Branch and Bound - Approximation - Exercices

Advanced Algorithms

Master CPS2/DSC/MLDM

1 Task assignment

We consider the task assignment problem seen during the lecture. Recap: we consider the following task assignment problem: we have n agents and n tasks and the problem is to associate a task to each agent in order to minimize the global cost. Indeed, each agent performs differently on the different tasks, and we assume to have a table c indicating for each agent the cost associated to each task. An example of a cost table is provided in Table 1.

Agent\Task	1	2	3	4
1	5	4	13	8
2	6	1	7	11
3	8	6	8	7
4	9	4	6	11

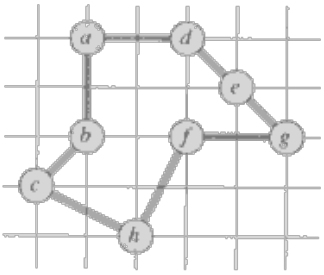
Table 1: Example of a cost table. The value 6 at position $c[2, 1]$ indicates that for agent number 2 the cost of achieving task 1 is 6.

1. Can you propose an improvement of the h function seen in class by defining a lower bound with respect to each remaining agent?
2. Consider now a new strategy where we consider the minimum value existing among the remaining values available in the cost matrix, we take it (ie we associate the corresponding task to the agent) and we remove all the values located in the same line and in the same column. Show with a simple example that this strategy cannot lead to an optimal solution.

2 Approximation TSP

Let us consider a weighted (undirected) graph $G = (V, E)$ such that the weights respect **the triangle inequality** modeling distances between cities for the traveling salesman problem. We want to solve the TSP problem corresponding to find an Hamiltonian path in G of minimum cost (*ie* a cycle containing all the nodes exactly one, the cost of the cycle (tour) being the sum of all the weights of the edges along the cycle). We want to define a solution computable in polynomial time and we do not mind if we are not able to output the best solution, we just want a good approximation.

Optimal TSP tour for a given problem (graph) would be



Try to propose a solution based first on the extraction of a particular tree in the graph. Can you justify that the cost of the proposed solution is at most twice the cost of the best solution?

3 Branch and Bound - TSP

We consider again the TSP problem, instead of building the search tree with respect to graph nodes, we consider edges of the graph. When we build the (binary) search tree, at each step and for an edge a: the left child contains solutions with a and the right son solutions without a.

1. **Using an adjacency matrix**. As a first example, we consider a simple problem with 3 cities, the cost of a using an edge from a city i to a city j is given by the following matrix M:

	A	B	C
A	∞	4	2
B	6	∞	5
C	2	6	∞

   Let $C(M)$ the optimal cost of a tour in the graph represented by the matrix M. Show that an optimal solution has at most one element per row/column

2. Show that if we substract a value from a whole row/column (all the values remain positive), we do not change the solution.

3. With this property, we can normalize the matrix and compute a value for the function f. We define
$$\forall i \forall j \ (\mathcal{M}_l(A))[i,j] = A[i,j] - min_{1 \leq k \leq n} A[i,k]$$
   and
$$M_{row}(A) = \sum_{1 \leq i \leq n} min_{1 \leq k \leq n} A[i,k]$$
   Then,
$$C(\mathcal{M}_l(A)) + M_{row}(A) = C(A).$$
   We do the same analysis for the columns. Now we can combine the two operators
$$C(\mathcal{M}_c(\mathcal{M}_l(A))) + M_{column}(\mathcal{M}_l(A)) = C(\mathcal{M}_l(A))$$
   and thus,
$$C(\mathcal{M}_c(\mathcal{M}_l(A))) + M_{column}(\mathcal{M}_l(A)) + M_{row}(A) = C(A).$$
   If $\mathcal{M} = \mathcal{M}_c \circ \mathcal{M}_l$, we can write
$$C(\mathcal{M}(A)) + M_{column}(\mathcal{M}_l(A)) + M_{row}(A) = C(A)$$

since $C(\mathcal{M}(A)) \geq 0$, we have

$$C(A) \geq M_{column}(\mathcal{M}_l(A)) + M_{row}(A).$$

$M_{column}(\mathcal{M}_l(A)) + M_{row}(A)$ defines a value that can be used for reducing the search in the right subtree. We define

$$f(A) = M_{column}(\mathcal{M}_l(A)) + M_{row}(A).$$

For example, compute for the matrix M above $\mathcal{M}_l(M)$, then $M_{row}(M)$ and $M_{column}(\mathcal{M}_l(M))$.

4. We can now define the search tree: the children of a node are built with respect to an edge $[l, c]$. The right son is obtain by putting ∞ in the entry $[l, c]$

$$\begin{array}{lll} (right(A))[i,j] = & \infty & \text{for } i = l \text{ and } j = c \\ (right(A))[i,j] = & A[i,j] & \text{otherwise.} \end{array}$$

For the left son, choosing an edge $[l, c]$ implies to take another edge on the same row or the same column:

$$\begin{array}{lll} (left(A))[i,j] = & \infty & \text{for } i = l \text{ or } j = c \\ (left(A))[c,l] = & \infty & \\ (left(A))[i,j] = & A[i,j] & \text{otherwise.} \end{array}$$

What is the best edge to select at each step? Illustrate it with the example.

5. Try on a more complex example with 7 cities, the cost of a using an edge from a city i to a city j is given by this new matrix M.

	1	2	3	4	5	6	7
1	∞	3	93	13	33	9	57
2	4	∞	77	42	21	16	34
3	45	17	∞	36	16	28	25
4	39	90	80	∞	56	7	91
5	28	46	88	33	∞	25	57
6	3	88	18	46	92	∞	7
7	44	26	33	27	84	39	∞

How can build the tree to potentially accelerate the search of the optimal solution?