
Xhafa, F. [et al.] (2014) A Tabu Search algorithm for ground station scheduling problem. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications: IEEE AINA 2014, 13-16 May 2014, University of Victoria, Victoria, Canada: proceedings*. IEEE. Pp. 1033-1040. Doi: http://dx.doi.org/10.1109/AINA.2014.151.

Xhafa, F. [et al.] (2014) A Tabu Search algorithm for ground station scheduling problem. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications: IEEE AINA 2014, 13-16 May 2014, University of Victoria, Victoria, Canada: proceedings*. IEEE. Pp. 1033-1040. Doi: http://dx.doi.org/10.1109/AINA.2014.151.

A Tabu Search Algorithm for Ground Station Scheduling Problem

Fatos Xhafa and Xavier Herrero
Technical University of Catalonia
Barcelona, Spain
Email: fatos@lsi.upc.edu, xharrero@lsi.upc.edu

Admir Barolli and Makoto Takizawa
Hosei University
Tokyo, Japan
EMail: admir.barolli@gmail.com, makoto.takizawa@computer.org

***Abstract*—Mission planning plays an important role in satellite control systems. Satellites are not autonomously operated in many cases but are controlled by tele-commands transmitted from ground stations. Therefore, mission scheduling is crucial to efficient 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). One common version of this problem is that of ground station scheduling, in which 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 paper we present a Tabu Search (TS) algorithm for the problem, while considering several objective functions, namely, windows fitness, clashes fitness, time requirement fitness, and resource usage fitness. The proposed algorithm is evaluated by a set of problem instances of varying size and complexity generated with the STK simulation toolkit. The computational results showed the efficacy of TS for solving the problem on all considered objectives.**

Keywords: Ground station scheduling, Satellite scheduling, Local Search, Tabu Search, STK –Satellite Simulation Toolkit.

I. Introduction

The design of intelligent mission planning for satellite systems is a long standing problem in satellite control systems. While in the past, mostly large aerospace agencies such as ESA (European Space Agency) [1], [5], [6] and NASA [3] developed mission planning systems, nowadays, missing planning is of interest to many smaller science and technology projects from research institutions and universities requiring mission planning [4], [15]. 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 gather 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 [11], tele imagery systems and earth observation [9], [13], 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 (see Table I for an example). 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], [10], [12], [17]. Therefore their resolution is tackled through heuristics approaches. For instance, Genetic Algorithms are used for both general setting [16]

Table I
AOS-LOS EXAMPLE.

GS	SC	AOS-VIS	LOS-VIS	TDur
1	1	08-FEB-2012. 12:00:00	08-FEB-2012. 13:00:00	60 min
1	1	08-FEB-2012. 14:00:00	08-FEB-2012. 15:00:00	60 min

or specific formulations such as image acquisition [8]. In this paper, we handle the resolution of the Ground-Station Scheduling using Tabu Search (TS) algorithm. TS is a local search algorithm and has shown its efficiency for solving highly complex optimization problems. The proposed TS is evaluated by a set of 48 problem instances of varying size (small, medium and large) and complexity generated with the STK simulation toolkit. The computational results showed the efficacy of TS for solving the problem on all considered objectives.

The rest of the paper is organized as follows. In Section II we describe the Ground-Station Scheduling. The different fitness types for the problem are formulated in Section III. The Tabu Search algorithm is given in Section IV and its experimental evaluation in Section V. We end the paper in Section VI with some conclusions and remarks for future work.

II. THE GROUND-STATION SCHEDULING PROBLEM

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 (see Fig. 1 for ESA Tracking Network).

Figure 1. ESA Tracking Network.

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

Table II
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 III.

Table III
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.

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

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

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

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

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 (see Fig. 2). The fitness of access window is normalized so that it's value is within 0 to 100.

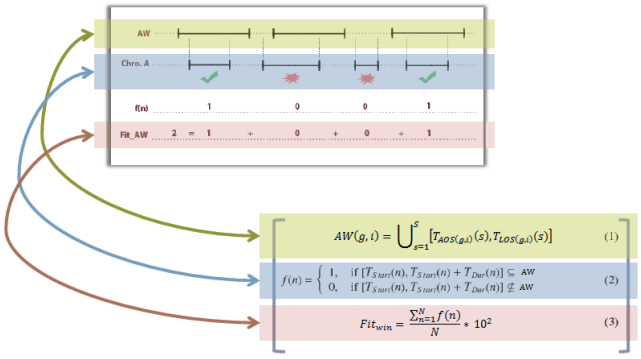

Figure 2. Access Window Fitness.

