

## Projet : Partage de biens d'une colonie spatiale (partie 1)

## I Problématique

Après chaque mission de ravitaillement, le commandant d'une colonie spatiale doit répartir des ressources critiques entre les colons. Parmi ces ressources, chaque colon doit recevoir une des ces ressources critiques (e.g., un équipement ou une ration de vivres essentiels). Pour maintenir l'harmonie dans la colonie, le commandant demande à chaque membre de lui soumettre ses préférences sur les ressources allouées et doit tenter de les respecter au mieux. En plus de ces préférences, il doit tenir compte des relations entre les colons. Certains s'entendent mal, et une mauvaise répartition des ressources pourrait entraîner des conflits ou mettre en danger la survie de la colonie...

Le commandant nous demande donc de l'aider, en développant un logiciel qui permet de :

- 1) représenter les colons et les relations entre eux;
- 2) simuler le partage des ressources entre colons;
- 3) calculer le coût d'une affectation (i.e., le nombre de colons jaloux), et de le minimiser.

Nous réaliserons ces tâches de façon incrémentale, afin d'avoir à la fin du semestre un logiciel fonctionnel.

## II Modélisation

Une instance du problème est composée d'une colonie de  $n$  colons, qui doivent se partager  $n$  ressources (avec  $n \geq 1$ ). On peut représenter les relations entre les membres de la colonie comme un graphe non orienté, dans lequel chaque noeud représente un colon, et chaque arête dans le graphe représente une relation « ne s'aime pas » entre deux colons. En plus du graphe, on connaît les préférences de chaque colon concernant les ressources à partager. Pour un colon donné  $c$ , ses préférences peuvent être représentées par une relation d'ordre strict. La Figure 1 décrit un exemple où la colonie est composée de 4 colons  $A, B, C$  et  $D$  qui doivent se partager 4 objets  $o_1, o_2, o_3$  et  $o_4$ . Les colons s'entendent bien dans l'ensemble, sauf le colon  $B$  qui n'aime personne (et n'est aimé de personne). On voit par exemple que les préférences de  $A$  sont  $o_1 >_A o_2 >_A o_3 >_A o_4$  ( $o_1$  est sa ressource préférée, puis  $o_2$ , etc.). Les préférences des autres colons ( $>_B, >_C$  et  $>_D$ ) sont également données.

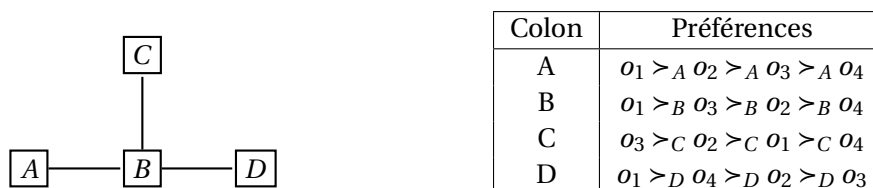


FIGURE 1 – Un exemple de colonie avec les préférences des colons

Une solution du problème est une affectation d'un objet à chaque colon. Un colon est jaloux si un des colons qu'il n'aime pas obtient un objet qu'il aurait préféré avoir. Par exemple, si le colon  $B$  obtient  $o_3$ , et  $C$  obtient  $o_2$ , alors  $C$  est jaloux de  $B$  car il aurait préféré avoir l'objet d'un des colons qu'il n'aime pas. Le nombre de colons jaloux représente le coût d'une affectation. Remarque : un colon ne compte que pour 1 dans le coût de la solution, même s'il est jaloux de plusieurs de ses voisins.

## 1. Exemples

Regardons ce qu'il se passe avec deux affectations de ressources possibles à chacun des colons.

