TD 4 : Séparation et évaluation

Recherche opérationnelle S3.

2024

Exercice 1 — *Sélection contrainte*

Dans le tableau suivant vous sont donnés des armes/accessoires de Counter Strike, leur coût en $ et leur utilité, en points. Vous disposez de 4200$. On souhaite choisir l'ensemble d'utilité maximum tout en respectant le budget fixé. Petite contrainte pour simplifier : pour chaque catégorie, on doit prendre un seul et unique objet.

	A : Arme de poing	B : Fusil	C : Grenade	D : Protection
1	Bereta 500$ 10pt	AK-47 2700$ 60 pt	Explosive 300$ 40pt	Kevlar 650$ 70pt
2	Desert Eagle 650$ 50pt	FAMAS 2250$ 40pt	Flash-Bang 200$ 20pt	Kevlar-Casque 1000$ 80pt

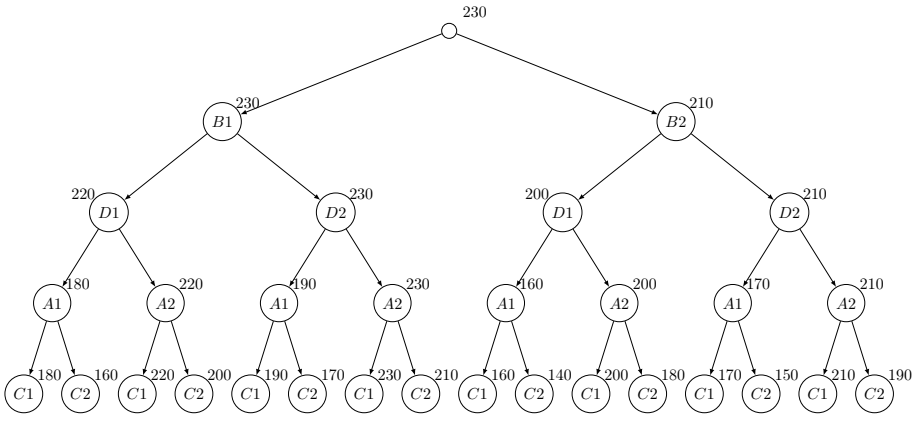
On souhaite résoudre ce problème avec une méthode de séparation et évaluation. A chaque itération, on force un choix pour une catégorie d'objets. Chaque noeud de l'arbre ci-après indique ce choix par la colonne et la ligne correspondante du tableau. Par exemple, la branche B1D2A1C1 correspond au AK-47, Kevlar-Casque, Bereta, Grenade explosive. Le sous-arbre de B2D2 est l'ensemble des solutions comprenant le FAMAS et le Kevlar-Casque. A côté de ce noeud est inscrit une borne supérieure de l'utilité que l'on peut atteindre avec ce choix, autrement dit, le mieux que l'on puisse espérer en tenant compte des choix déjà faits.

La formule de cette borne supérieure est la suivante :

— pour chaque catégorie où on a déjà choisi un objet, on ajoute l'utilité de l'objet choisi

— pour tout autre catégorie, on ajoute l'utilité max parmi celles des objets de cette catégorie.

Par exemple, pour la racine, on a fait aucun choix. Le maximum que l'on puisse espérer atteindre est 50+60+40+80 avec l'ensemble Desert Eagle, Ak47, Grenades explosives et Kevlar Casque. Pour B2D2, on force le choix du FAMAS et du Kévlar casque, d'utilité 40 + 80, on rajoute ensuite l'utilité du Desert Eagle et des grenades explosives (le meilleur des pistolets et la meilleure des grenades d'après le tableau), soit 50 + 40, ce qui donne 210.



1. Combien de noeuds de l'arbre explorerait un algorithme qui ne tient pas compte des bornes, sachant qu'il n'explore pas un noeud si le budget de 4200$ est déjà dépassé ?
2. Si vous utilisez l'algorithme vu en cours, qui adopte la stratégie de parcours en profondeur à gauche, dans quel ordre exploreriez-vous les noeuds de l'arbre et combien de noeuds exploreriez-vous ?

3. Si vous adoptez la stratégie de parcours en profondeur à droite, dans quel ordre exploreriez-vous les noeuds de l'arbre et combien de noeuds exploreriez-vous ?
4. Si vous adoptez la stratégie d'explorer toujours le meilleur (borne supérieur la plus grande), dans quel ordre exploreriez-vous les noeuds de l'arbre et combien de noeuds exploreriez-vous ? Pourriez vous expliquer ce qu'il se passe ? Est-ce le cas dès qu'on utilise cette stratégie ?
5. Si vous adoptez la stratégie d'explorer toujours le pire (borne supérieur la plus petite), dans quel ordre exploreriez-vous les noeuds de l'arbre et combien de noeuds exploreriez-vous ?

