# INFO-F-302 Informatique Fondamentale
# Projet D'Année

Boris IOLIS

20/03/2014

## 1 Présentation du Problème

Considérons une ville où habite un ensemble de musiciens, jouant d'un ou plusieurs instruments. Le but du jeu est de former des groupes de musique, de manière à ce que :

- Chaque musicien soit dans exactement un groupe
- Chaque groupe comprend exactement un musicien dans chaque rôle (batteur, guitariste,...)
- Un musicien ne peut pas cumuler plus d'un rôle par groupe (on ne peut pas être à la fois guitariste et batteur dans un même groupe)

Pour formaliser le problème, nous avons

- Un ensemble $G$ de groupes, de taille maximale $k$
- Un ensemble $M$ de musiciens, de taille $m$
- Un ensemble $I$ d'instruments de taille $n$
- Une relation $F \subseteq M \times I$ telle que $(a, b) \in F$ ssi $a$ sait jouer de l'instrument $b$

Ces paramètres d'entrée définissent une instance du problème. Ils seront encodés dans un fichier texte de manière suivante :

```
8 4 3
1 1
2 1 2
3 2 4
4 2
5 3
6 1 3
7 1 4
8 1 2 3 4
```

Dans l'exemple ci-dessus, la première ligne contient les paramètres $m$, $n$ et $k$, séparés par des espaces. Les lignes suivantes contiennent chacune l'information concernant un musicien ; chaque ligne commence par le numéro

du musicien (son identifiant), et reprend ensuite tous les instruments dont il sait jouer (tous les $i \in I$ tels que $(a, i) \in F$ où $a$ est l'identifiant du musicien en question). Dans cet exemple, on considère 8 musiciens jouant de 4 instruments différents et on essaye de former au plus 3 groupes. Le musicien 1 sait jouer uniquement de l'instrument 1, le musicien 6 sait jouer des instruments 1 et 4, tandis que le musicien 8 maitrise tous les instruments de $I$.

Une solution au problème (répartition des musiciens en groupes) devra suivre la forme suivante :

```
groupe 1: 1 2 5 7
groupe 2: 6 4 8 3
groupe 3:
```

Chaque ligne du fichier représente la composition d'un groupe ; après la mention *"groupe g :"*, où $g$ est le numéro du groupe, sont listés les musiciens qui composent le groupe, par ordre d'instrument. Dans cet exemple de solution, le groupe 1 est composé du musicien 1 jouant de l'instrument 1, du musicien 2 jouant de l'instrument 2, du musicien 5 jouant de l'instrument 3, et du musicien 7 jouant de l'instrument 4. Notez que pour former un groupe, il faut que chaque instrument de $I$ soit représenté une et une seule fois. On remarque que le groupe 3 est vide. La solution ne doit pas forcément contenir exactement $k$ groupes (mais au plus $k$). Les groupes qui restent vides ne doivent pas forcément figurer en fin de fichier (dans cet exemple, le groupe 2 aurait pu rester vide et le 3 être rempli).

Dans le cas où l'instance n'est pas satisfaisable, le fichier résultat devra contenir une seule ligne, avec la mention

```
impossible
```

# 2 Outils

## 2.1 MiniSAT

MiniSat, disponible sur `http://minisat.se/MiniSat.html`, est un solveur SAT développé en C++ qui travaille avec des formules de logique propositionnelles sous forme clausale. MiniSat fournit comme résultat une assignation des variables si le problème est satisfaisable. Pour l'utiliser, vous devez lui fournir un fichier texte au format DIMACS simplifié. Ce format est le suivant :

- Chaque ligne commençant par un c est un commentaire.
- La première ligne (hors commentaires) doit correspondre à $p\ cnf\ variables\ clauses$ où *variables* est le nombre de variables et *clauses* le nombre de clauses

– Chaque autre ligne est une clause finissant par 0 dont les variables sont séparées par des espaces. Les variables sont représentées par leur numéro d'ordre allant de 1 à *variables*.
– La négation d'une variable est représentée par le signe "-".

Par exemple :

```
c Commentaire.
p cnf 2 3
-1 0
2 0
1 -2 0
```

Remarque : si vous choisissez d'implémenter vos programmes en C++, vous pouvez utiliser la librairie C++ de MiniSat (voir les exemples C++ du cours). Le fichier de sortie quant à lui respectera le format de MiniSat qui est également un fichier texte. La première ligne se compose du mot clé UNSAT si le problème n'est pas satisfaisable et SAT si le problème est satisfaisable. Dans ce dernier cas, la seconde ligne est une interprétation des variables (comme les clauses au format DIMACS).

