A Web Interface for Satellite Scheduling Problems

Fatos Xhafa, Carlos Garcia
Department of Computer Science
Technical University of Catalonia, Spain
Email: {fatos,cgarcia}@cs.upc.edu

Admir Barolli
Department of Applied Informatics
Logos University, Albania
Email: admir.barolli@gmail.com

Makoto Takizawa
Department of Advanced Sciences
Hosei University, Japan
Email: makoto.takizawa@computer.org

***Abstract*—Mission planning plays an important role in satellite control systems, especially with increase of number of satellites and more complex missions to be planned. In a general setting, the satellite mission scheduling consists in allocating tasks such as observation, communication, etc. to resources (spacecrafts (SCs), satellites, ground stations). For instance, in ground station scheduling the aim is to compute an optimal planning of communications between satellites and operations teams of Ground Station (GS). Because the communication between SCs and GSs can be done during specific window times, this problem can also be seen as a window time scheduling problem. The required communication time is usually quite smaller than the window of visibility of SCs to GSs, however, clashes are produced, making the problem highly constrained. In this work we present a web interface for solving satellite scheduling problems through various heuristic methods. The web interface enables the users to remotely solve their problem instances through a selection of heuristic methods such as local search methods (Hill Climbing, Simulated Annealing and Tabu Search) and population-based methods (Genetic Algorithms and variants). The user can select to solve previously generated instances by the STK simulation toolkit or generate their own problem instances. The heuristic methods are easily configurable so that users can simulate a variety of scenarios, problem sizes, etc. The execution of the heuristics methods is done at a HPC Cluster infrastructure supporting efficient execution of various solvers. Additionally, the web application allows users to keep track of their executions as well as to share problem instances with other users.**

***Keywords*-Web Interface; Satellite Scheduling; Ground Station; Heuristics; Hill Climbing; Simulated Annealing; Tabu Search; Genetic Algorithms; Simulation; Meta-heuristics.**

I. Introduction

The design of intelligent mission planning for satellite systems is a long standing problem in satellite control systems. Nowadays, developed mission planning systems is of interest not only to large aerospace agencies such as ESA (European Space Agency) [1], [5], [6] and NASA [3] , bu also to many smaller science and technology projects from research institutions and universities requiring mission planning [4], [17]. Indeed, there is a growing number of small satellites being launched for science and technology missions. With such increasing number of satellites and of the missions, the mission planning optimization is crucial not only to optimize the resource usage but primarily to ensure mission accomplishment of resource-constrained satellites that need to communicate with capacity-constrained ground stations. In fact, there is an emerging trend of launching constellations of small satellites for scientific studies using data gathering from remote sensing.

Ground Station Scheduling is one of the most important problems in the field of Satellite-Scheduling. It consists in computing feasible planning of communications between satellites or spacecraft (SC) and operations teams of Ground Station (GS). The problem arises in many real life applications and projects, such as hurricane prediction [13], tele imagery systems and earth observation [11], [15], etc.

Ground Station Scheduling is a very complex problem due to its over-constrained nature.

Constraints and requirements: There is a large set of constraints. In fact, this is the first major difference between the problems of conventional scheduling and that of Ground Station scheduling. First, there are restrictions on the communication time required for each SC in a period of time. Secondly, there are restrictions on the visibility of each window on each Spacecraft Ground Station, i.e. the time at which each SC can communicate with each GS in a given time period. Resources are thus not available at all times for mission allocation.

Communication time requirement: The length of the communication is variable, where it should be at least the required communication time and at most the maximum time within which the window visibility ends or the visibility window of another communication starts.

Visibility requirements and clashes: A ground station can communicate with a SC only when SC is within the transmitting angle of the ground station. A spacecraft has three types of visibility to a ground station, namely: (1) AOS-VIS: Acquisition of Signal, Visible. This indicates the time when the SC appears in the line of sight of the GS; (2) AOS-TC: Acquisition of Signal, Tele-command. This is time when GS is allowed to send signal to SC. A visibility clash of two spacecraft happens when the AOS time of second spacecraft starts before the LOS time of first one.

All scheduling variants, in their general formulations, are highly constrained problems and have been shown computationally hard [2], [12], [14], [25]. Therefore their resolution is tackled through heuristics approaches such as Local Search and Genetic Algorithms [8], [9], [18]–[24] or specific formulations such as image acquisition [10].

In this work we present a web interface for solving satellite scheduling problems through various heuristic methods. The aim is to enable users to remotely solve their problem instances through a selection of heuristic methods such as local search methods (Hill Climbing, Simulated Annealing and Tabu Search) and population-based methods (Genetic Algorithms and variants).

The rest of the paper is organized as follows. In Section II we introduce some main concepts and background on satellite scheduling problems and its fitness functions in Section III. The resolution methods are briefly presented in Section IV. In Section V we present the architecture of the Web application and its integration with different resolution methods. The main use cases are shown as well. Finally, we conclude this work in Section VI.