► **Correction**

Remarque : Je désigne un nœud pas son nom et l'ensemble des noms de ses ancêtres.

Pour la première question, on calcule le budget de toutes les solutions partielles et on ne conserve que les solutions de coût 4200 ou moins. Il n'explorerait donc pas les nœuds $B1D1A2C1$, $B1D2A1C1$, $B1D2A1C2$, et $B1D2A2$.

La stratégie de profondeur à gauche explore dans cet ordre :

Racine, $B1$, $B1D1$, $B1D1A1$, $B1D1A1C1$ (nouvelle solution, 180), $B1D1A1C2$ (coupé), $B1D1A2$, $B1D1A2C2$ (non réalisable), $B1D1A2C2$ (nouvelle solution, 200), $B1D2$, $B1D2A1$ (coupé), $B1D2A2$, (non réalisable), $B2$, $B2D1$ (coupé), $B2D2$, $B2D2A1$ (coupé), $B2D2A2$, $B2D2A2C1$ (nouvelle solution, 210), $B2D2A2C2$ (coupé).

La stratégie de profondeur à droite : Racine, $B2$, $B2D2$, $B2D2A2$, $B2D2A2C2$ (nouvelle solution, 190), $B2D2A2C1$ (nouvelle solution, 210), $B2D2A1$ (coupé), $B2D1$ (coupé), $B1$, $B1D2$, $B1D2A2$ (non réalisable), $B1D2A1$ (coupé), $B1D1$, $B1D1A2$, $B1D1A2C2$ (coupé), $B1D1A2C1$ (non réalisable), $B1D1A1$ (coupé).

La stratégie du meilleur : Racine, $B1$, $B1D2$, $B1D2A2$ (non réalisable), $B1D1$, $B1D1A2$, $B1D1A2C1$ (non réalisable), $B2$, $B2D2$, $B2D2A2$, $B2D2A2C1$ (nouvelle solution, 210). Il coupe ensuite tout le reste. Ce comportement est dû au fait que, pour tout nœud de borne B, il existe au moins une solution réalisable de valeur B. L'algorithme va donc énumérer toutes les branches non réalisables, puis tomber sur la solution optimale. Ça n'arrive pas quand la borne est optimiste.

La stratégie du pire : Racine, $B2$, $B2D1$, $B2D1A1$, $B2D1A1C2$ (nouvelle solution, 140), $B2D1A1C1$ (nouvelle solution 160), ... Elle va énumérer beaucoup de solutions inutilement.

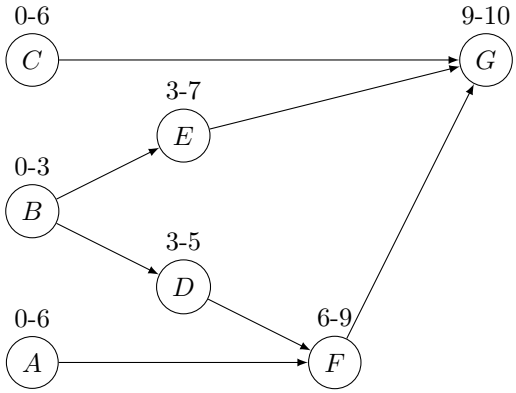
Exercice 2 — *Ordonnancement avec ressources*

On considère le problème d'ordonnancement où on suppose que le nombre d'employés est limité à 5. On a déjà rencontré ce problème dans un précédent TD. On souhaite savoir en combien de temps minimum on peut finir le projet tout en respectant cette contrainte de nombre d'employés.

tâche	durée	tâches précédentes	nombre d'employés
A	6	-	3
B	3	-	2
C	6	-	1
D	2	B	1
E	4	B	3
F	3	D A	3
G	1	F E C	2

1. Dessiner le graphe Potentiel Tache associé à ce projet. On suppose dans un premier temps qu'on ne tient pas compte des ressources. Combien de temps t dure le projet ? Pourquoi t est une borne inférieure de la solution optimale du problème avec ressource ?

► **Correction**



On indique sur chaque tâche le temps au plus tôt, suivi de la fin de l'exécution de la tâche si on la démarre au temps au plus tôt. Le projet a une durée de $t = 10$.

