ALGORITHMIC METHODS FOR MATHEMATICAL MODELS
Course Project Report

Anass Benali
FIB - UPC Student
Barcelona, Spain
anass.benali@est.fib.upc.edu

Oriol Borrell
FIB - UPC Student
Barcelona, Spain
oriol.borrell.roig@est.fib.upc.edu

December 2019

1 Formal statement problems A, B and C

1.1 Base problem

Having a company that produces cars, the problem can be formally stated as follows.

Given:

- The set O of options that a car can have.
- The set C of classes a car belongs. Each class c is a set of options.
- A Boolean matrix $ClassOption_{c,o}$ whether a $c \in C$ has option $o \in O$
- An integer array $carsOfClass_c$ representing the numbers of cars of class $c \in C$ that will be produced
- A sequence P of positions to fill in the production line

Find a production schedule of cars that satisfies:

- Each position $p \in P$ has one and only one car of class $c \in C$ assigned
- For each $c \in C$, the corresponding number $carsOfClass_c$ cars need to be produced
- Each class $c \in C$ has the corresponding options $o \in O$ specified in $ClassOption_{c,o}$

1.2 Problem A

Given:

- The statement defined in the **base problem**.
- Two integers k_o and m_o for every option $o \in O$

Find a production schedule of cars that satisfies:

- That for every window of k_o consecutive cars, at most m_o can have option o.

1.3 Problem B

Given:

- The statement defined in the **problem A**.
- That a position $p \in P$ has a change if $p-1$ and $p+1$ exist, the classes at positions $p-1$ and p are different from each other and the classes at positions p and $p+1$ are also different from each other.

The *objective* is to minimize the number of positions where there is a change.

1.4 Problem C

Given:

- The statement defined in the **base problem**.
- Two integers k_o and m_o for every option $o \in O$
- That for every window of k_o consecutive cars in P, for each option $o \in O$, there is a violation if there are more than m_o cars with option o.

The *objective* is to minimize the number of violations.

2 Integer linear programming formulation

2.1 Base problem

$$
\begin{aligned}
&\mathcal{O} : \text{Set of options, index } o\\
&\mathcal{C} : \text{Set of classes, index } c\\
&classOption_{c,o} : \text{ whether class } c \in \mathcal{C} \text{ has option } o \in \mathcal{O}\\
&carsOfClass_c : \text{ number of cars of class } c \in \mathcal{C} \text{ to be produced}\\
&\mathcal{P} = \{1, 2, .., nPositions\} : \text{ Set of positions to fill, } nPosition = \sum_{c \in C} carsOfClass_c
\end{aligned}
$$

We model the problem as a Integer Linear Program (ILP) with the following decision variables:

$$
\begin{aligned}
&pc_{p,c} : \text{ whether a car of class } c \text{ is assigned to position } p\\
&po_{p,o} : \text{ whether a class with option } o \text{ is assigned to position } p
\end{aligned}
$$

Taking into account that each variable has to fulfill the following constraints:

$$
\begin{aligned}
&\sum_{c \in C} pc_{p,c} = 1 && \text{for all } p \in P \text{ (all positions have one car)}\\
&\sum_{p \in P} pc_{p,c} = carsOfClass_c && \text{for all } c \in C \text{ (carsOfClass is fulfilled)}\\
&pc_{p,c} = 1 \implies pc_{p,o} = classOption_{c,o} && \text{for all } p \in P,\ c \in C,\ o \in O \text{ (ClassOptions is fulfilled)}
\end{aligned}
$$

2.2 Problem A

Taking into account the variables and constraints defined in the **base problem**, we will model the following statement:

$$
\begin{aligned}
&\text{minimize } 1 \text{ (try to find a satisfiable model)}\\
&\text{subject to:}\\
&\sum_{i=p}^{p+k_o-1} po_{i,o} \leq m_o \quad \text{for all } o \in O, 1 \leq p \leq nPositions - k_o + 1 \text{ (window violations)}
\end{aligned}
$$

2.3 Problem B

Taking into account the variables and constraints defined in the **problem A**, and the following decision variable:

$$z : \text{Positive integer with the number of changes}$$

We model the following statement:

$$
\begin{aligned}
&\text{minimize } z \text{ (minimize the number of changes)}\\
&\text{subject to:}\\
&z \geq \sum_{c \in C} \sum_{p=2}^{nPosition-1} pc_{p-1,c} \neq pc_{p,c} \land pc_{p,c} \neq pc_{p+1,c} \text{ (number of changes)}
\end{aligned}
$$

2.4 Problem C

Taking into account the variables and constraints defined in the **base problem**, and the following boolean decision variable:

$$zopt_{p,o}: \text{ Whether a option } o \text{ is violated in position } p.$$

We model the following statement:

$$\text{minimize} \sum_{p \in P, o \in O} zopt_{p,o} \text{ (minimize the number of violations)}$$

subject to:

$$zopt_{p,o} = \left(\sum_{i=p}^{p+k_o-1} po_{i,o} \right) > m_o \text{ for all } o \in O, 1 \leq p \leq nPositions - k_o + 1 \text{ (number of violations)}$$

3 Greedy and Grasp pseudocode

3.1 Greedy

Our $q(\cdot)$ FUNCTION differences two states:

- If is the **first car**, we choose the class that has lower number of cars in carsOfClass. The ideal case is to put a class that only have 1 car.
- If is **not the first car**...
  - We will try to put any car that does not lead to a change.
  - If its not possible, we will prioritize the classes that will allow the assignation of 2 or more cars of that class. In order to do this, we check that, for each option present in the class, the number of 1's in the last $k_o - 2$ positions plus the value of the option of the car that we are going to assign has to be smaller than m_o.