II. Satellite Scheduling Problems

1) Ground stations and spacecrafts/satellites: Ground Stations are terrestrial terminals designed for extra-planetary communications with SCs. SCs are extra-planetary crafts, such as satellites, probes, space stations, orbiters, etc. Ground stations communicate with a spacecraft by transmitting and receiving radio waves in high frequency bands (e.g. microwaves). A ground station usually contains more than one satellite dish. Each dish is usually assigned to a specific space mission. With the scheduling from control center, dishes are able to handle and switch among mission spacecrafts.

2) Problem input instance: The input instance is defined in Table I.

Table I
Parameters defining the input instance

Parameter	Description
$SC\{i\}$	List of Spacecrafts in the planning
$GS\{g\}$	List of Ground Stations in the planning
N_days	Number of days for the schedule
$TAOS\_VIS(i)(g)$	Visibility time of GS to SC
$TLOS\_VIS(i)(g)$	Time GS looses signal from SC
$TReq(i)$	Communication time required for spacecrafts

3) Objectives: Different types of objectives can be formulated, namely, maximizing matching of visibility windows of spacecrafts to communicate with ground stations, minimizing the clashes of different spacecrafts to one ground station, maximizing the communication time of spacecraft with ground station, and maximizing the usage of ground stations. The challenge here is to optimize several objectives.

4) Problem output: A solution procedure to the problem should output the values of the parameters defined in Table II.

Table II
Parameters defining the problem output

Parameter	Description
$T_{Start}(i)(g)$	Starting time of the communication $SC(i) - GS(g)$
$T_{Dur}(i)(g)$	Duration time of the communication $SC(i) - GS(g)$
$SC\_GS(i)$	The GS assigned to every $SC(i)$.
$Fit_{LessClash}$	The fitness of minimizing the collision of two or more SC to the same GS for a given time period (measured from 0 to 100).
$Fit_{TimeWin}$	The fitness value corresponding to time access window for every pair $GS - SC$ (measured from 0 to 100).
Fit_{Req}	Fitness value corresponding to satisfying the requirement on the mission communication time (measured from 0 to 100).
Fit_{GSU}	Fitness value corresponding to maximizing the usage of all GS during the planning (measured from 0 to 100).

III. Scheduling fitness types

One of the major complexities of the mission operations scheduling comes from the many objectives that can be sought for the problem. These objectives are related to visibility window, communication clashes, communication time and ground station resource usage, among others. The total fitness function, besides being composed of multiple objectives, poses the challenge of how to combine them and in which order to evaluate them. For the combination, one can adopt a hierarchical optimization approach based on the priority of the objectives or a simultaneous optimization approach. In the former, objectives are sorted according to some priority criteria and are optimized according that ordering. In the later, objectives are simultaneously optimized, e.g. by summing up all fitness functions into one single fitness function.

We define next the four main objectives that would compose the fitness function.

A. Access window fitness

Visibility windows are the time periods when a GS has the possibility to set-up a communication link with a SC. The objective is that all or the largest possible number of generated communication links to fall into access windows and thus achieve as many communications as possible. In the following equation, $W_{(g,i)}$ is the Access Window set for Ground Station g and Spacecraft i, $T_{Start}(s)$ and $T_{End}(s)$ are the start and end of each access window.

$$AW(g,i) = \cup_{s=1}^{S}[T_{AOS(g,i)}(s), T_{LOS(g,i)}(s)] \qquad (1)$$

Then, we define the final Access Window fitness of the scheduling solution (Fit_{AW}) calculated as follows:

$$f(n) = \begin{cases} 1, & \text{if } [T_{Start}(n), T_{Start}(n)+ \\ & +T_{Dur}(n)] \subseteq AW(n_g, n_i), \\ 0, & \text{otherwise.} \end{cases}$$

$$Fit_{AW} = \frac{\sum_{n=1}^{N} f(n)}{N} * 100, \tag{2}$$

where n value corresponds to an event, N is the total number of events of an entire schedule, g is a ground station and i a spacecraft. The fitness of access window is normalized so that it's value is within 0 to 100.

B. Communication clashes fitness

Communications clash represents the event when the start of one communication task happens before the end of another one on the same ground station. The objective is to minimize the clashes of different spacecrafts to one ground station. To compute the number of clashes, SCs are sorted by their start time. If, as a result of the sorting:

$$T_{Start}(n+1) < T_{Start}(n)+T_{Dur}(n),\ 1 \leq n \leq N-1 \tag{3}$$

where n value corresponds to an event and N is the total number of events of an entire schedule, then there is a clash. The fitness will be reduced, and one of the clashed entries has to be removed from the solution. The total fitness of communication clashes is then:

$$f(n) = \begin{cases} -1, & \text{if } T_{Start}(n+1) < T_{Start}(n) + T_{Dur}(n), \\ 0 & \text{otherwise.} \end{cases}$$

$$Fit_{CS} = \frac{N + \sum_{n=1}^{N} f(n)}{N} \tag{4}$$

C. Communication time requirement fitness

The objective is to maximize the communication time of spacecrafts with ground stations so that every spacecraft $SC(i)$ will communicate at least $T_{req}(i)$ time. Thus, a sufficient amount of time should be granted for TTC (Telemetry, Tracking and Command). For example, satellites that need to download huge amount of image data require more time for linking with ground stations. These communications, especially for data download tasks are usually periodical tasks (e.g. 2 hours communication for SC1 each day, 5 hours data downlink for SC2 every 2 days, etc.) A matrix is used to define those requirements, which is used as the input for the scheduling system.