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:

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

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 (see Fig. 3 for an example). The total fitness of communication clashes is then:

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

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

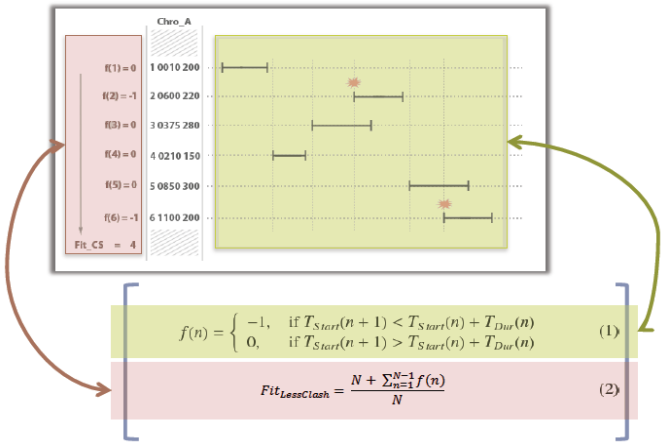

Figure 3. Fitness communication clashes.

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) and also Fig 4).

$$\begin{aligned} T_{Start}(m) &> T_{From}(k) \\ T_{Start}(n) + T_{Dur}(n) &< T_{TO}(k) \\ T_{Comm}(k) &= T_{Dur}(j) \end{aligned} \quad (5)$$
$$f(k) = \begin{cases} 1, & \text{if } T_{Comm}(k) \geq T_{REQ}(k), \\ 0 & \text{otherwise.} \end{cases}$$

$$FIT_{TR} = \frac{\sum_{k=1}^{K} f(k)}{N} \cdot 100.$$

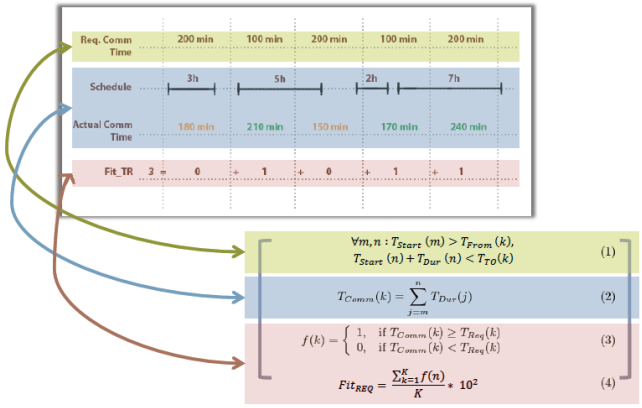

Figure 4. Fitness Requirements .

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 (see Fig. 5 for an example).

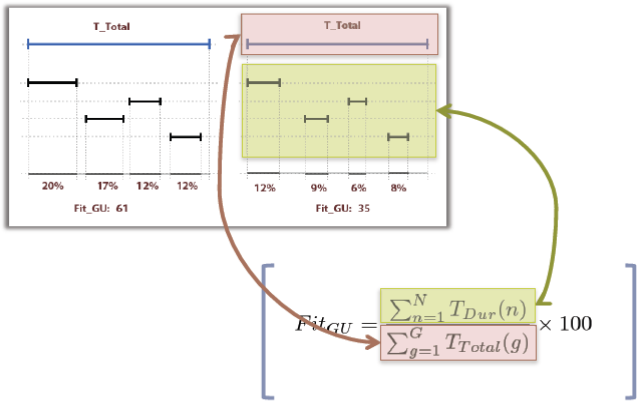

Figure 5. Ground station usage.

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.

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

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:

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

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:

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

for some λ (defined to $\lambda = 1.5$ for the experimental study).

IV. Tabu Search Algorithm

TS method [7] is a high-level local search algorithm, which uses proper mechanisms to guide the search. Unlike other local search methods such as Hill Climbing or Simulated Annealing and even population-based methods, such as Genetic Algorithms [16], its mechanisms enable to perform an intelligent exploration of the search space and avoid getting trapped into local optima. TS uses an *adaptive memory* and *responsive exploration*. The former takes decisions while exploring the neighborhood of solutions. The later enables the method to select some solutions which though might be not so good at the current search iteration could at long run lead to promising areas of good solutions in the search space.

