Programmation SAT - Générateur de championnat

Préliminaires

Glucose. Le TME utilise Glucose, un solveur SAT performant téléchargeable à l'adresse suivante : `https://www.labri.fr/perso/lsimon/downloads/softwares/glucose-4.2.1.zip`

Installation de glucose. Une fois le fichier `glucose-syrup.tgz` téléchargé, décompressez-le à l'endroit de votre choix, puis ouvrez un terminal et allez dans le repertoire où vous avez décompressé l'archive. De là, faites `cd sources/simp`, puis `make rs`. Cela crée un exécutable `glucose_static`.

Utilisation de glucose. Pour lancer glucose, il faut lancer l'exécutable `glucose_static`. Le premier argument est le fichier `.cnf` d'entrée, qui doit contenir la théorie clausale considérée au format DIMACS (voir ci-dessous). Le second argument, facultatif, permet de préciser un fichier de sortie.

- Pour lancer glucose afin de tester la satisfiabilité de la théorie clausale `exemple.cnf` donnée ci-dessous : `glucose_static exemple.cnf` (dans le répertoire `simp`)
- Pour afficher en plus un éventuel modèle satisfaisant la formule : `glucose_static -model exemple.cnf`

Le format DIMACS. Le format d'entrée des solveurs SAT est le format DIMACS, un format très simple permettant de représenter des formules propositionnelles sous forme normale conjonctive. Ce format suppose que toutes les variables propositionnelles sont identifiées par des entiers strictement positifs. Une clause est ensuite représentée par les identifiants des variables qu'elle contient, précédés d'un signe `-` dans le cas où il s'agit de la négation de la variable considérée.

On a au final 3 types de lignes dans un fichier DIMACS

1. *Lignes de commentaires :* toute ligne commençant par `c` est considérée comme un commentaire et ignorée par le solveur.
2. *Ligne de déclaration :* la première ligne non commentée doit déclarer le nombre de variables et de clauses de la formule en respectant le format suivant : `p cnf` nb_{var} $nb_{clauses}$ où nb_{var} est le nombre de variables propositionnelles et $nb_{clauses}$ le nombre de clauses de la formule.
3. *Ligne de clause :* toutes les autres lignes doivent être constituées d'une série d'entiers suivie d'un 0. Ces lignes représentent une clause par les numéros de chacune des variables apparaissant dans la clause (entier positif ou négatif selon que la variable apparaît positivement ou négativement dans la clause) séparés par un espace, puis 0 à la fin de la clause. Pour la lisibilité on met une clause par ligne, mais le véritable marqueur de fin de clause est le 0 (qui ne correspond à aucune variable).

Exemple : on peut considérer le fichier `exemple.cnf` suivant

```
c Petit exemple illustratif
c
p cnf 5 3
-5 4 0
-1 5 3 4 0
-3 -4 0
```

Cet exemple code la formule $(\neg x_5 \vee x_4) \wedge (\neg x_1 \vee x_5 \vee x_3 \vee x_4) \wedge (\neg x_3 \vee \neg x_4)$ dans un langage comportant 5 variables x_1, x_2, x_3, x_4, x_5 (même si la variable x_2 n'apparaît pas dans la formule et n'intervient donc pas sur sa satisfiabilité, elle est supposée exister).

Exercice 1 Prise en main glucose et DIMACS

Ecrire les deux théories de l'exercice 4 de la feuille de TD en DIMACS et vérifier le modèle rendu par glucose.

Faire de même avec la formule F de l'exercice 2 pour vérifier qu'elle est UNSAT.

Problème : Organisation d'un championnat

Objectif. Le but de ce TME est de programmer la grille de match d'un championnat. On considère qu'il y a n_e équipes participantes. Les matchs de championnat peuvent se dérouler le mercredi ou le dimanche. Le championnat dure (au maximum) n_s semaines (avec 2 matchs par semaine), soit $n_j = 2 \times n_s$ jours de match.

