ISS team scheduling problem

Alexander Lazarev[1,2,3,4], **Varvara Gushchina**[2,4]

[1] *National Research University Higher School of Economics, Moscow;*
[2] *Lomonosov Moscow State University, Moscow;*
[3] *Institute of Physics and Technology State University, Moscow;*
[4] *Institute of control sciences, Moscow;*
jobmath@mail.ru; vg@kvartasoft.ru

We consider the International Space Station (ISS) scheduling problem. Cosmonauts and dispatchers have to perform given set of tasks $N = \{1, 2, ..., n\}$ during the flight time. Every task is characterized by such parameters like duration l_i, priority p_i, possible time of commencement $T_i = \{t_{i_1}, t_{i_2}, ..., t_{i_{k_i}}\}$ and complexity d_i. The problem is to distribute the tasks to the time planning horizon $H = \{t_1, t_2, ..., t_T\}$ in the most optimal way: it means performing all set of tasks and achieving the uniform load of all team members. The unit of the time planning horizon equals to 5 minutes.

Mathematical problem.

We have:

1. M cosmonauts and dispatcher;
2. set of tasks $N = \{1, 2, ..., n\}$;
3. time planning horizon $H = \{t_1, t_2, ..., t_T\}$ (k days, h hours per day).

Every task is characterized by:

1. priority $p_i = p_{ij}(t_j), i = 1, ..., n, j = 1, ..., T$;
2. duration l_i;
3. complexity d_i.

If $t_j \in T_i$ then $p_i \neq 0$, else $p_i = 0$.
The problem is to maximize the objective function:

$$max \sum_{i=1}^{n} \sum_{j=1}^{P} x_{ij} p_{ij}(t_j), \tag{1}$$

subject to

$$\sum_{i=1}^{n} \sum_{j=1+ah}^{(a+1)h} x_{ij} d_j \leq C, a = 0, ..., k-1. \tag{2}$$

Binary variables x_{ij} have the following meaning: $x_{ij} = 1$ if the task i is performed at the time t_j, and $x_{ij} = 0$ otherwise.

The exact algorithm was proposed to solve this problem. The algorithm is a superposition of Greedy algorithm and Branch-and-bound [1]. Both algorithms are often used to solve Knapsack Problem which variation, Multiply Knapsack Problem with Initial Capacity, is similar to assigned problem [2].

The proposed algorithm consicsts of 3 stages:

1. sorting tasks (greedy);
2. finding Upper Bound;
3. brunching.

This algorithm was programmed on C++ and it's work with different initial data was analized using received program.

Conclusion:

1. algorithm is exact;
2. algorithm is suitable for parallelize;
3. algorithm is able to develop in the event of a complication of the problem;
4. the number of possible solutions depends on the severity of restrictions.

References

1. *A.A. Lazarev, E. R. Gafarov* Scheduling theory. Problems and algorithms, Lomonosov Moscow State University, Moscow (2011). (in Russian)
2. *David Pisinger, Hans Kellerer, Ulrich Pferschy.* Knapsack problems // Springer. 2004