We have used the Alg. 1 for designing the TS for Ground Station Scheduling. The inner methods implemented for the scheduling problem are described in next subsections.

A. Initial / starting solutions

The starting points in the solution space can be computed using some *ad hoc* heuristics, listed below.

- *Random First*: This method generates a solution with time intervals situated in the first half day of everyday in the specified period, that is:

$$N_d \in (0..N_{days} - 1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Start}[i] = random(1, \frac{MINPERDAY}{2}) + day * MINPERDAY$$

Algorithm 1 Tabu Search Algorithm

```
1: begin
2: Compute an initial solution s;
3: let ŝ ← s;
4: Reset the tabu and aspiration conditions;
5: while not termination-condition do
6:    Generate a subset N*(s) ⊆ N(s) of solutions such that:
7:          (none of the tabu conditions is violated) or (the
      aspiration criteria hold)
8:    Choose the best s' ∈ N*(s) with respect to the cost
      function;
9:    s ← s';
10:   if improvement(s', ŝ)) then
11:      ŝ ← s';
12:   end if
13:   Update the recency and frequency;
14:   if (intensification condition) then
15:      Perform intensification procedure;
16:   end if
17:   if (diversification condition) then
18:      Perform diversification procedures;
19:   end if
20: end while
21: return ŝ;
22: end;
```

where N_{SC} is the number of Spacecrafts, $MINPERDAY$ is a constant that indicates the amount of minutes per day.

- *Random Last*: This method generates a solution with time intervals situated in the second half day of everyday in the specified period, that is:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Start}[i] = random(\frac{MINPERDAY}{2}, MINPERDAY - 1) + \\ + day * MINPERDAY$$

- *Random Medium*: This method generates a solution with time intervals situated from one third to two third interval of everyday in the specified period, that is:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Start}[i] = random(\frac{MINPERDAY}{3}, \frac{2 * MINPERDAY}{3} + \\ + day * MINPERDAY$$

- *Random Altern*: This method generates the intervals in even position using the Random First and those in odd position using Random Last.
- *Random*: This method generates the intervals at random in the full available time of everyday in the specified period, that is:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Start}[i] = random(1, MINPERDAY - 1) + day * MINPERDAY$$

Finally, the values of $T_{Dur}[i]$ are generated based on the previously computed values assigned to T_{Start}, as follows:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Dur}[i] = random(1, MINPERDAY * (day+1) - T_{Start}[i]) + \\ + day * MINPERDAY$$

Neighborhood definition: For a solution s, the neighbourhood of s, denoted $\mathcal{N}(s)$, is defined as the set of feasible solutions reachable from s by applying a movement, as follows:

$$\mathcal{N}(s) = \{s' \mid s \xrightarrow{m} s', m \in \mathcal{M}(s), s' \in \mathcal{S}\} \quad (9)$$

where $\mathcal{S}$ is the solution space, $\mathcal{M}(s)$ is the set of movement that can be applied to s. Movements make small local perturbations to solutions yielding to a new solution (which differs little from the original one). For the coding of movement, use two structures *scheduleRow* and *resourceRow*, containing the local modification, which corresponds to the position and the modified values in the solution.

1) Tabu status: In order to avoid visiting solutions repeatedly, TS tags already visited solutions with *"tabu status"*. This mechanism can eventually break cycling among previously visiting solutions. However, the tabu status is a restrictive condition if kept unchanged over solutions for a long time. This could eventually prevent visiting good solutions during the search. Therefore, the tabu status to movements is removed if they satisfy some additional conditions known as *aspiration criteria*. In all, the set of admissible solutions to be explored in an iteration is defined as follows:

$$Admissible(s) = \{(\mathcal{N}(s) - \mathcal{T}(s)) \cup Aspiration(s)\} \quad (10)$$

where $\mathcal{T}(s)$ is the set of tabu solutions reachable from s:

$$\mathcal{T}(s) = \{s' | s \xrightarrow{m} s', s \in \mathcal{S}, \\ s' \in \mathcal{S}, m \in \mathcal{M}(s), is\_tabu(s', m) = true\} \quad (11)$$

and $Aspiration(s)$ is the set of tabu movements that satisfy aspiration criteria:

$$Aspiration(s) = \{s' | s \xrightarrow{m} s', s \in \mathcal{S}, s' \in \mathcal{S}, m \in \mathcal{M}(s), \\ is\_tabu(s', m) = true, aspirates(s', m) = true\} \quad (12)$$

It can be seen from Eq. (10) that the neighbourhood structure in TS is dynamic as the set of $Aspiration(s)$ can vary along the exploration of the neighbourhood of s.

