Algorithmic Methods for Mathematical Models
Lab 4 — Greedy + Local Search Heuristics

Marcel Cases Freixenet

April 30, 2021

Tasks and results

a) Prepare a pseudocode for the Greedy algorithm. Specify the greedy function.

Algorithm 1 Greedy algorithm

Input: set of tasks and CPUs
Output: a solution for task assignment
$w \leftarrow \emptyset$
forall $t \in T$ **do**
  $c^{min} \leftarrow q(t, w)$
  if $q(c^{\min}) = \infty$ **then**
    return *infeasible*
  end
  $w \leftarrow w \cup \{\langle t, c^{min} \rangle\}$
end
return w

$$q(t, w) = \min\{q(\langle t, c\rangle, w) \mid c \in C\}$$

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

b) Prepare a pseudocode for the Local search algorithm. What neighborhoods and exploration strategies are implemented?

Algorithm 2 Local Search algorithm

Input: set of tasks and CPUs
Output: a set of solutions for task assignment

$w' \leftarrow w$
while $t < T_{lim}$ **do**
  $C \leftarrow sort(w', DESC)$
  for $c \in C$ **do**
    foreach $t \in w'_c$ **do**
      for $c' \in Cs.t.c' \leq c$ **do**
        foreach $t' \in w'_c$ **do**
          $rc = r_c - \sum_{t'' \in w_c} r_{t''}$
          $rc' = r_{c'} - \sum_{t'' \in w_{c'}} r_{t''}$
          if $r_{t'} - r_t \leq rc \wedge r_t - r_{t'} \leq rc'$ **then**
            $rc_{new} = rc + r_t - r_{t'}$
            $rc'_{new} = rc' + r_{t'} - r_t$
            improvement $= argmin(rc_{new}, rc'_{new})$
          end
        end
      end
    end
  end
end
$w' \leftarrow w' \cup (w'_c | t \cup t')$
$w' \leftarrow w' \cup (w'_{c'} | t' \cup t)$
return w'

This Local Search algorithm uses a *swapping* (task swapping) method with a *best improvement* search strategy.

c) Generate instances of increasing size. Store these instances as they will be used in the coming lab sessions.

The following instances have been generated with parameters:

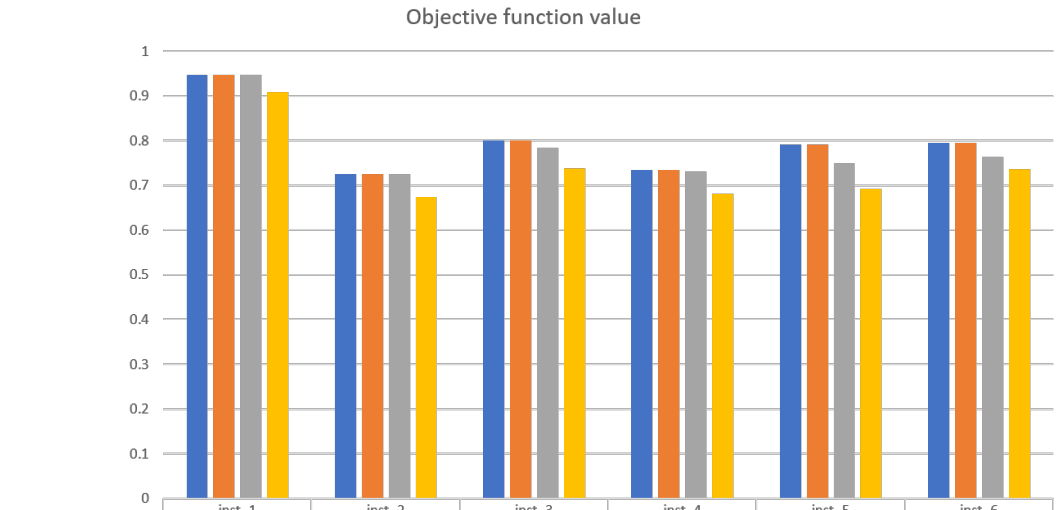
#	nCPU	nTasks
inst_1	4	7
inst_2	10	15
inst_3	15	25
inst_4	20	30
inst_5	25	35
inst_6	30	40

All of them have feasible solutions.

d) Solve the instances previously generated using:

- Random only
- Greedy function only
- Greedy + Local search (do for all combinations)

Plot the quality of the solutions and time to solve. Select the best combination.