Les matchs se font sur le terrain de l'une des deux équipes, qui est considérée comme jouant *à domicile*. L'autre équipe joue *à l'extérieur*. Etant donné les déplacements et la fatigue du match, chaque équipe ne peut jouer plus d'un match par jour. Sur la durée du championnat, chaque équipe doit rencontrer l'ensemble des autres équipes une fois à domicile et une fois à l'extérieur, soit exactement 2 matchs par équipe adverse.

Le but est de produire un planning des matchs, indiquant pour chaque jour quelles équipes s'affrontent en précisant où ont lieu les matchs.

Afin d'avoir une solution générique, paramétrable par n_j et n_e, et en raison du nombre important de variables propositionnelles et de contraintes à considérer, les exercices suivants ont pour objectif d'écrire des fonctions permettant de générer le fichier DIMACS représentant le problème. Le langage de programmation est au choix.

Exercice 2 Modélisation

Chaque équipe correspond à un numéro entre 0 et $n_e - 1$. De même, chaque jour de match correspond à un numéro entre 0 et $n_j - 1$. On considère que le championnat commence un mercredi. Les jours pairs (0 compris), correspondent donc à des mercredis, tandis que les jours impairs correspondent à des dimanches.

On représente par des variables propositionnelles $m_{j,x,y}$ le fait qu'il y ait (ou non) un match entre l'équipe x, jouant à domicile, et l'équipe y au jour j.

Au format DIMACS toutes les variables doivent être numérotées. On propose de coder la variable $m_{j,x,y}$ par v_k où $k = j \times n_e^2 + x \times n_e + y + 1$.

Question 1. Exprimer en fonction de n_e et n_j le nombre de variables propositionnelles utilisées.

Question 2. Ecrire une fonction de codage qui prend en argument n_e, n_j, j, x, y, et qui renvoie k .

Question 3. Ecrire une fonction de décodage qui retrouve j, x et y à partir de k, n_e.

Exercice 3 Génération d'un planning de matchs

Le but de cet exercice est de résoudre le problème par une méthode SAT directe. Il s'agit donc de traduire et encoder le problème en SAT, d'utiliser glucose pour résoudre le problème ainsi traduit, puis de décoder le modèle fourni en un format adapté au problème et lisible.

Question 1. **Contraintes de cardinalité.** Il s'agit ici de se doter de quelques fonctions pour faciliter l'encodage du problème avec des contraintes de cardinalité "au plus 1" et "au moins 1". Pour la contrainte "au plus 1", on considère dans cet exercice un encodage par paires.

1. Ecrire une fonction qui prend en argument une liste d'entiers et renvoie une clause, au format DIMACS, correspondant à la contrainte "au moins une de ces variables est vraie" ($\Sigma_i v_i \geq 1$).
2. Ecrire une fonction qui prend en argument une liste d'entiers et renvoie les clauses, au format DIMACS, correspondant à la contrainte "au plus une de ces variables est vraie" ($\Sigma_i v_i \leq 1$).

Question 2. **Traduction du problème.** On s'attache maintenant à traduire le problème en une formule logique propositionnelle, en passant éventuellement par l'intermédiaire de contraintes de cardinalité.

1. Traduire la contrainte C_1 "chaque équipe ne peut jouer plus d'un match par jour" en un ensemble de contraintes de cardinalité.
2. Ecrire une fonction `encoderC1` qui génère ces contraintes pour n_e et n_j donné.
3. Indiquer le nombre de contraintes et de clauses générées pour 3 équipes sur 4 jours et expliciter ces contraintes.