    If there's no class that for all options the ratio is smaller than 1 we will choose those that are equal to 1, but knowing that in the following step this will lead us to another change. We will add 1 to be sure that if all the classes do not have any option, will take the previous condition rather than this one.
  - If a class has less than 1 car left to assign and forces a change, will be the last to be chosen.

$$q(p,o) = \begin{cases} \frac{-1}{carsOfClass_c} & if \quad p = 0 \\ 0 & if \quad S_{p-1} = c \\ max \left\{ \frac{\sum_{i=max(0,p-k_o+2)}^{p} po_{i,o} \times classOption_{c,o} + classOption_{c,o}}{m_o} \middle| o \in O \right\} + 1 & if \quad S_{p-1} \neq c \wedge carsOfClass_c > 1 \\ 3 & if \quad S_{p-1} \neq c \wedge carsOfClass_c \leq 1 \end{cases}$$

$$r(p,o) = \sum_{i=max(0,p-k_o)}^{p} po_{i,o}$$

Algorithm 1 GREEDY

```
function getFeasibleClass(p):
    feasibleClass ← ∅
    for c in C do:
        if |{o ∈ O|r(p, o) > m(o)}| = 0 then
            feasibleClass ← feasibleClass ∪ c
        end if
    end for
    return feasibleClass
end function

function Greedy:
    S ← ∅
    po ← Matrix(p,o)
    p = 1
    while | S | ≤ nPositions do:
        feasibleClass =: getFeasibleClass(p)
        if feasibleClass = ∅ then
            return INFEASIBLE
        end if
        S ← S ∪ argmin{q(p, c)|c ∈ feasibleClass ∧ carsOfClass(c) ≠ 0}
        carsOfClass(c) ← carsOfClass(c) - 1
        Update po_p
        p ← p + 1
    end while
    return S
end function
```

3.2 GRASP

$$q(p,o) = \begin{cases} \frac{-1}{carsOfClass_c} & if \quad p = 0 \\ 0 & if \quad S_{p-1} = c \\ max\left\{ \frac{\sum\limits_{i=max(0,p-k_o+2)}^{p} po_{i,o} \times classOption_{c,o} \; + \; classOption_{c,o}}{m_o} \middle| o \in O \right\} + 1 & if \quad S_{p-1} \neq c \wedge carsOfClass_c > 1 \\ 3 & if \quad S_{p-1} \neq c \wedge carsOfClass_c \leq 1 \end{cases}$$

$$r(p,o) = \sum_{i=max(0,p-k_o)}^{p} po_{i,o}$$

Algorithm 2 GRASP

function GETFEASIBLECLASS(p):
  feasibleClass $\leftarrow \emptyset$
  for c in C **do**:
    if $|\{o \in O | r(p,o) > m(o)\}| = 0$ **then**
      $feasibleClass \leftarrow feasibleClass \cup c$
    end if
  end for
  return feasibleClass
end function

function GRASP:
  S $\leftarrow \emptyset$
  po $\leftarrow$ Matrix(p,o)
  p = 1
  while | S | $\leq$ nPositions **do**:
    feasibleClass = getFeasibleClass(p)
    if $feasibleClass = \emptyset$ **then**
      return INFEASIBLE
    end if
    $q_{min} \leftarrow min\{q(p,c) | c \in feasibleClass \wedge carsOfClass(c) \neq 0\}$
    $q_{max} \leftarrow max\{q(p,c) | c \in feasibleClass \wedge carsOfClass(c) \neq 0\}$
    $RCL \leftarrow \{c \in feasibleClass \wedge carsOfClass(c) \neq 0 | q(p,c) \leq q_{min} + \alpha(q_{max} - q_{min})\}$
    $c \leftarrow$ select $c \in RCL$ at random
    $S \leftarrow S \cup c$
    carsOfClass(c) $\leftarrow$ carsOfClass(c) - 1
    Update po_p
    p $\leftarrow$ p + 1
  end while
  return S
end function

3.3 Code execution

Position	1	2	3	4	5
Class	1	1	2	4	4

3.3.1 Greedy results

The following table shows the result of the $q(\cdot)$ function for each candidate in both iterations:

Candidate	$q(\cdot)$ Iteration 1	$q(\cdot)$ Iteration 2
1	3	3
2	Infeasible	Infeasible
3	2	2
4	1.5	0
5	3	3
6	2	2
7	Infeasible	Infeasible

3.3.2 GRASP results

We used the same $q(\cdot)$ of GRASP so the costs are the same as in the Greedy results.
Having $q_{min} = 1.5$ and $q_{max} = 3$, the RCL contains the candidates with cost between $[1.5, 2.25]$. The RCL for the first iteration with an α of 0.5 is the following one: $RCL_1 = \{1, 3, 4, 6\}$.
We can't know the results of the second iteration because they depend on the candidate chosen in the first iteration.