Il s'agit d'une borne inférieure car on a supprimé la contrainte du nombre d'employés. Puisqu'on est moins contraint, il y a plus de solutions possibles. Donc la solution optimale sans la contrainte est meilleure (i.e. a une durée plus faible) que la solution optimale avec contrainte.

Donc ici, 10 est bien une borne inférieure de la durée du projet avec la contrainte d'employés.

2. Trouver le premier instant où la contrainte du nombre d'employés est violée. Soient T les tâches en cause. Montrer que, dans une solution optimale, il existe au moins deux tâches t et t' de T telles que t s'effectue strictement avant t'.

► Correction

Il s'agit de l'instant 0, avec les tâches A, B et C.

Les tâches de T ne peuvent pas être toutes faites en même temps d'après la contrainte de nombre d'employé. Il y a donc nécessairement deux tâches t et t' qui ne se superposent pas. Cela est vrai pour toute solution réalisable, donc en particulier pour la solution optimale.

3. On va séparer le problème en sous-problèmes : pour chaque couple de tâches (t, t') de T, on crée un nouveau sous-problème où t doit être avant t' (en rajoutant une contrainte de précédence dans le tableau). On crée donc $|T| \cdot (|T| - 1)$ sous-problèmes. En utilisant cette séparation et la question 1, trouvez la solution optimale du problème d'ordonnancement.

► Correction

On obtient donc un arbre de branchement, chaque branchement correspond à une séparation. Si T est vide, alors la solution obtenue à la question 1 est optimale et respecte la contrainte du nombre d'employés. Ainsi, on est sur une feuille de l'arbre.

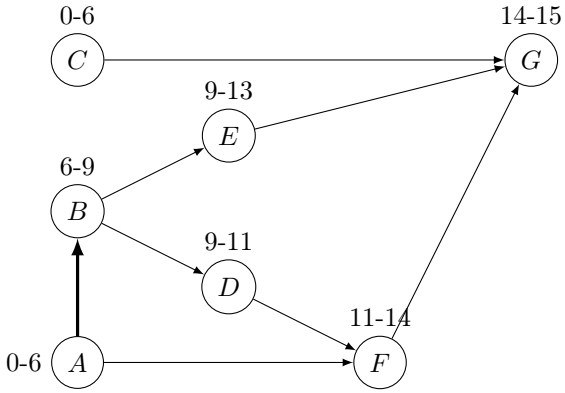
Si T est non vide, alors la question 1 nous donne une borne inférieure qu'on peut utiliser pour faire du branch and bound.

Voici le déroulé de l'algorithme sur l'instance de l'exercice.

— A la racine, on a pas encore fait de choix de précédence. On exécute donc exactement la question 1. On trouve $t = 10$ et $T = (A, B, C)$.

On a donc 6 choix : forcer A avant B, ou l'inverse, forcer B avant C, ou l'inverse, forcer A avant C, ou l'inverse. On a donc 6 fils. Observons les 6 fils les uns après les autres :

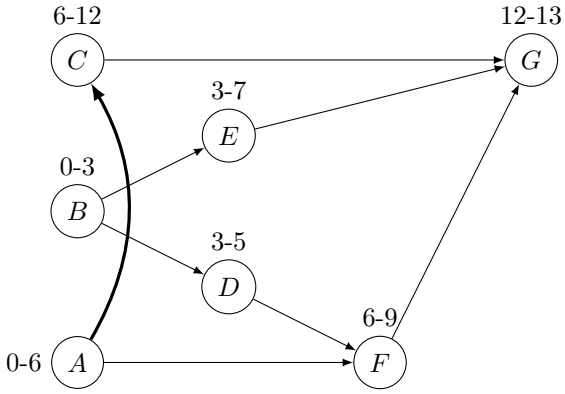
— A avant B. On obtient le diagramme suivant :



On obtient donc $t = 15$. On voit que E et F sont en conflit à l'instant 11. Donc $T = (E, F)$.

—

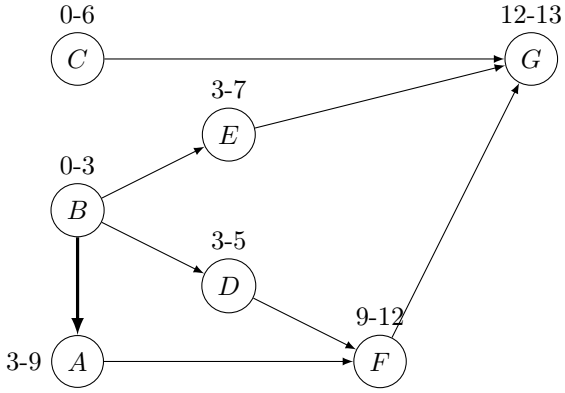
— A avant C. On obtient le diagramme suivant :



On obtient donc $t = 13$. On voit que A, D et E sont en conflit à l'instant 3. Donc $T = (A, D, E)$.

