Algorithmic Methods for Mathematical Models
Lab Session 5 - GRASP Metaheuristics

Arnau Abella
Universitat Politècnica de Catalunya

December 2, 2020

- Prepare a pseudocode for the GRASP constructive algorithm. Specify the greedy function and the RCL.

```
Algorithm 1: GRASP - Constructive Algorithm
Input:
  • A set $T$ of tasks, each task $t$ requires $r_t$ resources
  • A set $C$ of computers, each computer $c$ has a capacity of $r_c$ resources
  • The RCL threshold parameter $\alpha \in [0, 1]$
Output: A set $w$ of assignments, each assignment $\langle t, c\rangle$ associates a
        task $t$ with a computer $c$ $s.t.$ Each task $t \in T$ appears exactly
        once
$w \leftarrow \emptyset$
forall $t \in T$ do
    $Q \leftarrow \{\langle c, q(\langle t, c\rangle, w)\rangle \mid c \in C, q(\langle t, c\rangle, w) \neq \infty\}$
    if $|Q| = 0$ then
        return $INFEASIBLE$
    end

    $q^{min} \leftarrow \min_q Q$
    $q^{max} \leftarrow \max_q Q$
    $RCL \leftarrow \{c \in Q \mid q(\langle t, c\rangle, w) \leq q^{min} + \alpha(q^{max} - q^{min})\}$
    $c_{sel} \leftarrow u.a.r$ select $c \in RCL$
    $w \leftarrow w \cup \{\langle t, c_{sel}\rangle\}$
end
return $w$
```

$$q(\langle t, c\rangle, w) = \begin{cases} \infty & if r_t > r_c - \sum_{t' \in w_c} r_{t'} \\ \frac{r_t + \sum_{t' \in w_c} r_{t'}}{r_c} & otherwise \end{cases}$$

where $w_c = \{t_i, \ldots, t_j\} \subseteq T$ are the tasks assigned to computer c in the partial solution w.

- Tune parameter α. Generate at least 2 new random instances of medium size and run the constructive phase of GRASP for different values of a from 0-1 in steps of 0.1. For every value, run the algorithm at least three times and compute the average of the cost of the obtained solutions. Prepare plots with the obtained values. Find the best value of α and use it for the rest of experiments.

  The solution's quality in terms of $\alpha \in [0, 1]$ has a v-shape with best value at $\alpha = 0.5$.

