Rapport de Projet Tuteuré - ANDROIDE

Alexandre Bontems, Gualtiero Mottola, Hans Thirunavukarasu

17 mai 2018

Table des matières

1. Introduction

This is the intro

2. Analyse d'instances

Dans la partie théorique de ce projet, nous nous sommes attelés au calcul de métriques diverses permettant de résumer les caractéristiques d'une instance donnée. Puisque l'évaluation de la difficulté d'une instance se fait par rapport à une résolution par l'humain, plusieurs hypothèses ont été posées concernant les méthodes de résolution utilisées par un joueur. Elles dérivent directement d'observations réalisées avec l'application mobile développée dans le cadre de ce projet. Par la suite, les différentes métriques conçues sont détaillées ainsi que les hypothèses à leur source.

2.1. Solvabilité d'une instance

Afin d'obtenir des instances pertinentes pour notre analyse, il a d'abord été essentiel d'étudier leur solvabilité. Puisque le voisinage de chaque agent est connu, il a été possible de générer des instances comptant au moins une solution. L'idée est de choisir une allocation aléatoire, c'est-à-dire l'indice, dans les liste de préférences, de l'objet qui sera alloué à chaque agent. On s'assure ensuite qu'un agent donné ne préfère pas les objets choisis pour ses voisins à sa propre allocation. Le pseudo-code suivant a été implémenté dans ce projet.

```
1  Indices := []
2  Pour chaque agent a:
3      Si a est le premier agent ou le dernier agent:
4          Indices[a] := valeur aléatoire entre 1 et n-1
5      Sinon:
6          Indices[a] := valeur aléatoire entre 1 et n-2
7
8  Pour chaque agent a:
9      Prefs[a] := []
10     ValeurPossibles := {1, .., n}\(Indices[Voisins[a]] et Indices[a])
11     Pour chaque indice i des préférences:
12         Si i < Indices[a]:
13             k := valeur aléatoire parmi ValeurPossibles
14             Prefs[a, i] := k
15             ValeurPossibles := ValeurPossibles\{k}
16         Si i = Indices[a]:
17             Prefs[a, i] := Indices[a]
18         Si i > Indices[a]:
19             k := valeur aléatoire parmi ValeurPossibles
20             Prefs[a, i] := k
21             ValeurPossibles := ValeurPossibles\{k}
```

Un exemple d'instance générée se trouve en ADD REF

2.2. Résolution par backtracking

La première approche d'analyse abordée a été de résoudre le problème à l'aide d'un algorithme de backtracking. En effet, il s'est rapidement avéré évident qu'un processus similaire pouvait être utilisé comme méthode de résolution par un humain. De plus, il est plus facile de choisir des objets situés haut dans les listes de préférences car la possibilité de jalousie est moins probable. On en déduit donc l'ordre d'instantiation des variables de l'objet préféré au moins aimé pour chaque agent.

Attention, l'explication du problème peut influer sur l'ordre d'instantiation pour un humain.

Solutions Pareto-optimales L'algorithme de backtracking permet de trouver l'ensemble des solutions Pareto-optimales d'une instance. Ce sont par définition les instance

Nombre d'itérations