Tutorial 4 : Branch and bound

Operations research, 3rd semester.

2024

Exercice 1 — *Constrainted selection*

The following table gives you the costs and utilities of Counter Strike weapons and accessories. The costs are in dollars and the utilities in points. You have 4200$. We want to maximize the utility of the equipment and satisfy the budget constraint. In order to simplify, we add the following constraint : in each column, you have to choose one (and only one) object.

	A : Pistol	B : Rifle	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-Helmet 1000$ 80pt

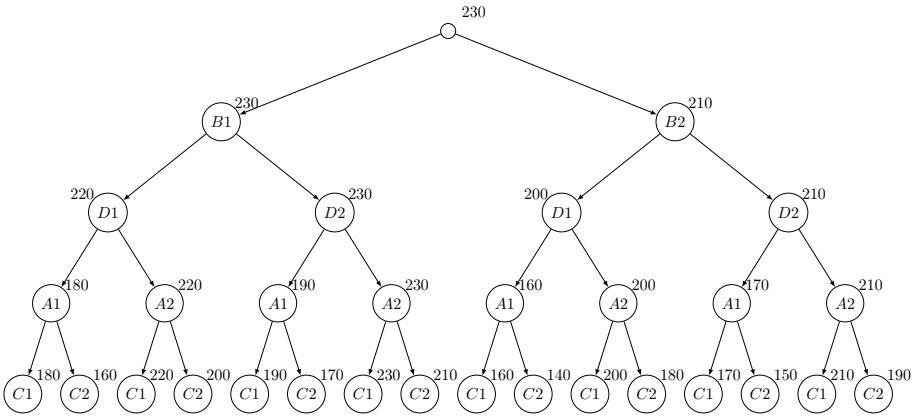
We want to solve the problem with a branch and bound method. At each iteration, we make a choice for a category. Each node of the following tree gives you the column and the line of the corresponding choice. For example, B1D2A1C1 corresponds to AK-47, Kevlar-Helmet, Bereta, Explosive grenade. The subtree of B2D2 is the set of solutions containing the FAMAS and the Kevlar-Helmet. We wrote an upper bound near each node : the maximum utility that can be reached considering the choices that were already made.

The formula of that bound is

— for each category in which a choice is already made, we add the utility of the chosen object ;

— for each other category, we add the maximum utility among all the objects of that category.

For example, for the root, no choice is made. The maximum utility we can achieve is 50+60+40+80 with the Desert Eagle, the AK-47, the Explosive grenades and the Kevlar-Helmet. For B2D2 we choose the FAMAS and the Kevlar-Helmet, of utility 40+80, and then, we add 50, the utility of the Desert Eagle, and 40, the utility of the Explosive grenades, (for the best pistol and the best grenade, according to the table). We then have 210.



1. How many nodes of the tree would explore an algorithm that does not take into account the upper bounds ? (It does not explore a node if the price is already larger than 4200$).
2. If you use the algorithm of the course, which use the depth first search strategy (the left child first, then the right child), in which order would the nodes be explored and how many nodes would you explore ?

3. If you use the depth first search strategy with the right child first, in which order would the nodes be explored and how many nodes would you explore ?
4. If you use the best first strategy (the node with the highest upper bound first), in which order would the nodes be explored and how many nodes would you explore ? Could you explain what happens ? Does it always happen when we use this strategy ?
5. If you use the worst first strategy (the node with the lowest upper bound first), in which order would the nodes be explored and how many nodes would you explore ?

▶ **Correction**

Note : I refer to a node by its name and the names of all its ancestors.

For the first question, we calculate the budget of all partial solutions and only keep the solutions with a cost of 4200 or less. Therefore, it would not explore the nodes $B1D1A2C1$, $B1D2A1C1$, $B1D2A1C2$, and $B1D2A2$.

The left depth-first strategy explores in this order :

Root, $B1$, $B1D1$, $B1D1A1$, $B1D1A1C1$ (new solution, 180), $B1D1A1C2$ (cut), $B1D1A2$, $B1D1A2C2$ (not feasible), $B1D1A2C2$ (new solution, 200), $B1D2$, $B1D2A1$ (cut), $B1D2A2$ (not feasible), $B2$, $B2D1$ (cut), $B2D2$, $B2D2A1$ (cut), $B2D2A2$, $B2D2A2C1$ (new solution, 210), $B2D2A2C2$ (cut).

