**Rapport TP2**

# 

# **Intelligence artif. : méthodes et algorithme**

**INF8215**

**Membres de l’équipe**

**Gabriel Campbell [1761276]**

**Gabriel St-Onge [1792574]**

**Kevin Takla [1799649]**

**Remis le 11 novembre 2018**

**Exercice 1)**

Avant tout, nous avons modélisé ce problème avec 5 tableaux pour chaque caractéristique soit : la couleur de maison, la boisson, la profession, l’animal de compagnie et la nationalité. Pour chaque tableau, nous allons placer les éléments selon la position de la maison dans le voisinage de gauche à droite. Nous avons procédé de cette façon, car certaines affirmations prennent en compte la position des maisons. À partir de ces définitions, nous pouvons créer nos contraintes. Les 5 premières contraintes que nous avons faites vérifient que chaque individu a des caractéristiques uniques par catégories. Après ceux-ci, nous pouvons construire une contrainte pour chacune des affirmations.

**Exercice 2)**

Dans le cas de ce problème, nous avons modélisé notre problème dans une matrice de taille n x (n – 1) ou n représente le nombre d’équipe qui participent au tournoi. Dans cette matrice, les lignes représentent le calendrier pour une équipe et les colonnes représentent un tour. Nous considérons que ce modèle est la meilleure façon de représenter le problème de round-robin.

À partir de ce modèle, nous pouvons créer nos contraintes. Premièrement, nous avons créé une contrainte afin que les équipes jouent contre toutes les autres équipes une seule fois. Deuxièmement, nous avons créé une contrainte afin d’empêcher qu’une équipe s’affronte lui-même durant le tournoi. Troisièmement, nous avons ajouté une contrainte afin de nous assurer que deux équipes s’affrontent durant le même tour. Quatrièmement, nous avons mis une contrainte afin de nous assurer qu’aucune équipe ne joue plus de 4 matches consécutifs à domicile ou à l’extérieur. Pour ce faire, nous avons utilisé le tableau location afin d’additionner tous les groupes de 4 matchs consécutifs. Avant tout, nous savons que cette matrice retourne la valeur 1, pour une équipe quelconque, si son match contre une équipe y est à domicile et cette matrice retourne 0 dans le cas contraire. Avec ce constat, si la somme d’un groupe des 4 matchs est égale à 4 ou à 0, cela veut dire ces 4 matchs-là se seront tous produits à domicile ou à l’extérieur. Bref, notre contrainte va restreinte tous les groupes de 4 matchs à une valeur entre 1 et 3.

Dans cette situation, nous avons remarqué un problème de symétrie. En fait, notre tableau de localisation des matchs ne retourne que les valeurs 1 ou 0. En plus, la contrainte avec les matchs à domicile/extérieur que nous avons ajoutés ne prennent pas en compte les permutations des équipes participant au tournois. Donc, cela peut devenir trop lourd pour le solveur. Pour régler ce problème, nous avons ajoutés une dernière contrainte qui dit que même si deux matchs se jouent dans le même contexte (à domicile ou à l’extérieur) que ce n’est pas la même rencontre. Par ce changement, nous avons passé à un modèle qui prenaient un temps infini à trouver une solution à un autre modèle qui a prend environ 5 secondes à résoudre. Cela peut s’expliquer par le fait que le solveur peut faire la différence entre deux matchs sans se préoccuper de leur localisation.

**Exercice 3)**

Pour ce problème, nous avions droit à un maximum de 2 paramètres par prédicats. Alors nous avons tout simplement donné la catégorie cours pour tous les cours et ensuite nous avons créé des catégories de prérequis et de corequis, chacun ayant comme premier paramètre un cours et come deuxième paramètre, le cours prérequis ou corequis à ce dernier. Cela définissait donc la relation entre les deux cours.

Pour ne pas avoir à réécrire les corequis symétriques dans la base de connaissance, nous avons donc défini une règle qui permet de prendre en compte les corequis symétriques en inversant les deux paramètres.

La méthode pour résoudre ce problème est d’utiliser un tableau qui est initialement vide. Donc, pour faire fonctionner le programme, l’utilisateur entre le prédicat *coursAPrendreComplet* et comme premier paramètre il entre le cours qu’il veut suivre et le deuxième paramètre servira comme nom de tableau ayant tous les cours à prendre. Il faut ce nom soit tout en majuscule.

Le tableau est initialement vide. On le remplit en allant chercher les cours prérequis et corequis. Ensuite, on peut itérer le tableau par récursion à chercher les cours prérequis et corequis des cours qui sont dans le tableau que nous venons d’ajouter. Nous faisons aussi attention à ne pas avoir de doublons, donc que le cours ne soit pas déjà dans le tableau lorsque nous le cherchons.  
  
 Lorsque nous avons tout terminé, il ne reste plus qu’à afficher chaque élément du tableau.

**Exercice 4)**

Pour ce problème, le programme est séparé en 2 grandes catégories. Celui pour identifier les objets et l’autre pour les personnages. Du côté des personnages, la base de connaissance est principalement séparée en catégorie selon les professions des personnes. Exemple, nous avons des catégories auteur, acteur, réalisateur, politicien et plusieurs autres. Il a fallu pouvoir distinguer ces personnes lorsque nous avions plusieurs personnes dans la même catégorie.

Pour avoir une très bonne distinction, nous avons créé la catégorie *homme\_personne* qui permet définir que cette personne est un homme, donc si la personne ne se retrouve pas dans cette catégorie, cela veut dire qu’elle est une femme.

Si nous nous retrouvions avec plusieurs hommes dans une même catégorie, alors nous pouvions les séparer selon leur pays d’origine, selon leur période de naissance ou tout autre distinction particulière qui permet d’avoir une catégorie assez grande où est-ce que l’on pourrait y ajouter une autre personne dans cette catégorie aussi.  
 La première question que le programme pose, c’est de savoir si la personne est un homme ou non. Cela nous permet donc d’éliminer la liste des hommes ou pas. Ensuite, nous posons la question au niveau de la profession et puis après nous essayons de distinguer ce personnage par rapport aux autres personnages de la même profession. Finalement, nous pouvons avoir notre résultat. C’est tout simplement le modèle d’un arbre que nous suivons.  
  
 Pour ce qui est de la catégorie objet, le principe est le même, cependant dans la base de connaissance, nous séparons les objets qui utilise de l’électricité en ayant une catégorie pour eux. Puis, nous avons une autre catégorie qui définit l’objet comme étant un objet qui permet de nettoyer quelque chose avec. Les autres catégories sont assimilées de façon générale pour les objets. Par exemple, l’objet fourchette appartient à la catégorie ustensile ou encore l’objet cactus appartient à la catégorie plante, ce qui facilite l’ajout de nouveaux objets dans ces catégories.  
  
 La première question que l’on pose est de savoir si cet objet consomme de l’électricité ou non. Par la suite, nous essayons de déterminer où cet objet pourrait être. Nous demandons si cet objet pour être dans une cuisine, ou bien si l’on pourrait le porter sur nous ou encore s’il serait dans un bureau, etc. Par la suite, on essaie de déterminer si cet objet est petit, gros ou bien tout autre distinction que l’on pourrait faire avec les catégories similaires.  
   
 Nous pouvons ensuite obtenir notre résultat après ces quelques questions.