps de résolution de Minisat, avec ou sans le pré-traitement SatElite.

Les expérimentations sont réalisées avec un CPU Intel Core i5-2540M cadencé à 2.60GHz CPU et 4 GB RAM. Pour chaque résolution, un temps limite de 300 secondes est accordé ("-" si le temps limite est expiré). De plus longs temps d'execution on été testés, mais aucune différence significative n'a été observée. Les résultats pour l'encodage DE ne sont pas présentés car, comme supposé, aucun résultats n'a été otenu dans un temps raisonnable.

Le tableau 4 montre que l'utilité de SatElite est difficile à prédire : selon les instances, il peut significativement améliorer ou détériorer les résultats. En moyenne, il n'améliore pas les résultats, et les meilleurs temps de résolution sont obtenus sans ce pré-traitement. La suppression de symétries par modification du modèle (SCE^{SBM}) donne les meilleurs (ou très proche du meilleur) résultats, avec ou sans pré-traitement. La propagation unitaire a peu d'influence sur les temps de résolution pour l'encodage SCE^{SBM}. La suppression de symétries par ajout de contraintes (SCE^{SBC}) n'améliore pas, sauf quand la propagation unitaire est appliquée ($\text{SCE}_{\text{UP}}^{\text{SBC}}$). En fait, $\text{SCE}_{\text{UP}}^{\text{SBC}}$ obtient des résultats aussi bons que SCE^{SBM}. Supprimer les symétries dans TME est plutôt un procédé fluctuant : selon les instances et l'usage ou non de SatELite, les résultats sont significativement améliorés ou dégradés.

En résumé, les meilleurs résultats sont obtenus avec notre modèle contraintes ensemblistes, avec $\text{SCE}_{\text{UP}}^{\text{SBC}}$ quand le pré-traitement est appliqué, ou de manière prédominante avec $\text{SCE}_{\text{UP}}^{\text{SBM}}$ quand le pré-traitement n'est pas appliqué. Finalement, les meilleurs résultats

TABLE 4 – Temps de résolution Minisat

Prob.	TME	TME^{SB}	SCE	SCE^{SBM}	SCE^{SBC}	SCE_{UP}	$\text{SCE}^{\text{SBM}}_{\text{UP}}$	$\text{SCE}^{\text{SBC}}_{\text{UP}}$	TME	TME^{SB}	SCE	SCE^{SBM}	SCE^{SBC}	SCE_{UP}	$\text{SCE}^{\text{SBM}}_{\text{UP}}$	$\text{SCE}^{\text{SBC}}_{\text{UP}}$
	avec SatElite								**sans** SatElite							
5-3-6	8.92	0.69	0.18	0.06	0.12	0.12	0.07	**0.04**	9.37	0.30	1.05	**0.01**	**0.01**	0.26	**0.01**	**0.01**
5-3-7	98.28	13.37	1.42	0.13	1.21	5.09	0.09	**0.08**	97.47	24.86	9.19	**0.06**	0.13	5.67	1.79	0.28
8-4-4	1.04	1.33	0.97	0.32	1.19	0.90	0.29	**0.27**	0.05	0.23	0.09	**0.03**	0.07	0.07	**0.03**	**0.03**
8-4-5	2.26	2.64	1.93	0.86	2.51	1.89	0.84	**0.78**	0.08	0.58	0.13	0.06	0.11	0.06	**0.05**	0.07
8-4-6	4.44	5.16	3.65	1.87	4.74	3.65	1.82	**1.71**	0.25	3.58	0.27	0.14	0.18	0.19	**0.08**	0.09
8-4-7	34.25	94.68	8.66	3.59	8.52	7.52	3.64	**3.46**	27.05	25.88	3.53	**0.48**	1.71	1.94	0.56	0.98
8-4-8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8-4-9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8-4-10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9-4-6	8.45	10.52	11.24	3.15	10.34	11.10	**2.71**	4.58	0.23	3.72	0.37	0.13	0.29	0.25	**0.11**	0.13
9-4-7	13.69	27.16	18.95	5.80	17.8	19.04	**5.12**	8.76	0.31	6.61	0.58	0.22	0.51	0.36	**0.14**	0.24
9-4-8	-	-	31.87	**11.10**	29.60	31.48	12.72	14.90	247.83	-	14.66	5.03	1.10	20.93	2.62	**0.68**
9-4-9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9-4-10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

sont obtenus sans pré-traitement.