B. Historical memory:

The historical memory is usually composed by a *short term memory* or *recency* (implemented through a tabu list and a tabu hash), with information on recently visited solutions, and a *long term memory* (or *frequency*), storing information gathered during the whole exploration process about the top best solutions.

C. Intensification and diversification procedures

These procedures are used for appropriately managing the exploration *vs.* exploitation tradeoff on the search space. The method uses long term memory to know those solution features that have most frequently appeared in solutions along the search process. In the case of intensification, most frequent features are rewarded while in case of diversification, the less frequent features are promoted (the most frequent ones are penalized) in a temporarily modified fitness function. The diversification procedure has been implemented in two variants: *soft diversification* and *trong diversification*. The former, is done by penalizing those movements that have been most occurring at solutions found along the search process and promote those that have been less occurring; the later is essentially an "escape" mechanism to restart the search at a different search area.

D. Evaluation of fitness function

The fitness function follows a simultaneous approach (see Eq. (8)), in which all objectives functions are summed up into one single objective function.

V. Computational results

A. Problem instances

The Satellite Tool Kit [14] is used to generate problem instances[1] of small, medium and large sizes (see Table IV).

Table IV
Different size instances description

Small size Instances	
Number of Ground Stations	5
Spacecrafts number	10
Number of days	10
Medium size instances	
Number of Ground Stations	10
Spacecrafts number	15
Number of days	10
Large size instances	
Number of Ground Stations	15
Spacecrafts number	20
Number of days	10

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

B. Computational results for different instance sizes

TS evaluation is studied using the set of 48 problem instances (each group consists of 16 instances). A total of 10 independent runs of the TS were performed under 400, 600 and 1000 evolution steps for each group of instances (small, medium and large –see Table IV), respectively. Averaged results are reported in Tables V, VII and IX, respectively. Execution times (averaged) are in seconds. In the table instances are denoted by **I_t_k** , where **t** is the type (**S**–Small, **M**–Medium, **L**–Large size) , and **k** the instance number (**k**=$1 \ldots 16$). The respective standard deviation values are given in Tables VI, VIII and X.

C. Evaluation

As can be seen from the computational results in Tables V, VII and IX, the TS algorithm achieved high quality solutions in very short times (even for large size instances). More precisely, from the column Fit_{win} we can observe that the algorithm could always achieve a 100% access window. Similarly, from $Fit_{TimeReq}$ columns it can be seen that the algorithm always allocated the required communication time to missions. With regard to other objectives, a good optimization was achieved overall. This can also be deduced by the fact that the number of clashes was minimized up to less than 10% (in average). Finally, with the increase in instance size, it was more challenging to maximize the usage of ground stations, although it should be mentioned that its usage was not given high priority among objectives.

From a perspective of performance and behavor, the TS algorithm performed consistently as can be confirmed by the small standard deviation values (see Tables VI, VIII and X).

VI. Conclusions and future work

In this paper we have presented the implementation and evaluation of a Tabu Search Algorithm for Ground Station scheduling problem. This problem arises in a variety of mission planning applications involving spacecrafts and ground stations, and is known for its high computational complexity. The Tabu Search method, which distinguishes for its efficiency in local search exploration, showed to effectively cope with the complexity of the problem. The computational results on a set of problem instances of varying size and complexity confirmed the good performance of the proposed algorithm. In our future work we would like to make a comparative study of different meta-heuristics for the problem, especially the performance of the local search *vs.* population-based algorithms for the problem. One potential extension of this work is the hybridization of different heuristics methods for the problem and considering *Pareto*-lile approaches for the problem.

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.

Table V
FITNESS VALUES FOR SMALL SIZE INSTANCES.