The fitness is calculated by summing up all the communication link durations of each spacecraft, and dividing them in the required period to compare if the scheduled time matches requirements (see Eqs. (5)).

$$T_{Start}(m) > T_{From}(k)$$
$$T_{Start}(n) + T_{Dur}(n) < T_{TO}(k)$$
$$T_{Comm}(k) = T_{Dur}(j) \tag{5}$$
$$f(k) = \begin{cases} 1, & \text{if } T_{Comm}(k) \geq T_{REQ}(k), \\ 0 & \text{otherwise.} \end{cases}$$
$$FIT_{TR} = \frac{\sum_{k=1}^{K} f(k)}{N} \cdot 100.$$

D. Ground station usage fitness

Given that the number of ground stations is usually smaller than the number of spacecrafts missions, the objective is to maximize the usage of ground stations, that is, try to reduce the idle time of a ground station. A maximized usage would contribute to provide additional time for SC communications.

This fitness value is calculated as the percentage of ground stations occupied time by the total amount of the possible communication time. The more a GS is used, the better is the corresponding schedule.

$$Fit_{GU} = \frac{\sum_{n=1}^{N} T_{Dur}(n)}{\sum_{g=1}^{G} T_{Total}(g)} \cdot 100. \tag{6}$$

where N is the number of events of an entire schedule, G is the number of ground stations and $T_{Total(g)}$ is the total available time of a ground station

E. Combination of fitness objectives

The fitness objectives defined above (FIT_{AW}, FIT_{CS}, FIT_{TR}, FIT_{GU}) are conceived as fitness modules so as to facilitate the design phase of the scheduler to easily plug-in other fitness objectives. From the definition of the fitness objectives, we can observe that some of them can be applied in serial fashion (due dependencies, denoted serial-FM), while some others can be applied in parallel (denoted parallel-FM). We can combine all the fitness modules into one total fitness function using weights for different fitness module:

$$Fit = \sum_{i=1}^{n} w_i \cdot Fit_S(i) + \sum_{j=1}^{m} w_j \cdot Fit_P(j) \tag{7}$$

where w_i, w_j are the weights of fitness modules, $Fit_S(i)$ and $Fit_P(j)$ are the fitness values from Serial-FMs and Parallel-FMs, and n, m are the number of fitness modules, resp. More precisely, we define the total fitness function as follows:

$$Fit_{TOT} = \lambda \cdot Fit_{Win} + Fit_{Req} + \frac{Fit_{LessClash}}{10} + \frac{Fit_{GSU}}{100}. \tag{8}$$

for some λ (here set to $\lambda = 1.5$).

IV. Resolution Methods

Due to computational intractability of the problem heuristic and meta-heuristic approaches are the *de facto* approach to solve the problem for practical purposes).

A. Local Search Methods

1) Hill Climbing: Hill Climbing (HC) is local search algorithm and is based on incremental improvements of solutions as follows: it starts with a solution (which may be randomly generated or *ad hoc* computed) considered as the current solution in the search space. The algorithm examines its neighboring solutions and if a neighbor is better than current solution then it can become the current solution; the algorithm keeps moving from one solution to another one in the search space until no further improvements are possible. There are several variants of the algorithm depending on whether a simple climbing, steepest ascent climbing or stochastic climbing is done:

- *Simple climbing*: the next neighbor solution is the first that improves current solution.
- *Steepest ascent climbing*: all neighbor solutions are examined and the best one is chosen as next solution.
- *Stochastic climbing*: a neighbor is selected at random, and according to yielded improvement of that neighbor is decided whether to choose it as next solution or to examine another neighbor. This kind of climbing has more general forms known as Metropolis and Simulated Annealing algorithms.

We present the pseudo-code of HC in Alg. 1. The algorithm first generates a solution which serves as starting point in the search space. Then, the algorithm iteratively selects a movement based on current solution, evaluates the resulting movement in terms of possible improvements with respect to current solution. If the resulting neighbor improves fitness of current solution, the current solution is moved to the new neighbor and so on. In the algorithm the function $\delta(s, m)$ computes the improvement yielded by applying movement m to current solution s (as usually, for maximization, a positive value of δ function means improvement with respect to fitness of current solution).

2) Simulated Annealing: Simulated Annealing (SA) algorithm is inspired by the cooling process of metals in which a material is heated and then cooled in a controlled way to increase the size of its crystals and reduce their defects. SA algorithm is a generalization of the Hill Climbing (HC) heuristic. Indeed, SA consists of a sequence of executions of HC with a progressive decrement of the temperature starting from a an “high” temperature, where almost any move is accepted, to a low temperature, where the search resembles HC. In fact, it can be seen as a hill-climber with an internal mechanism to escape local optima (see pseudo-code in Alg. 2). In SA, the solution s' is accepted as the new current solution if $\delta \leq 0$ holds, where $\delta = f(s') - f(s)$.