7 Discussion et Conclusion

Modélisation. Modéliser un problème avec des contraintes ensemblistes et ensuite générer automatiquement l'instance SAT correspondante est beaucoup plus simple que d'écrire directement les instances SAT comme avec DE ou TME. La suppression des symétries est plutôt ardue dans les encodages directs, et très simple par ajout de contraintes dans le modèle ensembliste (un peu plus ardue par rafinement de modèle). Utiliser un formalisme de haut niveau est donc bénéfique à la phase de modélisation : cela simplifie la tâche, et limite les possibilités d'introduction d'erreurs (généralement dues à la manipulation de nombreux indices). L'encodage SAT est ensuite automatisé.

Instances SAT Nous avons montré que les instances SAT générées automatiquement par notre encodage sont de bonne qualité : 1) elles contiennent toujours significativement moins de clauses (avec ou sans suppression de symétrie, et avec ou sans propagation unitaire) ; 2) après propagation unitaire, elles contiennent également moins de variables ; et 3) elles sont résolues plus efficacement par Minisat, sans "tuning de paramètres", avec ou sans pré-traitement par SatElite.

Suppression de symétries Nous avons montré que supprimer des symétries par ajout de contraintes au modèle ensembliste est très simple. De plus, après propagation unitaire, les instances SAT générées sont beaucoup plus petites, et leur temps de résolution est également amélioré. La suppression de symétries par modification du modèle est encore plus bénéfique. L'effort supplémentaire par rapport à l'ajout de contraintes, est très bénéfique au niveau de la taille des instances, mais juste valable en terme de temps de résolution (ceci dépend des instances et du prétraitement). Ainsi, il faut mettre en balance temps de résolution et de modélisation. La taille des instances générées peut aussi être un facteur décisif : de plus gros problèmes peuvent être générés par la suppression de symétries en modifiant le modèle (voir SCE^{SBM}).

Contraintes ensemblistes en programmation par contraintes En terme d'ensembles, l'expressivité de notre proposition est plus ou moins similaire à celle d'un système de programmation par contraintes tel que [1] : c'était notre but. L'avantage de [14] ou [1], est que les contraintes ensemblistes peuvent être mélangées à d'autres contraintes. Nous prévoyons aussi l'encodage d'autres contraintes dans le futur. Dans des sytèmes tels que [14], un solveur spécial doit être réalisé (à base de réduction de domaine et d'énumération). [2] compare des solveurs CSP ensemblistes pour le SGP : la plupart des résultats sont obtenus par des heuristiques de recherche (dynamiques) ou des mécanismes spécifiques de résolution. Notre approche est très différente : nous ne voulons pas réaliser un solveur spécifique, ni adapter ou régler un solveur existant pour résoudre efficacement nos instances SAT ; nous voulons transformer un modèle ensembliste de haut niveau en une instance SAT de "bonne" qualité (en terme de taille et résolution) qui est efficacement résolue par un solveur SAT classique.

En terme de résolution, l'instance 5-3-6 de notre modèle ensembliste n'est pas résolue avant le temps limite par le solveur standard de [1], alors que son encodage est résolu en moins d'une seconde par Minisat.

Afin de réduire encore la taille des instances SAT générées, nous prévoyons d'ajouter un pré-traitement (forcer une consistance locale) pour réduire les supports des ensembles. Pour le SGP, les supports sont "minimaux" (en terme de réduction comme dans [14]). Cependant, pour d'autres problèmes, les supports peuvent être réduits par un processus de déduction (sans perdre de solution), et donc, les instances SAT générées peuvent également être plus petites. Un tel mécanisme pourrait être équivalent à une phase de réduction du système [14].