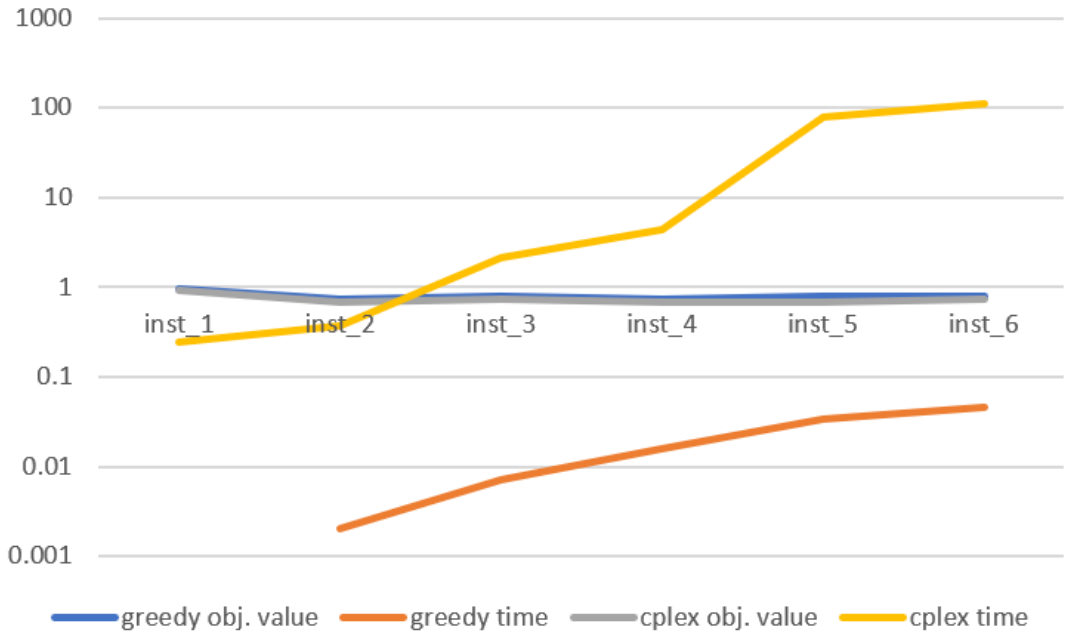
	inst_1	inst_2	inst_3	inst_4	inst_5	inst_6
Random	0.94661414	0.72540458	0.80060071	0.7341187	0.79188026	0.79516274
Greedy	0.94661414	0.72540458	0.80060071	0.7341187	0.79188026	0.79516274
Greedy + Local Search	0.94661414	0.72540458	0.78368074	0.73156501	0.74929824	0.76464096
CPLEX	0.9086282	0.6741311	0.7386987	0.6821929	0.6915781	0.736838

	inst_1	inst_2	inst_3	inst_4	inst_5	inst_6
Random	0	0.00302005	0.00584555	0.0150032	0.02599216	0.04398799
Greedy	0	0.0019989	0.0079987	0.01600099	0.03499985	0.04559016
Greedy + Local Search	0.00100017	0.00302553	0.02198529	0.02136993	0.06609797	0.10946345
CPLEX	0.243	0.369	2.091	4.361	79.675	112.34

e) Solve the instances previously generated using the ILP from lab session 2. Configure CPLEX to stop after 30min or GAP$\leq 1\%$.

The two figures above also contain the solutions obtained using CPLEX with $GAP = 0.01$ together with the computing time.

f) Plot the best combination for the Greedy and the ILP in terms of quality of the solutions and time to solve.



Conclusions

Random and Greedy offer almost the same solution for each instance with virtually no computing time. Greedy + Local Search takes a little more but provides a solution closer to the one obtained with CPLEX. The latter takes the highest time but seems to provide better solutions. According to the definition of *heuristics* —getting good solutions with a good time, maybe not the best—, all these results make sense.

References

- Class presentations & labs