4. Traduire la contrainte C_2 "Sur la durée du championnat, chaque équipe doit rencontrer l'ensemble des autres équipes une fois à domicile et une fois à l'extérieur, soit exactement 2 matchs par équipe adverse." en un ensemble de contraintes de cardinalités.
5. Ecrire une fonction `encoderC2` qui génère ces contraintes pour n_e et n_j donné.
6. Indiquer le nombre de contraintes et de clauses générées pour 3 équipes sur 4 jours et expliciter ces contraintes.
7. Ecrire un programme `encoder` qui encode toutes les contraintes C_1 et C_2 pour n_e et n_j donné.

Question 3. **Utilisation du solveur.** Utiliser glucose sur la CNF générée à la question précédente et vérifier la première solution proposée pour 3 équipes sur 4 jours. Qu'est-il nécessaire d'ajouter aux deux contraintes C_1 et C_2 ?

Question 4. **Décodage.** Ecrire une fonction ou un programme `decoder` qui prend pour argument un fichier contenant la sortie de l'appel à glucose (plus éventuellement n_j et n_e) et qui traduit le modèle rendu en une solution du problème de planning des matchs affichée lisiblement. On peut faire appel à un fichier extérieur donnant le nom des équipes : une par ligne dans leur ordre de numérotation.

Question 5. **Assemblage final.** Ecrire un programme ou un script qui étant donné n_e, n_j et un fichier de noms d'équipes, affiche un planning des matchs en utilisant les programmes écrits dans les questions précédentes.

Exercice 4 Optimisation du nombre de jours

En utilisant des appels SAT de façon itérative, indiquer comment trouver le n_j minimal pour pouvoir planifier tous les matchs de championnat pour un n_e donné.

Indiquer la valeur trouvée (ou une borne minimale) pour des n_e croissants entre 3 et 10.

Si le solveur met plus de 10s à traiter le problème, on considère cela comme un Time Out (réponse inconnue).

Exercice 5 Extension : équilibrer les déplacements et les week-ends

On veut ajouter deux contraintes :

1. Matchs le dimanche.
   (a) Pour éviter de trop perturber les études des joueurs, on souhaite s'assurer que chaque équipe joue au moins $p_{ext}\%$ de ses matchs à l'extérieur des dimanches (par défaut on pose $p_{ext} = 50$).
   (b) De même, pour s'assurer que les supporters puissent assister à suffisamment de matchs, on veut que chaque équipe joue au moins $p_{dom}\%$ de ses matchs à domicile des dimanches (par défaut on pose $p_{dom} = 40$).
2. Matchs consécutifs. On préfère alterner les matchs à domicile et à l'extérieur. On impose donc que :
   (a) Aucune équipe ne joue (strictement) plus de deux matchs consécutifs à l'extérieur.
   (b) Aucune équipe ne joue (strictement) plus de deux matchs consécutifs à domicile.

Le but de cet exercice est de modifier le modèle précédent pour prendre en compte ces contraintes.

Question 1. **Traduction en contraintes de cardinalité**. Traduisez ces contraintes en contraintes de cardinalité de la forme $\Sigma_i v_i \leq k$ ($k \in \mathbb{N}$). On note que $\Sigma_i v_i \geq k$ peut s'exprimer comme $\Sigma_i \neg v_i \leq N - k$ où N est le nombre de variables de la somme.

Question 2. **Encodage de contraintes de cardinalités "au plus k"**. Pour encoder les contraintes de la question précédente, il faut pouvoir encoder des contraintes de cardinalité $\Sigma_i v_i \leq k$, ce qui nécessite d'étendre les encodages vus en cours.

Pour l'encodage "par paire", il faut considérer quelles interprétations partielles contrediraient la contrainte de cardinalité afin de produire une clause contredite pour chacune d'elles.

1. Proposer une extension de l'encodage des contraintes de cardinalité par paire qui puisse gérer des contraintes "au plus k".
2. Ecrire une fonction qui réalise cet encodage (en prenant en entrée la liste des variables et k).

Question 3. **Encodage du problème étendu.** A l'aide des questions précédentes, modifier le modèle pour tenir compte des contraintes supplémentaires d'équilibrage introduites dans cet exercice.