The right depth-first strategy : Root, $B2$, $B2D2$, $B2D2A2$, $B2D2A2C2$ (new solution, 190), $B2D2A2C1$ (new solution, 210), $B2D2A1$ (cut), $B2D1$ (cut), $B1$, $B1D2$, $B1D2A2$ (not feasible), $B1D2A1$ (cut), $B1D1$, $B1D1A2$, $B1D1A2C2$ (cut), $B1D1A2C1$ (not feasible), $B1D1A1$ (cut).

The best strategy : Root, $B1$, $B1D2$, $B1D2A2$ (not feasible), $B1D1$, $B1D1A2$, $B1D1A2C1$ (not feasible), $B2$, $B2D2$, $B2D2A2$, $B2D2A2C1$ (new solution, 210). It then cuts all the rest. This behavior is due to the fact that, for any node with a bound B, there exists at least one feasible solution with a value of B. Therefore, the algorithm will enumerate all infeasible branches, then find the optimal solution. This does not happen when the bound is optimistic.

The worst strategy : Root, $B2$, $B2D1$, $B2D1A1$, $B2D1A1C2$ (new solution, 140), $B2D1A1C1$ (new solution 160), ... It will enumerate many solutions unnecessarily.

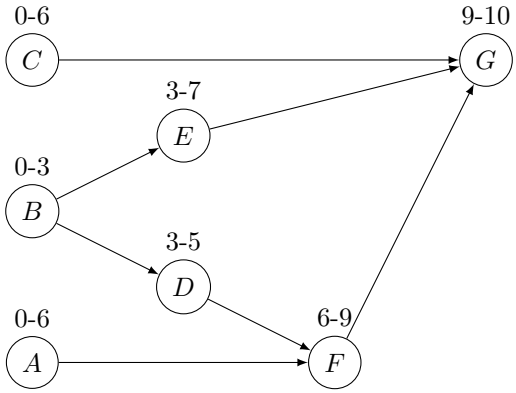
Exercice 2 — *Project planning with ressources*

We consider the following project planning problem where the number of employees is 5. The same problem was already given in a previous tutorial. We want to know how much time we need to finish the projet while satisfying the comstraint with the number of employees.

task	duration	previous tasks	nb employees
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. Draw the potential-metra graph of that project. If we do not consider the resources constraint, what is the minimum duration t of the project ? Why t is a lower bound of the optimal solution of the problem where we satisfy the resources constraint ?

▶ **Correction**



Each task indicates the earliest start time, followed by the end of the task execution if it is started at the earliest time. The project has a duration of $t = 10$.

This is a lower bound because we have removed the constraint on the number of employees. Since we are less constrained, there are more possible solutions. Therefore, the optimal solution without the constraint is better (i.e., has a shorter duration) than the optimal solution with the constraint.

Thus, here, 10 is indeed a lower bound for the duration of the project with the employee constraint.

2. What is the first time where the capacity constraint is violated ? Let T be the jobs that violate the constraint. Show that, in an optimal solution, there are at least two tasks t and t' of T such that t is before t'.

▶ **Correction**

This is the moment 0, with tasks A, B, and C.

The tasks of T cannot all be performed at the same time due to the employee constraint. Therefore, there must necessarily be two tasks t and t' that do not overlap. This is true for any feasible solution, and in particular for the optimal solution.

3. In order to branch the problem, we can do that way : for each couple of jobs (t, t') of T, we define a new sub-problem where t must be before t' (by adding a precedence constraint in the table). We then build $|T| \cdot (|T| - 1)$ sub-problems. Using that branching algorithm and the lower bound of question 1, could you find an optimal solution of the example of the exercice ?

▶ **Correction**

We obtain a branching tree, where each branch corresponds to a separation. If T is empty, then the solution obtained from question 1 is optimal and respects the constraint on the number of employees. Thus, we are at a leaf of the tree.

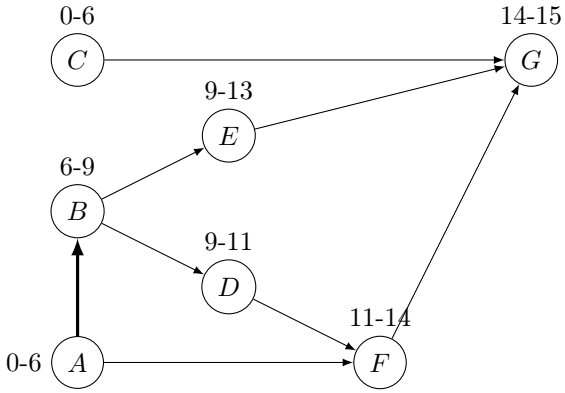
If T is non-empty, then question 1 gives us a lower bound that we can use for branch and bound.

Here is the outline of the algorithm on the instance from the exercise.