—

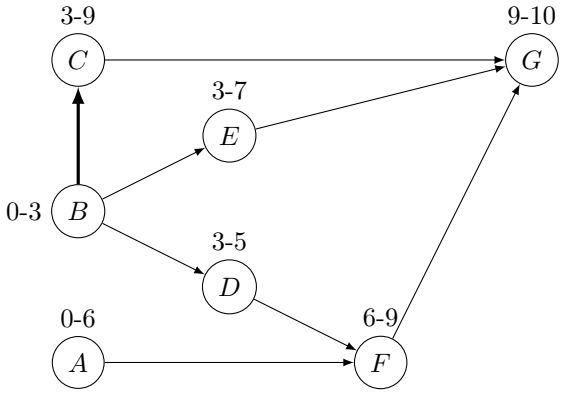
— B avant A. On obtient le diagramme suivant :



On obtient donc $t = 13$. On voit que A, C, D et E sont en conflit à l'instant 3. Donc $T = (A, C, D, E)$.

—

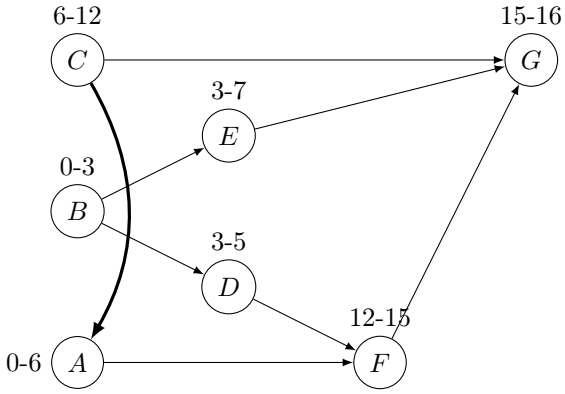
— B avant C. On obtient le diagramme suivant :



On obtient donc $t = 10$. On voit que A, C, D et E sont en conflit à l'instant 3. Donc $T = (A, C, D, E)$.

—

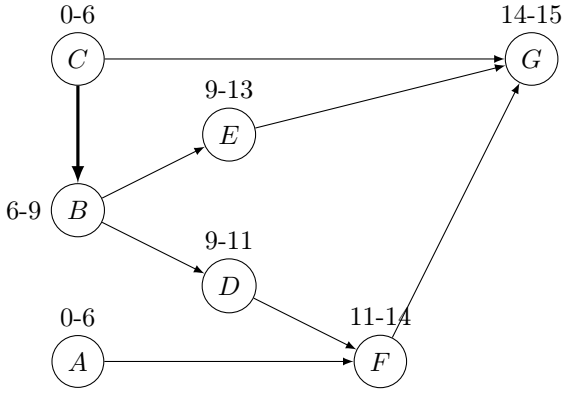
— C avant A. On obtient le diagramme suivant :



On obtient donc $t = 16$. On voit que C, D et E sont en conflit à l'instant 3. Donc $T = (C, D, E)$.

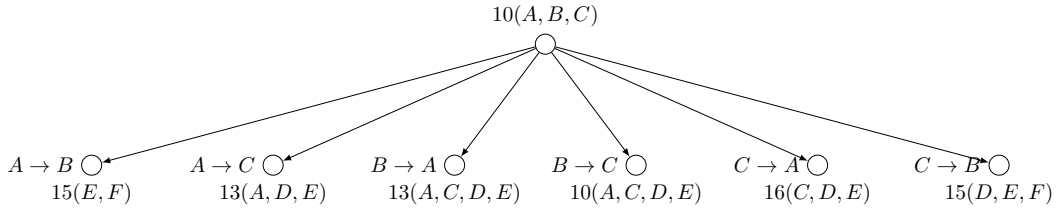
—

— C avant B. On obtient le diagramme suivant :



On obtient donc $t = 15$. On voit que D et E sont en conflit à l'instant 9. Donc $T = (D, E, F)$.

On obtient donc le début de l'arbre suivant :



On recommence ensuite, soit au nœud de gauche si on parcours en profondeur à gauche, soit au 4e fils si on effectue le meilleur d'abord.

