OPTIMIZATION ALGORITHMS
Tabu search exercise

Diego Rossi - 724217

Initial feasible solution

As metaheuristic approach, tabu search needs an initial feasible solution to start its steps.

Table 1: Given data with a new computed column

Job	w_j	p_j	d_j	$d_j - p_j$
1	1	6	9	3
2	1	4	12	12
3	1	8	15	7
4	1	2	8	6
5	1	10	20	10
6	1	3	22	19

In order to find an initial feasible solution, I computed the difference array $d_j - p_j$, which gives a first idea of priority of the different jobs. Then, I ordered the rows according to the ascending order of the value of this column, getting the table below.

Table 2: Ordered rows, in ascending order, according to the last column

Job	w_j	p_j	d_j	$d_j - p_j$
1	1	6	9	3
4	1	2	8	6
3	1	8	15	7
2	1	4	12	12
5	1	10	20	10
6	1	3	22	19

The first column contains the ordered jobs, which define the initial solution:

$$x_0 = \begin{bmatrix} 1 & 4 & 3 & 2 & 5 & 6 \end{bmatrix}$$

that has an objective function value of $f(x_0) = 30$.

Move and neighborhood

I defined as *move* a swap between two adjacent elements of a current solution. If the swap if applied to the last element, ie. the one in position 6, it gets swapped with the first one.
To give an idea, below is showed an example: all the possible moves, starting from x_0, which populate $N(x_0)$:

$$x_A = \begin{bmatrix} 4 & 1 & 3 & 2 & 5 & 6 \end{bmatrix} \qquad x_B = \begin{bmatrix} 1 & 3 & 4 & 2 & 5 & 6 \end{bmatrix} \qquad x_C = \begin{bmatrix} 1 & 4 & 2 & 3 & 5 & 6 \end{bmatrix}$$

$$x_D = \begin{bmatrix} 1 & 4 & 3 & 5 & 2 & 6 \end{bmatrix} \qquad x_E = \begin{bmatrix} 1 & 4 & 3 & 2 & 6 & 5 \end{bmatrix} \qquad x_F = \begin{bmatrix} 6 & 4 & 3 & 2 & 5 & 1 \end{bmatrix}$$

By defining the move in this way, I maintain the feasibility of the solution, since none of the computed solutions has less or more jobs than the initial amount (6).

Tabu information

When a solution among the neighborhood is selected as next current solution, the correspondent move gets tabu active. This means that I don't want the swapped values to get back in their previous positions: so I forbid the opposite of the move I've just applied.
The tabu list keeps memory of vectors of 3 elements: $<$ *moved job*, *starting position of the moved job*, *tabu tenure* $>$.
Note that the values used in the code, go from 0 to 5, for both jobs and positions (ie. indexes). The adjustments to 1 to 6 are only made in the output/view, by adding +1.
With the considered move, we add 2 atomic moves to the tabu list at each iteration.
I chose a value of 5 for the tabu tenure, which provides a good intensification, but it also allows to explore some other different solutions, reaching anyway the optimum.

By looking at an example, if we are on position x and we move to x_A we add in the tabu list: $[0, 0, 5]$, $[4, 1, 5]$. This means that:

- I don't want job 0 (which is outputted as job $0 + 1 = 1$) to go back at position 0 (which is outputted as position $0 + 1 = 1$): this move has a tabu tenure of 5
- I don't want job 4 (which is outputted as job $4 + 1 = 5$) to go back at position 0 (which is outputted as position $1 = 2$): this move has a tabu tenure of 5

Tabu search steps

At this point, the tabu search method can start with its 1^{st} iteration.