Algorithm 1 Hill Climbing algorithm for maximization. f is the fitness function.

```
1: START: Generate an initial solution $s_0$;
2: $s = s_0$; $s^* = s_0$; $f^* = f(s_0)$;
3: repeat
4:     MOVEMENT SELECTION: Choose a movement
       $m = select\_movement(s)$;
5:     EVALUATE & APPLY MOVEMENT:
6:     if $\delta(s, m) \geq 0$ then
7:         $s' = appply(m, s)$;
8:         $s = s'$;
9:     end if
10:    UPDATE BEST SOLUTION:
11:    if $f(s') > f(s^*)$ then
12:        $f^* = f(s')$;
13:        $s^* = s'$;
14:    end if
15:    return $s^*$, $f^*$;
16: until (stopping condition is met)
```

Additionally, to allow escaping from a local optimum, moves that increase the energy function are accepted with a decreasing probability $\exp(-\delta/T)$ if $\delta > 0$, where T is the temperature parameter (see function *Accept* in Alg. 2). The decreasing values of T are controlled by a *cooling schedule*, which specifies the temperature values at each stage of the algorithm, what represents an important decision for its application (a typical option is to use a proportional method, like $T_k = \alpha \cdot T_{k-1}$). SA usually gives better results in practice, but tends to be rather slow to converge to good solutions.

Algorithm 2 : Pseudo-code of Simulated Annealing.

```
$t$ := 0
Initialize $T$
$s0$ := Initial_Solution()
$v0$ := Evaluate($s0$)
while (stopping condition not met) do
    while $t$ mod MarkovChainLen = 0 do
        $t$ := $t$+1
        $s1$ := Generate($s0$,$T$)    //Move
        $v1$ := Evaluate($s1$)
        if Accept($v0$,$v1$,$T$) then
            $s0$ := $s1$
            $v0$ := $v1$
        end if
    end while
    $T$ := Update($T$)
end while
return $s0$
```

3) Tabu Search: Tabu Search (TS) method was introduced by Glover [7] as a high-level algorithm that uses other specific heuristics to guide the search; the objective is to perform an intelligent exploration of the search space that would eventually allow to avoid getting trapped into local optima. The objective is thus to remedy one of the main issues of local search methods, namely the useless search in neighborhood of local optima without further improvements due to re-visiting solutions or paths of solutions already explored. This is achieved by giving the tabu status to solutions visited in the recent search. TS is also designed to be a flexible method, so that the tabu status of solutions can be waived, in case they have been prohibited for a long while or if they satisfy some aspiration criteria. The classification of some solutions as tabu is achieved through the intelligent use of adaptive memory, which is allowed to evolve and eventually change the status of tabu solutions.

The main features of the TS method are that of *adaptive memory* and *responsive exploration*. Again, the adaptive memory is the basis to guide the search in taking intelligent decisions. This gives the TS method advantages with regard to other memoryless methods, being these local search methods (Hill Climbing, Simulated Annealing, etc.) or population based methods (Genetic Algorithms, Memetic Algorithms, etc.). On the other hand, the responsive exploration enables the method to select some solutions which though not so good at the current search iteration might at long run lead to promising areas of good solutions (see Alg. 3).

Algorithm 3 Tabu Search Algorithm

```
begin
Compute an initial solution $s$;
let $\hat{s} \leftarrow s$;
Reset the tabu and aspiration conditions;
while not termination-condition do
  Generate a subset $N^*(s) \subseteq N(s)$ of solutions such that:
      (none of the tabu conditions is violated) or (the
  aspiration criteria hold)
  Choose the best $s' \in N^*(s)$ with respect to the cost
  function;
  $s \leftarrow s'$;
  if improvement($s', \hat{s}$)) then
    $\hat{s} \leftarrow s'$;
  end if
  Update the recency and frequency;
  if (intensification condition) then
    Perform intensification procedure;
  end if
  if (diversification condition) then
    Perform diversification procedures;
  end if
end while
return $\hat{s}$;
end;
```

B. Population-based Methods: Genetic Algorithms

GAs have shown their usefulness for the resolution of many computationally hard combinatorial optimization problems. Their main features are briefly described next (see Alg. 4 for a template). Besides, the web interface offers variations of GA such as Steady State, Struggle and Struggle Hash GA implementations.

Population of individuals: Unlike local search techniques that construct a path in the solution space jumping from one solution to another one through local perturbations, GAs use a population of individuals giving thus the search a larger scope and chances to find better solutions. This feature is also known as "exploration" process in difference to "exploitation" process of local search methods.

Fitness: The determination of an appropriate fitness function, together with the chromosome encoding are crucial to the performance of GAs. Ideally we would construct objective functions with "certain regularities", i.e. objective functions that verify that for any two individuals which are close in the search space, their respective values in the objective functions are similar.