Instance	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time (s)*
I_S_01	100	86	100	75.37	609.35	15.55
I_S_02	100	97	100	66.3	610.36	15.7
I_S_03	100	83	100	78.45	609.09	15.65
I_S_04	100	87	100	70.69	609.41	15.59
I_S_05	100	87	100	74.12	609.44	15.78
I_S_06	100	84	100	77.89	609.18	15.71
I_S_07	100	88	100	73.36	609.53	15.66
I_S_08	100	79	100	72.9	608.63	15.8
I_S_09	100	95	100	69.5	610.2	15.89
I_S_10	100	91	100	71.41	609.81	15.79
I_S_11	100	82	100	77.63	608.98	15.72
I_S_12	100	86	100	72.73	609.33	15.89
I_S_13	100	83	100	77.77	609.08	15.78
I_S_14	100	85	100	75.21	609.25	15.97
I_S_15	100	85	100	68.66	609.19	15.83
I_S_16	89	93	100	79.6	555.1	14.93

Table VI
MEAN AND STD DEVIATION FOR SMALL SIZE INSTANCES.

	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time (s)*
Mean	100.00	86.57	100.00	73.33	609.39	15.77
Std deviation	0.0	4.76	0	3.69	0.45	0.12

Table VII
FITNESS VALUES FOR MEDIUM SIZE INSTANCES.

Instance	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time (s)*
I_M_01	100	86	100	62.17	609.22	53.1
I_M_02	100	89.33	100	64.98	609.58	52.58
I_M_03	100	93.33	100	62.32	609.96	52.31
I_M_04	100	92.67	100	63.66	609.9	52.52
I_M_05	100	85.33	100	57.67	609.11	52.79
I_M_06	100	87.33	100	62.53	609.36	52.58
I_M_07	100	93.33	100	63.38	609.97	52.37
I_M_08	100	90	100	56.76	609.57	52.9
I_M_09	100	90.67	100	65.69	609.72	52.78
I_M_10	100	88.67	100	56.67	609.43	53.38
I_M_11	100	90	100	63.51	609.64	52.9
I_M_12	100	90.67	100	64.95	609.72	53.07
I_M_13	100	94.67	100	61.87	610.09	52.87
I_M_14	100	94.67	100	61.45	610.08	52.53
I_M_15	100	93.33	100	63.77	609.97	52.57
I_M_16	72	96	100	80.36	470.4	50.14

Table VIII
MEAN AND STD DEVIATION FOR MEDIUM SIZE INSTANCES.

	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time (s)*
Mean	100	90.67	100	62,09	609.69	52.75
Std deviation	0	2.99	0	2.88	0.31	0.29

Table IX
FITNESS VALUES FOR LARGE SIZE INSTANCES.

Instance	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time (s)*
I_L_01	100	91.5	100	50.45	609.65	248.63
I_L_02	100	95.5	100	53.05	610.08	248.1
I_L_03	100	95.5	100	50.88	610.06	249.33
I_L_04	100	94	100	53.58	609.94	247.96
I_L_05	100	92.5	100	52.6	609.78	248.41
I_L_06	100	96.5	100	51.86	610.17	249.39
I_L_07	100	96	100	53.01	610.13	247.91
I_L_08	100	93.5	100	52.9	609.88	249.58
I_L_09	100	95.5	100	52.03	610.07	247.69
I_L_10	100	96	100	52.87	610.13	248.73
I_L_11	100	94.5	100	51.44	609.96	247.07
I_L_12	100	94	100	46.48	609.87	248.57
I_L_13	100	97	100	54.65	610.25	248.5
I_L_14	100	96.5	100	59.85	610.25	250.35
I_L_15	100	94.5	100	51.34	609.96	246.22
I_L_16	78	99.5	100	74.28	500.69	236.25

Table X
MEAN AND STD DEVIATION FOR LARGE SIZE INSTANCES.

	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time (s)*
Mean	100	94.87	100	52.47	610.01	248.43
Std deviation	0	1.56	0	2.67	0.17	1.03

[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] ESA Science & Technology. http://www.esa.int/

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

[8] 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, 2012, pp. 951-957.

[9] 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

[10] 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.

[11] 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.

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

[13] 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 (2005) 1079-1096.

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

[15] 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.

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

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