Commentary on the paper entitled
Towards Scheduling Over-Constrained Remote-Sensing Satellites

Gérard Verfaillie
ONERA/DCSD, Toulouse Center
2, av. Édouard Belin, BP 4025
31055 Toulouse Cedex 4, France
phone: +33 5 62 25 26 32
fax: +33 5 62 25 25 64
email: verfaillie@cert.fr

At first, the paper entitled *Towards Scheduling Over-Constrained Remote-Sensing Satellites*, presented by J. C. Pemberton from *Veridian Pacific-Sierra Research*, reminds us that the problem of the management of an observation satellite (either Earth observation, or space observation) is a difficult combinatorial constrained optimization problem.

As it is clearly explained in the paper, its input data are:

- a scheduling horizon h;
- a set R of user requests;
- associated with each user request $r \in R$, a set $C_u(r)$ of user-defined constraints, which define the conditions in which the user wants his/her request to be achieved;
- a set C_s of constraints expressing the limitations of the satellite: trajectory, manoeuvring ability, power, energy, memory on board, visibility windows, data downloading ...
- an optimization criterion c, which is often a function of the set of the selected requests.

The problem is to find a subset R' of R, which is feasible (the constraints in $C_s \cup [\cup_{r \in R'} C_u(r)]$ are all satisfied) and optimal according c.

In fact, there are three kinds of decision to make:

- about the selection: which requests to select ?
- about their order: in which order to achieve them ?
- about their starting times: at which time to start each of them ?

This problem is close to well-known problems in the *Operations Research* community, as the *Multi-Knapsack* problem, the *Job-Shop Scheduling* problem, or the *Traveling Salesman* problem with time windows: well-known but difficult problems, at least if one looks for optimal solutions or solutions the distance to the optimum of which can be guaranteed.

Generic frameworks, currently used to represent combinatorial constrained optimization problems, such as *Mixed Linear Programming* or *Constraint programming*, can be used to represent it.

Consequently, generic tools dedicated to these frameworks, such as *CPlex* (*Ilog Planner*) for *Mixed Linear Programming* or *Ilog Solver* for *Constraint programming*, can be used to solve it.

But, as J. C. Pemberton said, if these tools allow the observation satellite scheduling problem to be easily represented and small instances to be solved efficiently, they do not allow generally large realistic instances to be solved correctly: for these instances, they do not manage to produce optimal solutions within a reasonable time; if they produce optimal solutions, they do not manage to prove their optimality; what is worse, the quality of the solutions they deliver when interrupted is generally poor.

According to J. C. Pemberton, and I agree with him, it is due to the difficulty of the selection of a small optimal feasible subset of requests among a large set of candidate requests, when the priority levels or the weights associated with each request are very similar. The problem is at once over-constrained and uniform, and it is known that over-constrained uniform optimization problems are very difficult to solve: neither bound computing mechanisms used in the *Linear Programming* tools, nor constraint propagation mechanisms used in the *Constraint Programming* tools manage to make up for the explosion of the search space.

In fact, all the methods available to solve the problem, at least as well as possible, in order to use these satellites as efficiently as possible, belong to the following four families:

- *Heuristic Greedy* algorithms;
- *Local Search* algorithms;
- *Tree Search* algorithms;
- *Dynamic Programming* algorithms.

Algorithms of the first two families are inexact (no optimality guarantee). Those of the last two families are exact (optimality guarantee) when they are not interrupted. In each of these families, a lot of variants can be defined. Between them, a lot of hybrid algorithms can be defined too.

What J. C. Pemberton proposes in this paper is a particular hybrid algorithm: a combination of *Greedy* and *Tree Search* algorithms.

As he observed that *Tree Search*, combined with *Constraint Propagation*, as it is implemented in *Ilog Solver* and *Scheduler*, can solve efficiently small instances, he orders the set of the requests by decreasing priority, he cut it into small subsets, solves optimally the problem that includes only the first subset, then solves optimally the problem that includes only the second subset with the solution of the previous problem locked, and so on.

The algorithm he describes can be viewed as a *Greedy* algorithm, the decisions of which are made on sets of choices rather than on elementary choices. Besides, when the subsets he considers are reduced to be singletons, his algorithm is a simple *Greedy* algorithm. At the contrary, when the set of the requests is not cut, his algorithm is a simple *Tree Search* algorithm. Between both these extrema, any tradeoff between quality and time is possible.

Research is currently active about hybrid search mechanisms, because people think that it is one of the ways of improving the efficiency of the known mechanisms. J. C. Pemberton's proposal is interesting, simple and generic. It could be used on many other problems.

Experiments he carried out show improvements in terms of quality of the produced solutions, when compared with a simple greedy algorithm. But, I would like to take advantage of this commentary to ask people who report experiments about the use of inexact optimization methods to give information both in terms of quality and of time. And the best way of reporting seems to be a quality profile, which shows how the quality of the best solution found so far evolves with time.

Finally, I would like to note that, at least for the simplified problem used in the experiments (fixed request starting times, fixed request durations, one instrument), there exists a very simple exact *Dynamic Programming* algorithm, which can certainly outperform all the other algorithms, in terms of quality (due to optimality) and in terms of time. Unfortunately, it is not the case for the general observation satellite management problem.