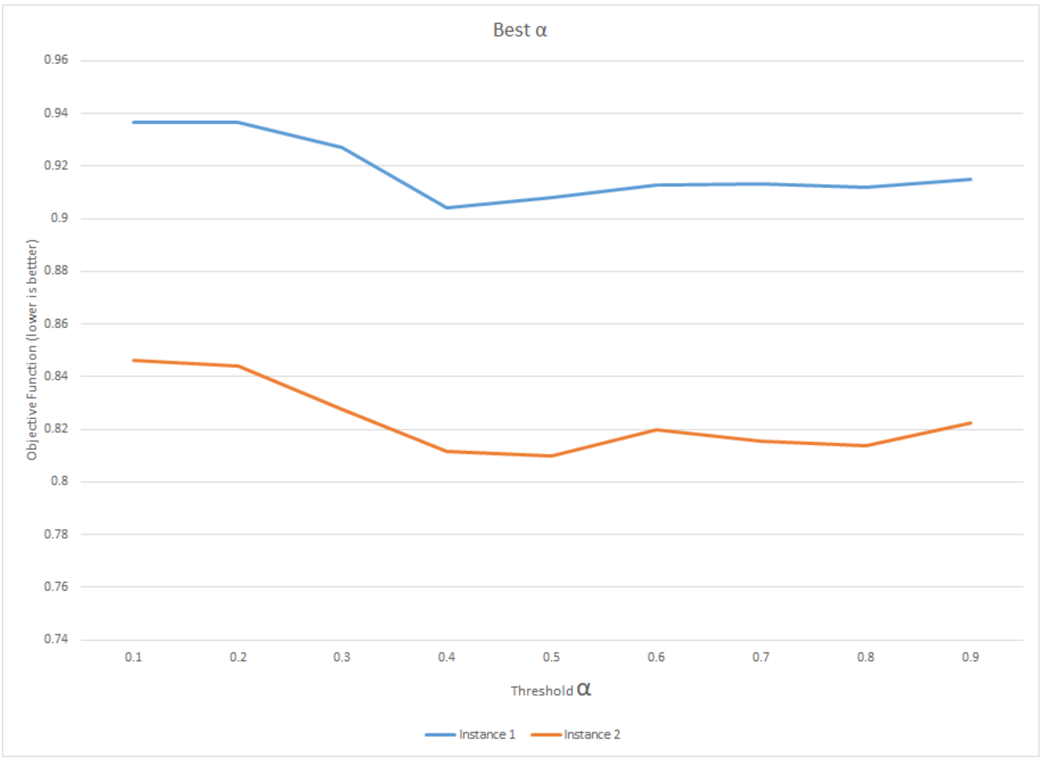

Figure 1: Objective value in terms of threshold α

- Solve the same instances that you generated in the last lab session using:
  - Constructive phase only.
  - Constructive + Local search (do for all combinations)

  Configure the heuristics to stop after 10min and plot the quality of the solutions, the number of iterations performed, and the time to solve against the size of the instances.

The quality of the solution, in average, is better in the *Constructive + LS* algorithm (see figure 2). It is important to notice that although you are doing less iterations i.e. narrower searches, the searches are more exhaustive in the neighbor of the solution (see table 1).