— At the root, we haven't made any precedence choices yet. We will therefore execute question 1 exactly. We find $t = 10$ and $T = (A, B, C)$.

So we have 6 choices : force A before B, or the reverse, force B before C, or the reverse, force A before C, or the reverse. Thus, we have 6 branches. Let's examine the 6 branches one after the other :

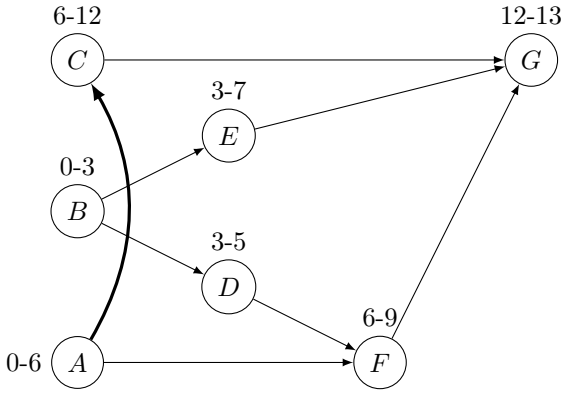
— A before B, we get the following diagram



We obtain $t = 15$. We see that E and F are in conflict at time 11. Therefore, $T = (E, F)$.

—

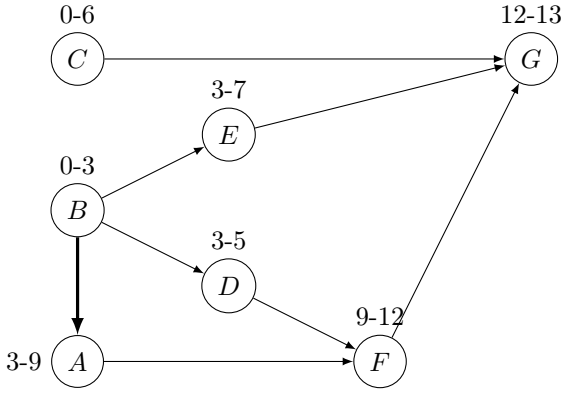
— A before C, we get the following diagram



We obtain $t = 13$. We see that A, D and E are in conflict at time 3. Therefore, $T = (A, D, E)$.

—

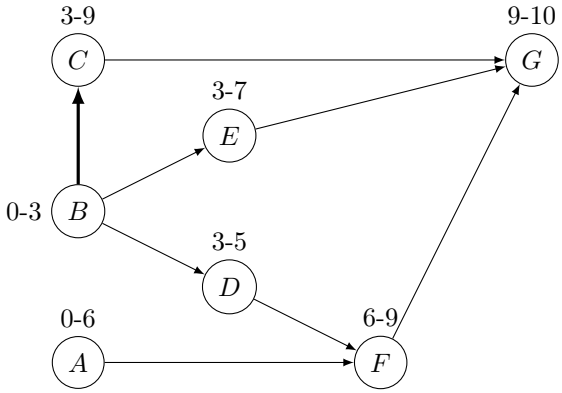
— B before A, we get the following diagram



We obtain $t = 13$. We see that A, C, D, and E are in conflict at time 3. Thus, $T = (A, C, D, E)$.

—

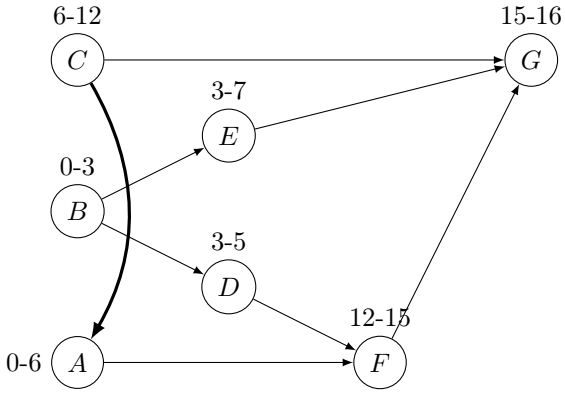
— B before C, we get the following diagram



We obtain $t = 10$. We see that A, C, D, and E are in conflict at time 3. Thus, $T = (A, C, D, E)$.

—

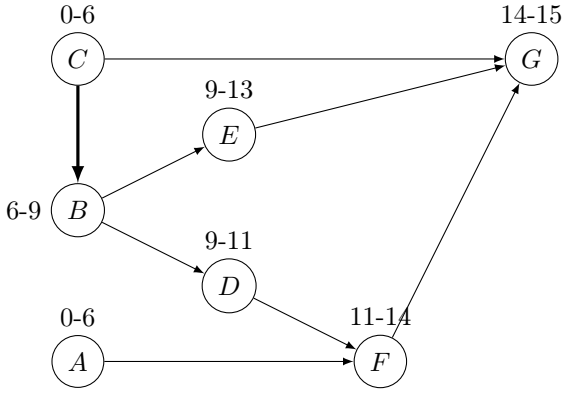
— C before A, we get the following diagram



We obtain $t = 16$. We see that C, D, and E are in conflict at time 3. Therefore, $T = (C, D, E)$.