Selection: The selection of individuals to be crossed is another important aspect in GAs as it impacts on the convergence of the algorithm. Several selection schemes have been proposed in the literature for selection operators trying to cope with premature convergence of GAs.

Crossover operators: Use of crossover operators is one of the most important characteristics. Crossover operator is the means of GAs to transmit best genetic features of parents to offsprings during generations of the evolution process.

Mutation operators: These operators intend to improve the individuals of a population by small local perturbations. They aim to provide a component of randomness in the neighborhood of the individuals of the population.

Escaping from local optima: GAs have the ability to avoid falling prematurely into local optima and can eventually escape from them during the search process.

Convergence: The convergence of the algorithm is the mechanism of GAs to reach to good solutions. A premature convergence of the algorithm would cause that all individuals of the population be similar in their genetic features and thus the search would result ineffective and the algorithm getting stuck into local optima. Maintaining the diversity of the population is therefore very important to this family of evolutionary algorithms.

V. Web Interface for Satellite Scheduling Problems

A. Architecture

The Web application follows a standard Client-Server architecture and is implemented using LAMP (Linux + Apache + MySQL+ PHP) technology (see Fig. 1). Remote users (clients) submit their requests according to several

Algorithm 4 Genetic Algorithm template

```
Generate the initial population $P^0$ of size $\mu$;
Evaluate $P^0$;
while not termination-condition do
    Select the parental pool $T^t$ of size $\lambda$; $T^t := Select(P^t)$;
    Perform crossover procedure on pairs of individuals in $T^t$ with probability $p_c$; $P_c^t := Cross(T^t)$;
    Perform mutation procedure on individuals in $P_c^t$ with probability $p_m$; $P_m^t := Mutate(P_c^t)$;
    Evaluate $P_m^t$ ;
    Create a new population $P^{t+1}$ of size $\mu$ from individuals in $P^t$ and/or $P_m^t$ ;
    $P^{t+1} := Replace(P^t; P_m^t)$
    $t := t + 1$;
end while
return Best found individual as solution;
```

use cases (problem creation/selection, resolution method selection and configuration, solver execution and solution reception and visualisation).

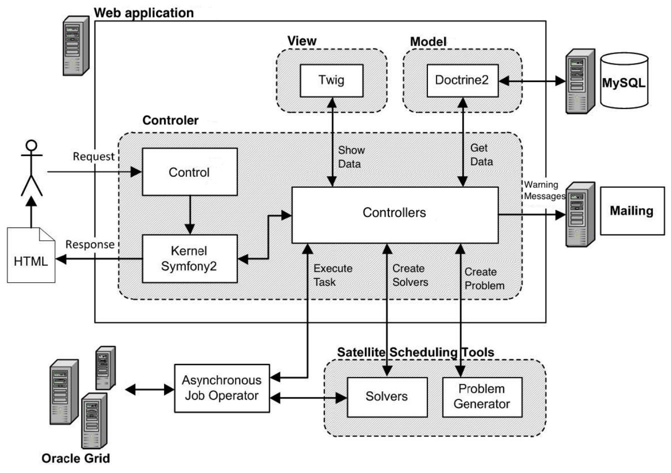


Figure 1. The architecture of the web application.

In this architecture we can distinguish:

- The **web application** is implemented according to Model-View-Controller paradigm (using Symfony2 in Controller layer, the Doctrine2 in the Model layer for data persistence with MySQL and Twig in the View layer).
- The **Satellite Scheduling Tools** include the existing solvers, creation of new solvers as well as using existing problem instances and creation of new problem instances.
- The **Asynchronous Job Operator** is a module implemented at RDLab[1] for submitting executions to the HPC Cluster via the Oracle Grid Engine queuing system.
- The **Mailing** server is used to notify the user about the executions and other warning services.

[1]http://rdlab.cs.upc.edu/index.php/en/

B. Main Use Cases

To apply the main use cases of the web application[2] prior registration in the site is required.

1) Problem instance creation and selection procedure: The user can select a problem instance (see Fig. 2) either from a general list of problems (see Fig. 3) or can create his own problem instance (see Fig. 4). In the former case the list of problems include default instances[3] generated using the Satellite Tool Kit [16] of small, medium and large sizes. All instances are coded in XML.

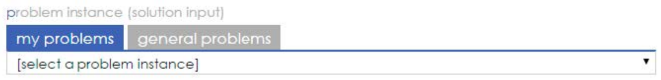
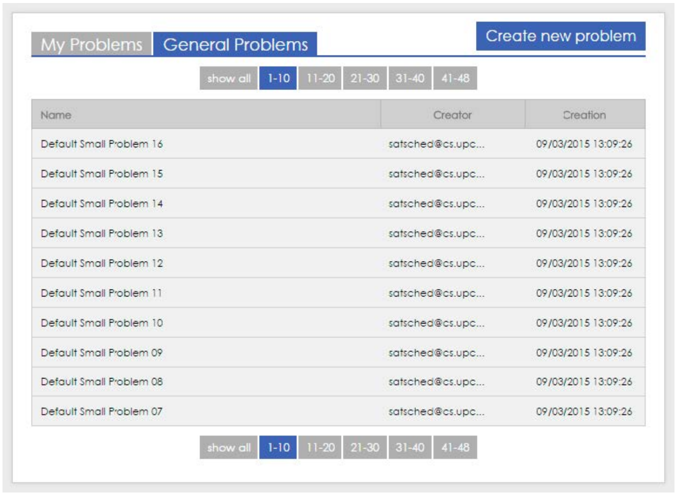

problem instance (solution input)
my problems
general problems
[select a problem instance]

Figure 2. A snapshot of problem selection procedure.