Pour la première affectation (Figure 2), le commandant a décidé d'affecter la ressource  $o_1$  au colon  $A$ , la ressource  $o_2$  au colon  $B$ , la ressource  $o_3$  au colon  $C$  et la ressource  $o_4$  au colon  $D$ . On observe donc que :

- $A$  reçoit sa ressource préférée donc il n'est pas jaloux. Même chose pour le colon  $C$ .
- $D$  reçoit son deuxième objet préféré. Il aurait préféré  $o_1$ , mais ce n'est pas grave dans ce cas car aucun des colons qu'il n'aime pas (ici en l'occurrence  $B$ ) n'a reçu cet objet.  $D$  n'est donc pas jaloux.
- $B$  est jaloux de  $A$  qui a reçu  $o_1$  (car  $o_1 >_B o_2$ ) et de  $C$  qui a reçu  $o_3$  (car  $o_3 >_B o_2$ ).

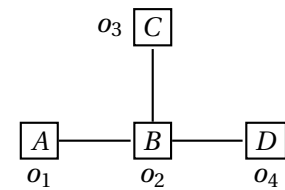


FIGURE 2 – Affectation 1

Le coût de cette affectation est donc 1 (seul  $B$  est jaloux).

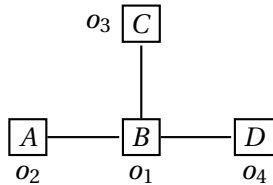


FIGURE 3 – Affectation 2

Pour la deuxième affectation (Figure 3), le commandant a décidé d'affecter la ressource  $o_2$  au colon  $A$ , la ressource  $o_1$  au colon  $B$ , la ressource  $o_3$  au colon  $C$  et la ressource  $o_4$  au colon  $D$ . On observe donc :

- $B$  et  $C$  ont reçu leur objet préféré, ils ne sont donc pas jaloux.
- $A$  a reçu l'objet  $o_2$ , or il aurait préféré l'objet  $o_1$  qui a été attribué à  $B$ . Donc  $A$  est jaloux de  $B$ . Même remarque pour  $D$  avec  $o_4$  qui est jaloux de  $B$ .

$A$  et  $D$  sont tous les deux jaloux de  $B$ , donc le coût de cette affectation est 2. Cette affectation est donc moins bonne que la précédente (car le but est

de minimiser le nombre de colons jaloux).

## 2. Remarques et Restrictions

Concernant les restrictions, il est important de noter que dans le cadre de la réalisation de ce projet :

- On considère qu'il y aura toujours autant de ressources que de colons. Cela implique qu'il ne faut pas traiter les cas où il y a moins de ressources que de colons et inversement.
- La relation de préférences pour un colon est une relation d'ordre strict, i.e., on peut comparer tous les couples de ressources et il ne peut pas y avoir deux ressources avec le même niveau de préférence.
- La relation "ne s'aime pas" est une relation symétrique, i.e., si  $A$  n'aime pas  $B$  alors cela implique forcément que  $B$  n'aime pas  $A$  également.

Et quelques remarques supplémentaires :

- Pour certaines configurations, il sera impossible de trouver une affectation avec un score de 0 (i.e., aucun colon n'est jaloux). Dans ce cas, la solution optimale sera la ou les affectations qui minimisera le nombre de colons jaloux.
- Il y a forcément une solution optimale (i.e., affectation qui minimisera le nombre de colons jaloux) pour chaque configuration de colonie possible.
- Pour une configuration de colonie donnée, le score minimal peut être obtenu par une ou plusieurs affectations possibles.

### III Instructions

#### 1. Tâches à réaliser

Pour la première étape du projet, vous devez développer un programme qui permet à un utilisateur de configurer manuellement la colonie. Au démarrage, le programme doit demander à l'utilisateur le nombre de colons  $n$ . Ce nombre sera également égal au nombre d'objets que les colons doivent se partager. On considèrera pour l'instant que les colons sont nommés par les lettres de l'alphabet en majuscule, le premier s'appelant  $A$ , le deuxième  $B$ ,... Le projet sera donc testé avec au plus 26 colons (pour la partie 1, mais cela changera dans la partie 2). Les ressources/objets sont numérotés de 1 à  $n$ .