### 2.2 Scarab

Il s'agit d'une interface pour encoder des problèmes de satisfaisabilité. Scarab fournit un langage d'encodage de contraintes, basé sur Scala, qui permet de représenter des problèmes logiques sans passer par l'encodage des contraintes individuelles forme normale conjonctive, comme ce serait le cas de solveurs bas-niveau comme MiniSAT. L'avantage de ce genre d'outil est de pouvoir encoder des problèmes complexes en quelques lignes, et de manière bien plus intuitive. Par exemple, il est possible d'encoder des contraintes arithmétiques sans devoir les formuler en logique propositionnelle. Scarab utilise Sat4J comme solveur sous-jacent et effectue lui-même la traduction des contraintes en logique propositionnelle.

Scarab est disponible à l'adresse suivante, qui contient également un tutoriel et des exemples : `http://kix.istc.kobe-u.ac.jp/~soh/scarab/index.html`. Pour utiliser Scarab il vous faudra coder en langage Scala (n'oubliez pas de l'installer, comme spécifié sur le site de Scarab !). Vous pouvez vous servir des exemples donnés sur le site comme point de départ. Si vous souhaitez utiliser Eclipse, il existe un plugin Scala qui fonctionne très bien. Il est disponible ici : `http://scala-ide.org/`

## 3 Questions

Pour chaque question, il vous est demandé d'écrire un fichier, nommé selon le format Question*i*.bin pour le binaire C++ (accompagné d'un Ma-

kefile), ou Question$i$.jar pour Java et Scala (si vous optez pour Scarab), où $i$ est le numéro de la question, qui prendra en paramètre

- un nom de fichier qui contient une instance du problème au format défini plus haut
- un nom pour le fichier de sortie.

### 3.1 Question 1 (8 points)

**Partie a** Expliquez dans votre rapport votre représentation en forme normale conjonctive du problème tel qu'il est décrit plus haut. Utilisez les variables $X_{a,b,c}$ telles que, pour tout musicien $a$, instrument $b$ et groupe $c$, $X_{a,b,c} = vrai$ ssi le musicien $a$ joue de l'instrument $b$ dans le groupe $c$.

**Partie b** Implémentez votre solution en utilisant soit MiniSAT soit Scarab.

### 3.2 Question 2 (7 points)

Pour cette question, nous allons complexifier le problème initial en ajoutant la possibilité aux musiciens de rejoindre plus d'un groupe. Chaque musicien $a$ se verra attribuer une contrainte $Max_a > 0$ qui spécifiera le maximum de groupes dans lequel le musicien $a$ peut jouer. On peut voir ça comme une contrainte de temps libre (certains musiciens ont davantage de temps libre que d'autres pour rejoindre des groupes). A noter qu'il n'est toujours pas possible pour un musicien de cumuler plus d'un instrument au sein d'un même groupe, et que chaque musicien doit toujours être affecté à au moins un groupe.

Le format de fichier en entrée est modifié de manière suivante :

```
8 4 3
1 2 1
2 1 1 2
3 2 2 4
4 1 2
5 1 3
6 1 1 3
7 1 1 4
8 3 1 2 3 4
```

Il s'agit du même exemple que précédemment, à la différence que chaque ligne décrivant un musicien contient à présent, juste après l'identifiant du musicien, sa contrainte $Max_a$. On voit ici que tous les musiciens sont limités à un groupe, excepté les musiciens 1 et 3 qui peuvent en rejoindre 2, et le musicien 8 qui peut en rejoindre 3. Voici un exemple de solution qui devient

acceptable dans ce cas-ci (à noter que la solution présentée pour la variante précédente reste tout de même valide) :

```
groupe 1: 1 2 8 7
groupe 2: 6 4 8 3
groupe 3: 1 3 5 8
```

**Partie a** Cette modification du problème introduit une contrainte de type *« au plus k »*. Expliquez dans votre rapport comment encoder ce type de contraintes par une formule de logique propositionnelle en forme normale conjonctive, ainsi que le nombre de clauses que votre encodage créera dans le cas présent (en fonction des paramètres $n$, $m$ et $k$). Essayez d'avoir un encodage le plus efficace possible en termes de nombres de variables et de clauses. Pour cela, vous pouvez par exemple essayer de généraliser l'encodage binaire vu en cours pour la contrainte *« exactement une »* (voir séance 6, slide 264) [1].