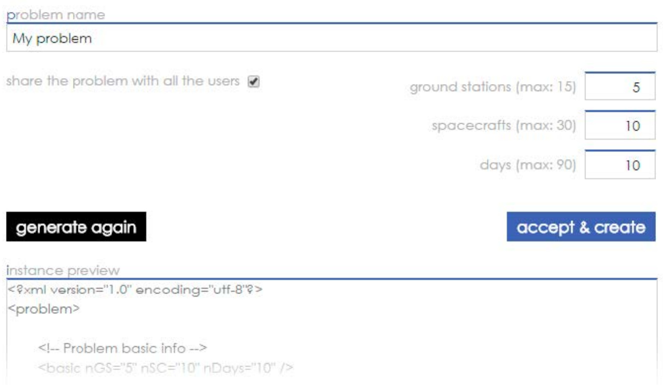
Name	Creator	Creation
Default Small Problem 16	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 15	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 14	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 13	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 12	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 11	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 10	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 09	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 08	satsched@cs.upc...	09/03/2015 13:09:26
Default Small Problem 07	satsched@cs.upc...	09/03/2015 13:09:26

Figure 3. A snapshot of selecting a problem instance from an existing list.

2) Resolution method selection procedure: The user can select a resolution method from a set of existing solvers, which include local search solvers (Hill Climbing, Simulated Annealing and Tabu Search) and population based solvers (GA base, GA Steady State, GA Struggle and GA Struggle Hash) (see Fig. 5). First, the user can choose to execute an existing solver again (see Fig. 6) from a list or to create a new execution (see Fig. 7).

[2]http://weboptserv.cs.upc.edu/satsched/web/

[3]The XML problem instance files can be downloaded from http://www.cs.upc.edu/∼fatos/GSSchedulingInputs.zip


problem name

My problem

share the problem with all the users ✓

ground stations (max: 15) 5

spacecrafts (max: 30) 10

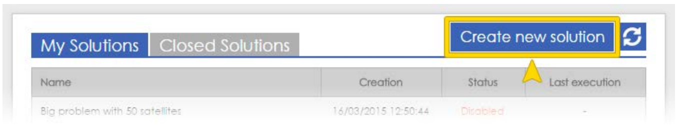
days (max: 90) 10

instance preview

```
<?xml version="1.0" encoding="utf-8"?>
<problem>

    <!-- Problem basic info -->
    <basic nGS="5" nSC="10" nDays="10" />
```

Figure 4. A snapshot of parameter specification for creating a new problem instance.



Name	Creation	Status	Last execution
Big problem with 50 satellites	16/03/2015 12:50:44	Disabled	-

Figure 5. A snapshot of solving a problem instance procedure.

Solver selection and configuration: Once a solver is selected from the top down list, the corresponding parameter configuration formulaire is presented (see Fig. 8) where the user can specify the desired parameter values (checking procedures are also implemented to control the valid values of parameters).

Reception and visualisation of the results: The results of the solution can be both downloaded as a file and visualised (see Fig. 9). The user is also informed about the ground station usage and the communication of ground stations with the satellites.

Archive of executions: The user can maintain in his account the archive of his previous executions and their status (see a snapshot in Fig. 10). For every execution, the problem instance, parameter setting and the best solution found are kept.

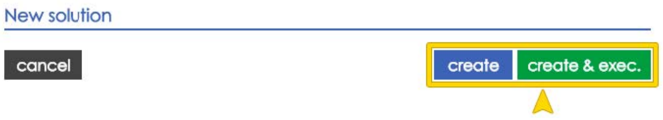
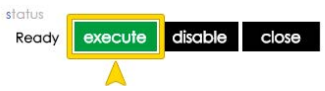
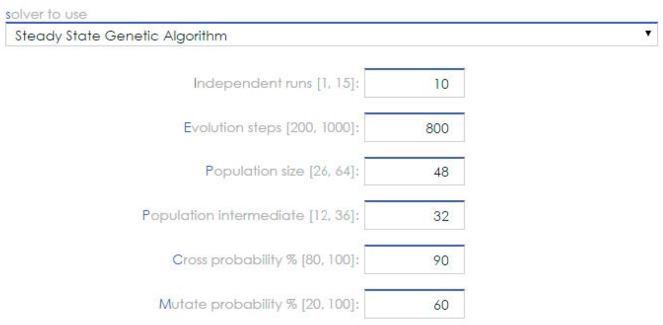


