Assignment MH

Guillem Cabo
Adrián Munera Sánchez

December 1, 2018

1 Greedy algorithm

S ← {}
for each e in E **do** incompatibles[e] $\leftarrow \sum_{b \in E} = I[e][b]$
for h in $1 \leq h \leq 72$ **do**
  $demand_h = c_t$
  while $demand_h > 0$ **do**
    $Set_h \leftarrow \{\}$
    for each e in E **do**
      if $(< e, h > \notin S$ **and** $\sum_{b \in S} = I[e][b] = 0$ **and** (h mod= 1 **or** $(< e, h-1 > \notin S$ **and** $< e, h-2 > \notin S)))$
      then $Set_h \leftarrow Set_h \cup e$
    if $\mid Set_h \mid = 0$ **then return** INFEASIBLE
    $e_{best} \leftarrow$**argmin** $\{q(< e, h > \mid e \in Set_h)\}$
    $S \leftarrow S \cup < e_{best}, h >$
    $demand_h \leftarrow demand_h - 60/pt$
return S

Greedy Function :

$$q < e, h >= \mathbf{min} \begin{cases} 0 & \text{if } e \in S \\ incompatibles[e] + 1 & \text{if } e \notin S \end{cases}$$

The greedy function selects the minimum cost for using each available employee (note that an employee is available if the hours constraints are satisfied, he is not already assigned at that hour and there is no incompatible employee assigned). The greedy function returns a 0 if the employee was already assigned before to the solution before, this is the best case so we don't have to add another employee. In the second case the greedy function returns the number of incompatibilities plus one (to avoid conflicts with 0), this is because its better to assign an employee with lower number of incompatibilities for the immediate future.

2 GRASP

S $\leftarrow$ {}
for each e in E **do** incompatibles[e] $\leftarrow \sum_{b \in E} = I[e][b]$
for h in $1 \leq h \leq 72$ **do**
  $demand_h = c_t$
  while $demand_h > 0$ **do**
    $Set_h \leftarrow \{\}$
    for each e in E **do**
      if $(< e, h > \notin S$ **and** $\sum_{b \in S} = I[e][b] = 0$ **and** (h mod= 1 **or**
        $(< e, h-1 > \notin S$ **and** $< e, h-2 > \notin S)))$
      then $Set_h \leftarrow Set_h \cup e$
    if $\mid Set_h \mid = 0$ **then return** INFEASIBLE
    $e_{min} \leftarrow$**argmin** $\{q(< e, h > \mid e \in Set_h)\}$
    $e_{max} \leftarrow$**argmax** $\{q(< e, h > \mid e \in Set_h)\}$
    $RCL_{min} \leftarrow \{q(< e, h >) \leq e_{min} + \alpha(e_{max} - e_{min}) \mid e \in Set_h\}$
    $e_{random} \leftarrow$**random** $e \in RCL_{min}$
    $S \leftarrow S \cup < e_{random}, h >$
    $demand_h \leftarrow demand_h - 60/pt$
return S

Greedy Function :

$$q < e, h >= \mathbf{min} \begin{cases} 0 & \text{if } e \in S \\ incompatibles[e] + 1 & \text{if } e \notin S \end{cases}$$

2.1 GRASP RCL construction

For the hour 4, c_t is 60, so we need at least 5 employees. From the full set of available employees, 11 and 15 are discarded, as they dont satisfy the hours constraint. The value of $q(\cdot)$ for the previous assigned employees is 0, as they are already part of the solution $5, 6, 16, 18$, and for the rest of employees the value will be the number of incompatibilities of each one (incompatibilities[e]).

So the e_{min} and e_{max} will be 0 and 5 respectively (4 is the maximum number of incompatibilities). RCL is computed as following:

$$RCL_{min} \quad q(< e, h >) \leq 0 + 0.3(5 - 0) \mid e \in Set_h$$

And employees with a $q(\cdot) \leq 1.5$ will be selected:
$RCL_{min} \quad q(< e, h >) = 5, 6, 16, 18, 22, 25, 27, 29$

Where $q(\cdot)(5, 6, 16, 18) = 0$ and $q(\cdot)(22, 25, 27, 29) = 1$

3 Local Search

Given a solution S
Sort $e \in S$ by decreasing amount of work
for h in $1 \leq h \leq 72$ **do**
  for each e' in S_h **do**
    for each e in S **do**
      $S' \leftarrow S \setminus < e', h > \cup < e, h >$
      if S' is **not** UNFEASIBLE **and** $f(S') < f(S)$
      then
        $S \leftarrow S'$
return S

This local search algorithm uses a best improvement strategy, as it tries to get the optimal assignment for each employee, instead of using only the first assignment that improves the solution. Also, it uses a reassignment neighborhood, because it focus on moving one employee to hours, instead of doing exchanges between employees in different hours. The algorithm sorts the employees by amount of work, so the employees with less work will be selected first, then a hour is selected so the algorithm starts moving employees from the available pool to that hour, substituting the assigned employees until it finds a better solution.