**Partie b** Implémentez votre solution en utilisant soit MiniSAT soit Scarab.

### 3.3 Question 3 (5 points)

Pour cette question, nous allons complexifier le problème de la question précédente, en y ajoutant la composante du chant. Le chant sera considéré comme un instrument spécial, dans la mesure où un musicien peut cumuler le chant avec un autre instrument au sein d'un même groupe, si il maitrise le chant ainsi que cet autre instrument bien entendu. Cela revient à dire, de manière intuitive, qu'un guitariste peut aussi chanter par exemple [2]. Dès lors, un groupe peut à présent avoir plus d'un chanteur (mais toujours au moins un). A noter que la contrainte $Max_a$, introduite lors de la question précédente, est calculée en fonction du nombre de groupes dans lesquels joue $a$, indépendamment du nombre de rôles qu'il cumule au sein de chaque groupe.

De manière pratique, on considèrera le chant comme l'instrument numéro $n$ dans l'ensemble $I$ (donc le dernier de la liste). Le format de fichier en entrée reste le même. Par contre, le format de fichier en sortie reste également le même ; dans le cas où un groupe se voit assigner plusieurs chanteurs, ils sont énumérés en fin de liste, comme suit :

```
groupe 1: 1 2 5 7 3 8
groupe 2: 6 4 8 3 8
groupe 3:
```

1. Cette généralisation est une question bonus.
2. On supposera donc, pour des raisons évidentes, que l'ensemble d'instruments $I$ ne contient pas d'instruments à composante de souffle...

Dans cet exemple de solution, qui respecte les contraintes présentées dans l'exemple de la question 2, on voit que le groupe 1 a maintenant trois chanteurs (7, 3, et 8), tandis que le groupe 2 en a deux (3 et 8). A noter que le musicien 8 cumule son rôle de chanteur avec celui de l'instrument 3. La liste de chanteurs d'un groupe ne doit pas nécessairement être triée.

**Partie a** Expliquez l'impact de cette modification sur votre représentation du problème en forme normale conjonctive de la question précédente

**Partie b** Implémentez votre solution en utilisant soit MiniSAT soit Scarab.

### 3.4 Question 4 (Bonus)

Cette question sort quelque peu du cadre de ce cours. Lors des questions précédentes, il vous a été demandé de trouver une solution satisfaisant le problème, quelle qu'elle soit. Pour cette question, vous devrez trouver une solution qui se rapproche le plus possible de la contrainte $Max_a$ pour tous les musiciens $a \in M$. Pour cela, la solution devra faire en sorte que la somme du nombre de groupes dans lesquels joue chaque musicien se rapproche le plus possible de la somme des $Max_a$ pour tout $a \in M$, sans pour autant enfreindre aucune des contraintes $Max_a$. Les modifications apportées au problème lors des questions 2 et 3 restent d'application dans ce cas.

**Partie a** Expliquez comment vous comptez résoudre ce problème d'optimisation, en vous basant sur les mêmes outils que précédemment. Faites-en sorte que votre méthode effectue une recherche plus ou moins intelligente, ou en tout cas soit plus efficace qu'une recherche exhaustive.

**Partie b** Implémentez votre solution en utilisant soit MiniSAT soit Scarab.

## 4 Remarques Additionnelles

Pour toutes les questions, vous pouvez choisir d'utiliser soit MiniSAT soit Scarab. MiniSAT est plus simple d'utilisation et plus efficace, alors que Scarab est plus puissant, et vous permettra donc d'implémenter plus facilement vos contraintes, une fois que vous en aurez compris le fonctionnement. Le choix de l'un ou l'autre outil n'aura pas d'impact sur la cotation du projet.

Si vous utilisez Scarab, vous n'aurez pas besoin de traduire vous-mêmes vos contraintes en forme normale conjonctive. Il vous est tout de même demandé d'expliquer la manière d'effectuer une telle traduction dans votre rapport (**partie a** de chaque question).

Que vous utilisiez Scarab ou MiniSAT, vous devrez respecter le format de sortie présenté en Section 1. Le format brut retourné par MiniSAT ne sera pas accepté !

Des exemples pour chaque question sont fournis avec l'énoncé. Les fichiers résultat correspondant à chaque exemple contiennent un exemple de solution ; il est donc possible de trouver d'autres solutions correctes.

Pour finir, veillez à ne pas trop négliger la forme de votre rapport, ni l'efficacité et l'ergonomie de votre implémentation.

# 5 Modalités

Le projet se fait en binôme. Le rapport papier est à remettre au secrétariat étudiant au plus tard le **Mercredi 21 Mai à midi** (heure de Bruxelles). Ce rapport devra inclure les réponses aux questions ainsi que tout détail d'implémentation que vous jugerez pertinent de mentionner.
Une version électronique est à envoyer, pour la même date et heure, à Boris Iolis (biolis@ulb.ac.be). Elle devra être envoyée sous format ZIP, et comporter les NetIDs des deux étudiants. Elle devra contenir :

- Le rapport au format pdf
- Les exécutables pour chaque question
- Le code source