Figure 6. A snapshot of selecting a solver execution from an existing list.


status

Ready

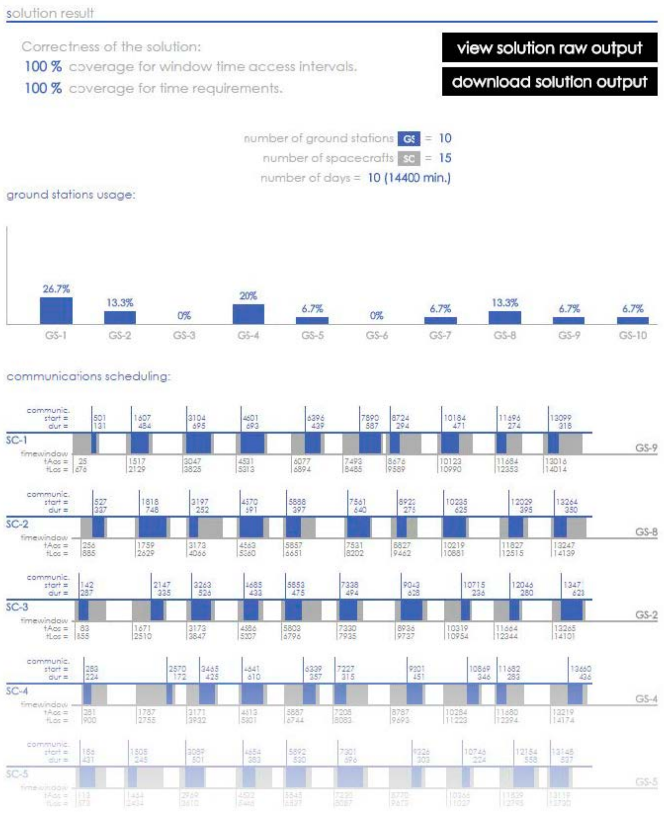
Figure 7. A snapshot of the procedure for creating a new solver instance.


solver to use

Steady State Genetic Algorithm

Independent runs [1, 15]:	10
Evolution steps [200, 1000]:	800
Population size [26, 64]:	48
Population intermediate [12, 36]:	32
Cross probability % [80, 100]:	90
Mutate probability % [20, 100]:	60

Figure 8. A snapshot of the procedure for selecting and configuring a concrete solver.


solution result

Correctness of the solution:

100 % coverage for window time access intervals.

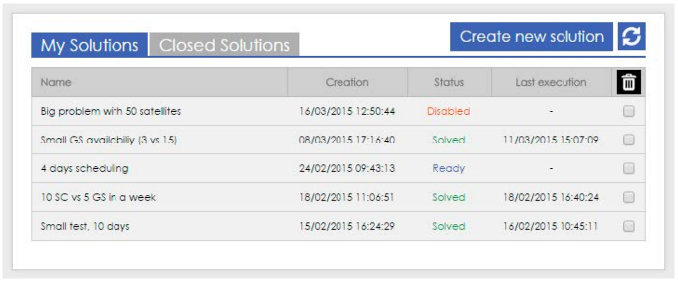
100 % coverage for time requirements.

number of ground stations GS = 10

number of spacecrafts SC = 15

number of days = 10 (14400 min.)

Figure 9. A snapshot of the procedure for downloading and visualising the solution.



Name	Creation	Status	Last execution
Big problem wi h 50 satellites	16/03/2015 12:50:44	Disabled	-
Small GS availabiliy (3 vs 15)	08/03/2015 17:16:40	Solved	11/03/2015 15:07:09
4 days scheduling	24/02/2015 09:43:13	Ready	-
10 SC vs 5 GS in a week	18/02/2015 11:06:51	Solved	18/02/2015 16:40:24
Small test, 10 days	15/02/2015 16:24:29	Solved	16/02/2015 10:45:11

Figure 10. A snapshot of archive of “My Solutions”.

VI. Conclusions and Future Work

In this work we have presented a Web interface for solving satellite scheduling problems through various heuristic methods. The satellite mission scheduling consists in allocating tasks such as observation, communication, etc. to resources (spacecrafts (SCs), satellites, ground stations) and is known to be highly constrained problem and hard to solve to optimality. The web interface enables the users to remotely solve their problem instances through a selection of heuristic methods such as local search methods (Hill Climbing, Simulated Annealing and Tabu Search) and population-based methods (Genetic Algorithms). The user can select to solve previously generated instances by the STK simulation toolkit or generate their own problem instances. The heuristic methods are easily configurable so that users can simulate a variety of scenarios, problem sizes, etc. The execution of the heuristics methods is done at a HPC Cluster infrastructure supporting efficient execution of various solvers. Additionally, the web applications allows user to keep track of their executions as well as share instance problems with other users.

In our future work, we would like to add new features to the web interface such as sharing the configurations for the best found solutions by different users.

References

[1] L. Barbulescu, A. Howe, J. Watson, L. Whitley. Satellite range scheduling: a comparison of Genetic, Heuristic and Local Search. *PPSN*, VII: 611-620, 2002.