Conclusion Nous avons présenté une technique pour l'encodage en SAT de contraintes ensemblistes : la modélisation est effectuée en utilisant des contraintes ensemblistes expressives et déclaratives qui sont ensuite automatiquement converties en variables et clauses SAT grâce à nos règles d'encodage $\Leftrightarrow_{enc}$. Cette technique a été appliquée avec succès au problème du Social Golfer, et à la suppression de symétries pour ce problème. Les avantages de notre technique sont les suivants : 1) la modélisation est expressive, déclarative et lisible. De plus, les modèles sont indépendants du solveur et peuvent être résolus par un solveur CSP ou SAT. 2) cette technique limite l'introduction d'erreurs, ce qui est fréquent avec des codages directs en SAT ; 3) des symétries peuvent être supprimées par le simple ajout de contraintes supplémentaires ou par raffinement du modèle ; 4) les instances SAT qui sont automatiquement générées sont plus petites que celles de [16] qui sont directement écrites et ont déjà subi des améliorations ; après propagation unitaire, nos instances contiennent également moins de variables que celles de [16] ; 5) finalement, par rapport au temps de résolution, nos instances du SGP automatiquement générées sont résolues plus rapidement, avec ou sans propagation unitaire, avec ou sans suppression de symétries et avec ou sans SatElite.

Nous avons appliqué notre technique à d'autres problèmes (tels que le car-sequencing, les n-reines, le sudoku, ...). Nous obtenons toujours des modèles très simples et très lisibles. Les instances SAT générées semblent également convenir parfaitement à un solveur tel que Minisat.

Nous avons l'intention d'adapter notre encodage pour d'autres domaines, tels que les domaines finis et les séquences. Nous prévoyons également de raffiner la notion de supports et d'intégrer un pré-traitement afin de les réduire. Cela n'aura pas d'impact sur le modèle du SGP que nous avons proposé, mais pour beaucoup de problèmes, cela permettra d'obtenir des instances SAT encore plus petites.

Références

[1] Minizinc. http ://www.minizinc.org/.

[2] F. Azevedo. An attempt to dynamically break symmetries in the social golfers problem. In *CSCLP*, pages 33–47, 2006.

[3] F. Bacchus. Gac via unit propagation. In *Proc. of CP 2007*, volume 4741 of *LNCS*, pages 133–147. Springer, 2007.

[4] O. Bailleux and Y. Boufkhad. Efficient cnf encoding of boolean cardinality constraints. In *Proc. of CP 2003*, volume 2833, pages 108–122. Springer, 2003.

[5] C. Bessière, E. Hebrard, and T. Walsh. Local consistencies in sat. In *Selected Revised Papers of SAT 2003.*, volume 2919 of *LNCS*, pages 299–314. Springer, 2004.

[6] C. Cotta, I. Dotú, A. J. Fernández, and P. V. Hentenryck. Scheduling social golfers with memetic evolutionary programming. In *HM 2006*, volume 4030 of *LNCS*, pages 150–161. Springer, 2006.

[7] J. M. Crawford, M. L. Ginsberg, E. M. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *Proc. of KR'96*, pages 148–159. Morgan Kaufmann, 1996.

[8] N. Eén and A. Biere. Effective preprocessing in sat through variable and clause elimination. In *SAT 2005*, volume 3569, pages 61–75, 2005.

[9] N. Eén and N. Sörensson. An extensible sat-solver. In *SAT 2003*, vol. 2919, pages 502–518, 2003.

[10] P. Flener, A. M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In *CP 2002*, vol. 2470, pages 462–476. Springer, 2002.

[11] A. M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In *Proc. of CP 2002*, volume 2470, pages 93–108. Springer, 2002.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman & Company, San Francisco, 1979.

[13] I. Gent and I. Lynce. A sat encoding for the social golfer problem. In *IJCAI'05 workshop on modelling and solving problems with constraints*, 2005.

[14] C. Gervet. Conjunto : Constraint propagation over set constraints with finite set domain variables. In *Proc. of ICLP'94*. MIT Press.

[15] F. Rossi, P. van Beek, and T. Walsh, eds. *Handbook of Constraint Programming.* Elsevier, 2006.

[16] M. Triska and N. Musliu. An improved sat formulation for the social golfer problem. *Annals of Operations Research*, 194(1) :427–438, 2012.