M2 Université de Nantes
X3IO020 2021/2022

Projet n^o 1
Set Constraint Modelling and Solving

The Social Golfer Problem (SGP) The Social Golfer Problem (problem number 10 of the CSPLib) is the following : *q golfers play every weeks during w weeks split in g groups of p golfers ($q = p.g$). How to schedule the play of these golfers such that no golfer plays in the same group as any other golfer more than once.*
Various instances of the Social Golfer Problem are still open, and the problem is attractive since it is related to problems such as encryption and covering problems.
(Problem number 26 of the CSPLib)

1 Modélisation

1.1 Modèles initiaux

Modélisez le SGP par :

1. un modèle de contraintes ensemblistes ;
2. un modèle SAT.

Plusieurs modélisations sont possibles, n'hésitez pas à donner des variantes. Tentez de donner des modèles équivalents en SET et SAT. Commentez les différences et similitudes entre modèles.

1.2 Modèles améliorés

Améliorez les modèles ci-dessus :

1. trouvez des symétries et essayez de les casser dans les modèles où c'est possible ;
2. pensez aux contraintes redondantes qui seraient possibles.

Commentez et argumentez les améliorations selon les modèles. Comparez at analysez bien les symmétries selon les modèles.

1.3 Résolution

1. implantez les modèles ci-dessus (au choix : MiniZinc, ECLiPSe, PyCSP3, Essence, MiniSAT, ...) ;
2. comparez ces différents modèles pour diverses instances du SGP ;
3. **analysez** les résultats obtenus (comparaisons entre les modèles, le cassage de symétries, ce qui est faisable dans un modèle et pas dans un autre, etc.) ; le SGP est basé sur 3 paramètres, peut être est-il judicieux d'en fixer certains, ou en tout cas, de ne pas les faire varier de façon aléatoire ;
4. proposez des améliorations, même si vous n'êtes pas en mesure de pouvoir les faire.

2 Solveur pour les contraintes ensemblistes

Au choix, réalisez l'un des trois points suivants. Il faudra bien sûr régler des problèmes de langage permis, puissance recherchée, langage d'implantation, etc. Les solveurs réalisés seront testés sur les instances ensemblistes vues au-dessus. Il y aura bien sûr une phase d'analyse très développée. Au niveau des langages pour la modélisation et des fonctionnalités, n'essayez pas d'être extensifs, mais de couvrir les besoins liés à vos modèles du SGP.

1. **Solveur ensembliste : modélisation avec des contraintes ensemblistes, et résolution de celles-ci.** Implantez un solveur ensembliste à base de propagation de contraintes (filtrage de contraintes ensemblistes). Vous avez le choix de la consistance locale, ou vous pouvez aussi tentez d'en comparer deux. L'énumération est également libre.
2. **Encodage en SAT : modélisation avec des contraintes ensemblistes, et encodage en SAT** Encodez en SAT les contraintes ensemblistes afin de les résoudre avec un solveur SAT classique (du type MiniSAT ou Glucose). Le langage pour l'encodage est libre : langage classique (C++, Java, ...) ou plus spécifique (PySAT, ...) pour générer des instances en format DIMACS.
3. **Réduction et encodage : modélisation avec des contraintes ensemblistes, filtrage basé sur les contraintes au niveau Set CSP, et ensuite encodage en SAT** Réalisez des fonctions de filtrage permettant par propagation de réduire l'espace de recherche des modèles qui seront ensuite encodés en SAT pour résolution. Attention, cette troisième option est plus longue. Il est conseillé de faire d'abord 1 ou 2, et selon le temps de complèter pour effectuer 3.

Quelque soit votre choix, comparez et **analisez** les résultats obtenus, également avec la partie précédente.

3 Délivrables

Trois parties seront considérées : le travail effectué, un rapport, et une présentation orale. Une attention particulière devra surtout être apportée aux analyses, à la recherche de solutions éventuelles permettant d'améliorer les réalisations, etc. Vous aurez à lire quelques articles afin d'être plus à l'aise.

Voici des points de départ possibles :

Conjunto : Carmen Gervet, Conjunto : Constraint Logic Programming with Finite Set Domains, ILPS 1994, pp. 339–358.

Cardinal : Francisco Azevedo, Cardinal : A Finite Sets Constraint Solver. Constraints 12(1) : 93-129 (2007)

Cardinality : Bailleux, O., Boufkhad, Y. Efficient CNF Encoding of Boolean Cardinality Constraints. In CP 2003, pp. 108—122.

minSet : Correas, J., Martın, S. E., Saenz-Perez, F. Enhancing set constraint solvers with bound consistency. Expert Systems with Applications, 2018, 92 :485–494.