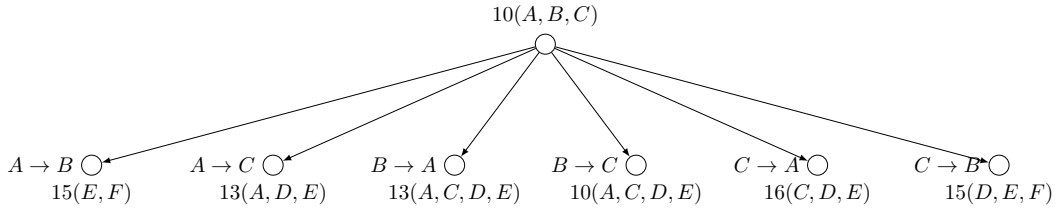
—

— C before B, we get the following diagram



We obtain $t = 15$. We see that D and E are in conflict at time 9. Therefore, $T = (D, E, F)$.

We get the following beginning of the tree :



We then start again, either at the left node if we are traversing depth-first to the left, or at the 4th child if we are performing best first.

If a brave person attempts to carry out this algorithm with a depth-first left traversal, they will need to perform 112 explorations and 295 cuts will be sufficient. If they do it with the best-first technique, only 27 explorations will be necessary.

In both cases, an optimal solution with a value of 14 is found. It consists of adding the following previous ones : $(A \to E), (B \to C), (C \to F), (E \to F)$.

Exercice 3 — *Late genetic*

Two students in the field of genetic are finishing a project in order to get their master diploma. They did experiments on ten mouses in which the color of the fur changes. Because they started the experiments the day before the deadline, they need to analyse the data as fast as possible so that they can finish the report on time.

Consequently, they decided to divide the analysis into two equal parts. However, it is hard to conclude anything from only half of the experiments. Indeed, each mouse had a more or less long contact with the other mouses, so that the way the color of the fur changed on one mouse can impact the color of the fur of the other mouses : the data are correlated.

How does those students must part the mouses so that the total impact of the mouses of one group to the mouses on the other group is minimized ? Model this problem by a graph problem and solve it using branch and bound. Do you know a better algorithm ?

The following table gives, for each couple of mouses, the contact duration in minutes between those two mouses. A line means that there were no significant 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**

The graph problem is as follows : we have a graph $G = (V, E)$ with weights on the edges and where $|V|$ is even. We need to partition V into two sets V_1 and V_2 of equal size such that the capacity of the cut separating V_1 and V_2 is minimized ; in other words, the sum of the weights of the edges connecting a node from V_1 to a node from V_2 must be minimized. Each node represents a mouse, and each edge indicates that there has been contact between the mice, with the weight of the edge equal to the duration of that contact.

Here is a simple and efficient Branch and Bound method for this example :

For the separation, we order the nodes (arbitrarily or not), and at each step, we choose a node and decide to place it in V_1 or in V_2. In the next step, we make the decision for the next node in the chosen order, and so on.

For the evaluation : the separation gives us 3 categories of nodes : those for which we have decided to place them in V_1, those that are in V_2, and the others, for which no decision has yet been made. A lower bound can be established by calculating the sum of the weights of the edges connecting the nodes currently in V_1 to the nodes currently in V_2, and ignoring the third category

of nodes. For example, if we choose to put A in V_1, and B and C in V_2, and we have made no decisions about the rest, we know that the cut will be at least valued at 13.

I provide the tree below. Regarding the question "do you know better," some students might be tempted to talk about the Ford-Fulkerson method. It is, of course, unnecessary for this problem because it does not guarantee that the cut partitions V into two sets of equal size.

A few remarks on the drawing below :

— v means that we place node v in V_1 and $\bar{v}$ means that we place it in V_2

— to the left of each node, there is the lower bound ; to the right, the index of the order in which the nodes are explored

— below the leaves, there is the corresponding feasible solution and its value

— I employed the following strategy : find a feasible solution as quickly as possible, then always explore the node whose lower bound is maximum, in order to maximize the chances of cutting.

— I did not explore $\bar{A}$ because it is symmetric with A

— when 4 nodes are in V_1 or V_2, I will not go further because a feasible solution is defined