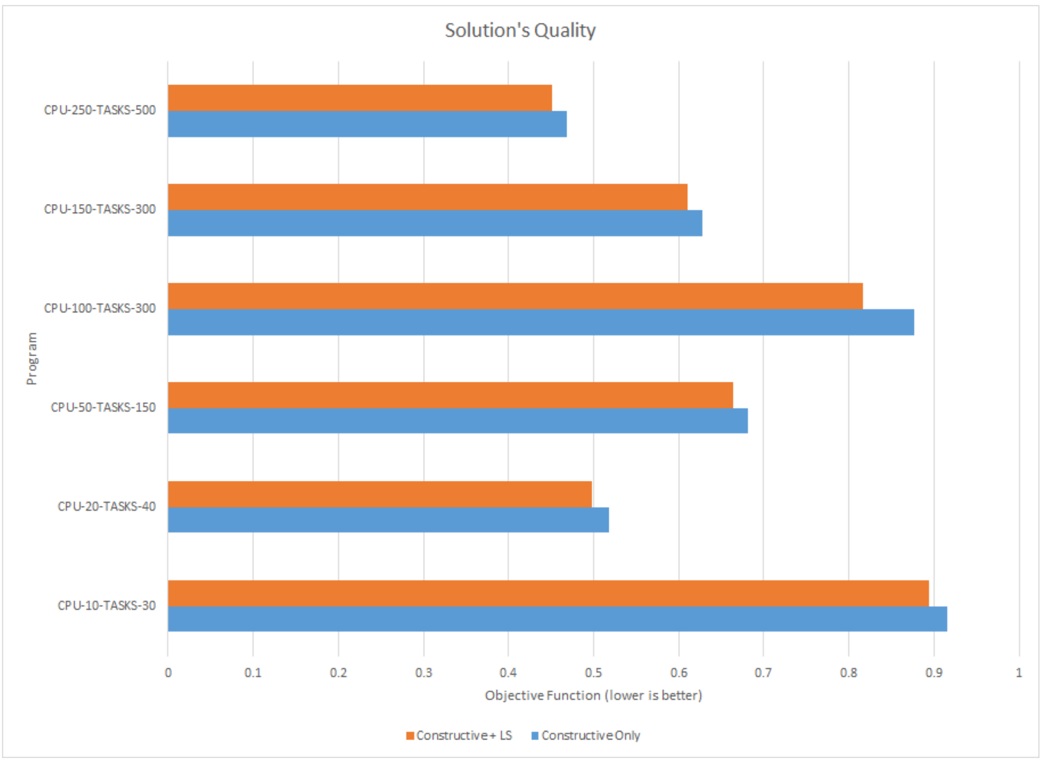

Figure 2: Constructive vs. Constructive + LS

Program	Iterations	
	Constructive	Constructive + LS
CPU-10-TASKS-30	21313	4335
CPU-20-TASKS-40	4024	370
CPU-50-TASKS-150	227	10
CPU-100-TASKS-300	28	3
CPU-150-TASKS-300	47	2
CPU-250-TASKS-500	3	2

Table 1: Iterations: Constructive vs. Constructive + LS

- Compare the results with those obtained in the previous lab session using greedy + LS and CPLEX.

  In this particular problem, *CPLEX* always performs better, in terms of execution time and solutions' quality, than the *heuristic algorithms*. Among the heuristic algorithms, *GRASP* performs better in term of solutions' quality than the *greedy algorithm*.

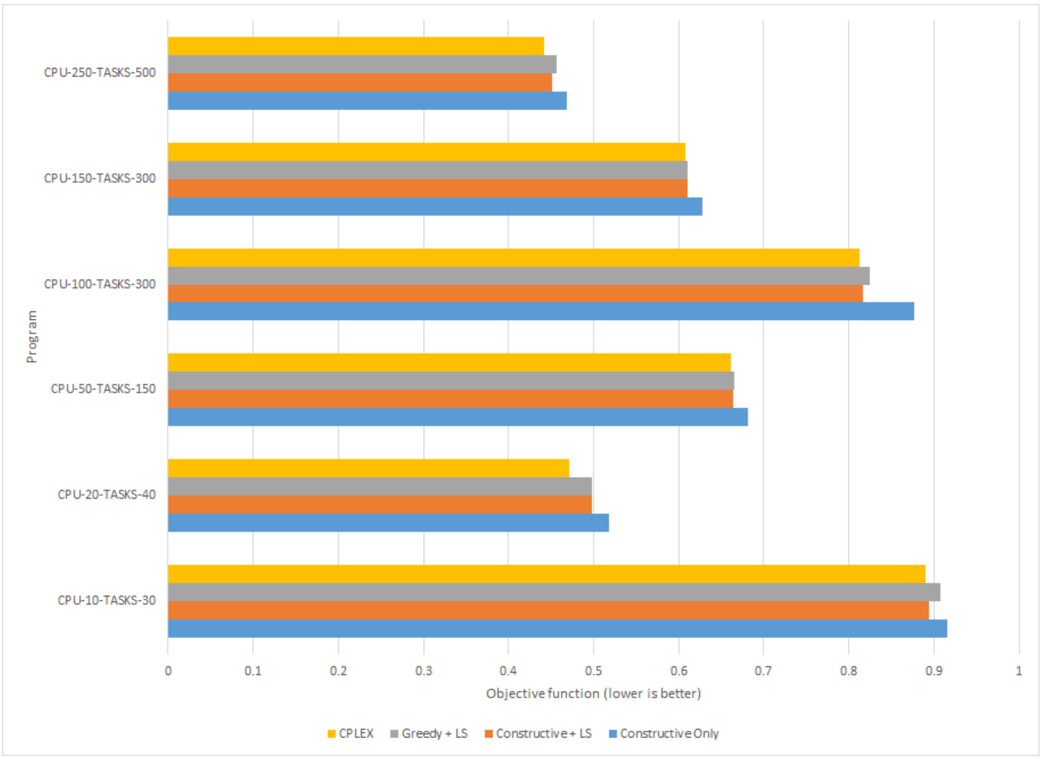

Figure 3: Comparison between Constructive, Constructive + LS, Greedy + LS and ILP (CPLEX)