Si une personne courageuse tente d'effectuer cet algorithme avec un parcours en profondeur à gauche, elle devra effectuer 112 explorations et 295 coupes suffiront. Si elle le fait avec la technique du meilleur d'abord, seulement 27 explorations suffiront.

Dans les deux cas, on trouve une solution optimale de valeur 14. Elle consiste à rajouter les précédentes suivantes : $(A \to E), (B \to C), (C \to F), (E \to F)$.

Exercice 3 — *Génétique tardive*

Deux étudiants en génétique font un projet d'étude pour réussir leur master. Ils ont effectué des expériences sur un groupe d'une dizaine de souris pour changer la couleur de leur pelage. Ayant commencé leur expérience la veille de la date limite pour rendre leur projet, ils doivent se dépêcher d'analyser leurs données pour rédiger leur rapport.

Ils décident donc de se répartir chacun la moitié des expériences. Malheureusement, il est difficile de tirer des conclusions de la moitié des expériences seulement. En effet, chaque souris a eu un contact plus ou moins long avec les autres, de sorte que chacune a eu un impact plus ou moins fort sur le changement de couleur du pelage des autres souris : les données sont corrélées.

Comment ces étudiants doivent-ils se répartir les souris de sorte que l'impact total des souris d'un groupe envers les souris de l'autre groupe soit minimal ? Modéliser le problème par un problème de graphe, puis le résoudre avec une procédure de Branch and Bound. Connaissez-vous un algorithme plus rapide ? Le tableau ci-après indique la durée du contact en minutes entre deux souris pendant l'expérience. Un trait signifie qu'il n'y a pas eu contact.

	A	B	C	D	E	F	G	H
A	-	1	12	-	-	1	-	7
B	1	-	8	-	-	-	-	-
C	12	8	-	2	-	-	-	-
D	-	-	2	-	6	-	5	-
E	-	-	-	6	-	4	-	10
F	1	-	-	-	4	-	5	-
G	-	-	-	5	-	5	-	3
H	7	-	-	-	10	-	3	-

► **Correction**

Le problème de graphe est le suivant : on a un graphe $G = (V, E)$ avec des poids sur les arêtes et où $|V|$ est pair, il faut partitionner V en deux ensembles V_1 et V_2 de même taille tels que la capacité de la coupe séparant V_1 et V_2 soit minimum, autrement dit, la somme des poids des arêtes reliant un noeud de V_1 à un noeud de V_2 doit être minimum. Chaque noeud représente une souris et chaque arête signifie qu'il y a eu contact entre les souris, le poids de l'arête est égal à la durée de ce contact.

Voici une méthode de B et B simple et efficace pour cet exemple :

Pour la séparation, on ordonne les noeuds (arbitrairement ou non), et à chaque étape, on choisit un noeud et on décide de le mettre dans V_1 ou dans V_2. A l'étape d'après, on prend la décision pour le noeud suivant dans l'ordre choisi, etc.

Pour l'évaluation : la séparation nous donne 3 catégories de noeuds : ceux pour lesquels on a pris la décision de les mettre dans V_1, ceux qui sont dans V_2 et les autre, pour lesquels aucune décision n'a encore été prise. Une borne inf consiste à calculer la somme des poids des arêtes reliant les noeuds actuellement dans V_1 aux noeuds actuellement dans V_2, et de ne pas tenir compte de la 3e catégorie de noeuds. Par exemple, si on choisit de mettre A dans V_1, et B et C dans V_2, et qu'on a rien décidé pour le reste, on sait que la coupe sera au moins de valeur 13.

Je donne l'arbre ci après. Concernant la question "connaissez vous mieux", certains élèves seraient tentés de parler de ford fulkerson. Il est bien sûr inutile pour ce problème car il ne garanti pas que la coupe partitionne V en deux ensemble de même taille.

Quelques remarques sur le dessin ci après :

— v signifie qu'on place le noeud v dans V_1 et $\bar{v}$ signifie qu'on le place dans V_2

— à gauche de chaque noeud, il y a la borne inf, à droite l'indice de l'ordre dans lequel les noeuds sont explorés

— en dessous des feuilles, il y a la solution réalisable correspondante et la valeur

— j'ai employé la stratégie suivante : trouver le plus vite possible une solution réalisable, puis toujours explorer le noeud dont la borne inf est maximum, afin de maximiser les chances de couper.

— je n'ai pas exploré $\bar{A}$ car c'est symétrique avec A

— quand 4 nœuds sont dans V_1 ou V_2, je ne vais pas plus loin car une solution réalisable est définie