1. I enter in the iteration with the current solution x (x_0 since it's the first iteration)
2. I generate its neighborhood $N(x)$, by swapping its element, as defined before
3. by looking at the tabu list, I check which of the solutions in $N(x)$ do not violate any tabu move. Those which are related to tabu moves, get removed.
   If the neighborhood remains empty, it means that all the solutions contained a tabu move. In this case, I apply the default aspiration criteria, which makes all the tabu moves free again. Note that in this instance the default aspiration criteria is never applied.
4. I search the solution, which provides the best objective function value, between the remained solutions
5. the moves by which the best solution was obtained, get now added to the tabu list
6. the tabu tenure values of the elements in the tabu list are decremented: if tabu tenure reached 0, they get removed, ie. made free again
7. best solution and tabu list are updated

These steps are repeated for the number of iterations.

10 first iterations

The 10 first iterations provide the following solutions:

$$x_1 = \begin{bmatrix}1 & 4 & 3 & 2 & 6 & 5\end{bmatrix} \quad x_2 = \begin{bmatrix}1 & 4 & 2 & 3 & 6 & 5\end{bmatrix} \quad x_3 = \begin{bmatrix}4 & 1 & 2 & 3 & 6 & 5\end{bmatrix}$$
$$x_4 = \begin{bmatrix}4 & 1 & 2 & 6 & 3 & 5\end{bmatrix} \quad x_5 = \begin{bmatrix}4 & 2 & 1 & 6 & 3 & 5\end{bmatrix} \quad x_6 = \begin{bmatrix}4 & 2 & 6 & 1 & 3 & 5\end{bmatrix}$$
$$x_7 = \begin{bmatrix}4 & 2 & 6 & 1 & 5 & 3\end{bmatrix} \quad x_8 = \begin{bmatrix}4 & 2 & 6 & 5 & 1 & 3\end{bmatrix} \quad x_9 = \begin{bmatrix}2 & 4 & 6 & 5 & 1 & 3\end{bmatrix}$$
$$x_{10} = \begin{bmatrix}2 & 6 & 4 & 5 & 1 & 3\end{bmatrix}$$

They provide the objective function values below.

$$f(x_1) = 23 \quad f(x_2) = 19 \quad f(x_3) = 19 \quad f(x_4) = 21 \quad f(x_5) = 24$$

$$f(x_6) = 27 \qquad f(x_7) = 29 \qquad f(x_8) = 34 \qquad f(x_9) = 34 \qquad f(x_{10}) = 35$$

Below it's represented the behavior of the objective function values and of the minimum value found found till the n^{th} iteration.

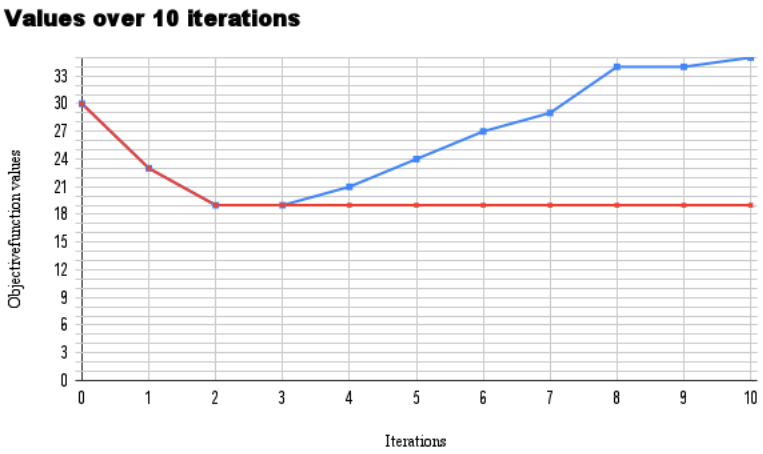

Figure 1:

As we can see, the value of the solutions deteriorates over the 10 iterations, after finding the global minimum. This is not a big deal, since by using tabu search it's known that in order to explore new areas of the solution space and to escape from (local) minimums, the value of the objective function may get worse.

Source code

The source code can be found at the address:
https://github.com/drossi99/OA_tabuSearchEx_rossi.