Une fois que le nombre de colons  $n$  est fixé par l'utilisateur, un menu s'affiche avec deux options possibles :

- 1) ajouter une relation entre deux colons ;
- 2) ajouter les préférences d'un colon ;
- 3) fin.

Dans le cas où l'option 1 est retenue, on demande à l'utilisateur les colons entre lesquels il faut ajouter une relation «ne s'aiment pas», puis on revient au menu précédent. Dans le cas où l'option 2 est retenue, l'utilisateur doit d'abord taper le nom d'un colon, suivi de la liste qui correspond aux préférences du colon. Par exemple,

A 1 2 3

signifie que le colon préfère l'objet 1, puis l'objet 2, puis l'objet 3. Veillez à bien séparer les informations par (au moins un) espace. Dans le cas où l'option 3 est retenue, l'utilisateur a terminé de représenter les relations entre les membres de la colonie. Dans ce cas, vous devez vérifier que chaque colon possède bien une liste de préférences. Si ce n'est pas le cas, vous donnerez le nom des colons pour lesquelles il y a un problème avec ses préférences (absentes, incomplètes, ...).

L'utilisateur peut maintenant essayer de trouver une solution au problème. En effet, une fois la construction de la colonie terminée et validée, **une première solution naïve est proposée automatiquement à l'utilisateur** : chaque colon (dans l'ordre de votre choix) reçoit son objet préféré s'il est disponible, ou son deuxième objet préféré s'il est disponible,...

Ensuite, l'utilisateur aura la possibilité de changer manuellement l'affectation des objets entre deux colons afin d'espérer trouver une meilleure affectation. Pour cela, il fait face à un menu qui propose trois options :

- 1) échanger les ressources de deux colons ;
- 2) afficher le nombre de colons jaloux ;
- 3) fin.

Quand l'option 1 est sélectionnée, on demande à l'utilisateur d'indiquer les noms de deux colons, puis le programme échange les ressources affectés à ces deux colons. Quand l'option 2 est sélectionnée, le programme affiche le nombre de colons jaloux avec l'affectation actuelle. Après chaque action de l'utilisateur, un message rappelle qui possède actuellement quel objet, avec le format suivant :

A : 2

B : 1

C : 3

D : 4

qui correspond à l'affectation décrite dans la Figure 3.

Enfin, si l'option 3 est sélectionnée, le programme s'arrête.

Il est nécessaire de gérer certaines erreurs. Par exemple, il faudra vérifier que les préférences rentrées par l'utilisateur concernent un colon qui existe et que les préférences ont été donné sur toutes les ressources existantes. Même chose pour l'échange de ressources dans le seconde menu, il faudra vérifier que les colons donnés par l'utilisateur existe bien. Un autre exemple d'erreur à gérer concerne le menu. En effet, quand le menu propose trois options, il faut indiquer à l'utilisateur que sa réponse est incorrect s'il essaye de taper 4. Par contre, il n'est pas demandé (du moins pour cette première partie) de gérer les erreurs dues à l'entrée

d'une information de mauvais type (par exemple si l'utilisateur entre le nombre 1,5 au lieu d'un nombre entier). Ce type d'erreur sera à gérer dans la second partie.

## **2. Remise du projet**

Ce projet est à réaliser par groupes de **deux ou trois étudiants** à constituer vous-mêmes, issus du **même groupe de TD**. Votre code source de cette première partie (et uniquement de cette partie), correctement documenté, sera à remettre sur Moodle au plus tard le **8 novembre 2024**, sous forme d'une **archive jar ou zip (un seul dépôt par binôme/trinôme)**. Cette partie ne sera pas évaluée mais chaque groupe devra faire une démonstration de son programme à son chargé de TD durant la séance de TP la semaine du 25 novembre. Celui-ci vous donnera des conseils, si nécessaire, aussi bien sur le rendu du programme mais aussi sur les choix de conception faits et le code réalisé.