[2] L. Barbulescu, J.-P. Watson, L. D. Whitley and A. E. Scheduling space-ground communications for the air force satellite control network. *Journal of Scheduling*, **7**(1), 7-34, 2004.

[3] L. Barbulescu, A.E. Howe, D. Whitley. AFSCN scheduling: How the problem and solution have evolved. *Mathematical and Computer Modelling*, **43**(9-10): 1023-1037, 2006.

[4] D.N. Baker and S.P. Worden. The large benefits of Small-Satellite missions. *Transactions American Geophysical Union*, **89**(33):301, 2008.

[5] S. Damiani, H. Dreihahn, J. Noll, M. Nizette, and G.P. Calzolari. A Planning and Scheduling System to Allocate ESA Ground Station Network Services. *The Int'l Conference on Automated Planning and Scheduling*, USA, 2007.

[6] European Space Agency: http://www.esa.int/

[7] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Op. Res.*, **5**, 533–549, 1986.

[8] V. Kolici, X. Herrero, F. Xhafa, L. Barolli: Local Search and Genetic Algorithms for Satellite Scheduling Problems. *BWCCA 2013*: 328-335, IEEE CPS, 2013.

[9] A. Lala, V. Kolici, F. Xhafa, X. Herrero, A. Barolli: On Local vs. Population-Based Heuristics for Ground Station Scheduling. *CISIS 2015*: 267-275, IEEE CPS, 2015.

[10] J. Lee, S. Wang, D. Chung, K. Ko, S. Choi, H. Ahn, O. Jung, Scheduling optimization for image acquisition missions for multi-satellites via genetic algorithms, *The Korean Society For Aeronautical And Space Sciences*, 951-957, 2012.

[11] J. Noguero, G. Garcia Julian, T.W. Beech, Mission control system for Earth observation missions based on SCOS-2000, *IEEE Aerospace Conference*, 4088-4099, 2005.

[12] J. C. Pemberton, F. Galiber. A constraint-based approach to satellite scheduling. *DIMACS workshop on Constraint programming and large scale discrete optimization*, 101-114, 2000.

[13] C. Ruf, M. Unwin, J. Dickinson, R. Rose, D. Rose, M. Vincent, A. Lyons, CYGNSS: Enabling the Future of Hurricane Prediction, *Geoscience and Remote Sensing Magazine*, IEEE **1**, 52-67, 2013.

[14] W. T. Scherer, F. Rotman. Combinatorial optimization techniques for spacecraft scheduling automation. *Annals of Operations Research*, **50**(1):525-556, 1994.

[15] T.J. Schmit, M.M. Gunshor, W.P. Menzel, J.J. Gurka, J. Li, A.S. Bachmeier, Introducing the next-generation Advanced Baseline Imager on GOES-R, *Bulletin of the American Meteorological Society*, **86**, 1079-1096, 2005.

[16] Satellite Tool Kit: http://www.agi.com/products/by-product-type/applications/stk/

[17] K. Woellert, P. Ehrenfreund, A.J. Ricco, and H. Hertzfeld. Cubesats: Cost-effective science and technology platforms for emerging and developing nations. *Advances in Space Research*, **47**(4):663-684, 2011.

[18] F. Xhafa, J. Sun, A. Barolli, A. Biberaj, L. Barolli, Genetic algorithms for satellite scheduling problems, *Mobile Information Systems*, **8**(4), 351-377, 2012.

[19] F. Xhafa, X. Herrero, A. Barolli, M. Takizawa: Using STK Toolkit for Evaluating a GA Base Algorithm for Ground Station Scheduling. *CISIS 2013*: 265-273, IEEE CPS, 2013.

[20] F. Xhafa, A. Barolli, M. Takizawa: Steady State Genetic Algorithm for Ground Station Scheduling Problem. *AINA 2013*: 153-160, IEEE CPS, 2013.

[21] F. Xhafa, X. Herrero, A. Barolli, M. Takizawa: A Simulated Annealing Algorithm for Ground Station Scheduling Problem. *NBiS 2013*: 24-30, IEEE CPS, 2013.

[22] F. Xhafa, X. Herrero, A. Barolli, L. Barolli, M. Takizawa: Evaluation of struggle strategy in Genetic Algorithms for ground stations scheduling problem. *J. Comput. Syst. Sci.* **79**(7): 1086-1100, 2013.

[23] F. Xhafa, X. Herrero, A. Barolli, M. Takizawa: A Tabu Search Algorithm for Ground Station Scheduling Problem. *AINA 2014*: 1033-1040, IEEE CPS, 2014.

[24] F. Xhafa, X. Herrero, A. Barolli, M. Takizawa: A Comparison Study on Meta-Heuristics for Ground Station Scheduling Problem. *NBiS 2014*: 172-179, IEEE CPS, 2014.

[25] N. Zufferey, P. Amstutz, P. Giaccari. Graph Colouring Approaches for a Satellite Range Scheduling Problem. *Journal of Scheduling*, **11**(4):263-277, 2008.