
Xhafa, F. [et al.] (2014) A comparison study on meta-heuristics for ground station scheduling problem. *2014 International Conference on Network-Based Information Systems, NBiS 2014, 10-12 September 2014, University of Salerno, Salerno, Italy: proceedings.* [S.l.]: IEEE, 2014. Pp. 172-179 Doi: http://dx.doi.org/10.1109/NBiS.2014.29.

Xhafa, F. [et al.] (2014) A comparison study on meta-heuristics for ground station scheduling problem. *2014 International Conference on Network-Based Information Systems, NBiS 2014, 10-12 September 2014, University of Salerno, Salerno, Italy: proceedings.* [S.l.]: IEEE, 2014. Pp. 172-179 Doi: http://dx.doi.org/10.1109/NBiS.2014.29.

A Comparison Study on Meta-Heuristics for Ground Station Scheduling Problem

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

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

***Abstract*—In ground station scheduling problem the aim is to compute an optimal planning of communications between Spacecrafts (SCs) and operations teams of Ground Stations (GSs). While such allocation of tasks to ground stations traditionally is mostly done by human intervention, modern scheduling systems look at optimization and automation features. Such features, on the one hand, would increase the efficiency and productivity of the mission planning systems by handling a larger number of missions, achieve a higher usage of the infrastructure (grand stations' antennae) and, on the other, would avoid error-prone human allocation and reduce human labour costs. Designing such modern, automated scheduling/planning systems is however challenging due to the highly constraint and complex nature of the problem seeking to optimize along various objectives or system parameters. In this paper we present a study on the performance of several meta-heuristics methods for solving ground station scheduling problem. Local search methods (Hill Climbing, Simulated Annealing and Tabu Search) and population-based methods (GA, Steady State GA and Struggle GA) have been considered for the study. The performance of these resolution methods was measured by a set of instances of varying size and complexity generated by STK toolkit. The study revealed the strengths and weaknesses of the considered methods while solving different size instances and considering several objective functions, namely, windows fitness, clashes fitness, time requirement fitness, and resource usage fitness.**

Keywords: Ground station scheduling, mission planning, Local Search, Genetic Algorithms, STK –Satellite Simulation Toolkit.

I. Introduction

Mission planning in satellite communications is a complex process, if the planning of the tasks are to be automated and system resources are to be optimized. Such process usually goes through different phases such as collection and analysis of information about the tasks, collecting and formulating requirements on tasks and resources, generating the *mission planning timeline* as well as providing interactive support during the mission tasks execution. The efficient generation of the mission planning timeline is at the core of the mission planning system. Indeed, given a number of resources (SCs and GSs) and a number of tasks to be allocated to GSs, there are many possible configurations, which cannot be exhaustively evaluated even for a small number of tasks and a few ground stations. This has raised the need for developing automated scheduling systems to efficienty handle the problem for practical purposes. Large aerospace agencies such as ESA (European Space Agency) [1], [5], [6] and NASA [3] account for their own mission planning systems, although highly optimized schedulers are not reported. Additionally, with the increasing needs for mission planning from emerging small projects [4], [17] while using the same infrastructure, calls for optimized solutions to attend as many as possible missing planning requests from growing number of end users and real life application scenarios [13], [11], [15].

Informally, Ground Station Scheduling consists in computing feasible planning of communications between satellites or spacecraft and operations teams of Ground Station under several constraints and requirements, such as the communication time required for each SC in a specified period of time, restrictions on the visibility of each window time for each Spacecraft Ground Station, i.e. the time at which each SC can communicate with each GS in a given time period, meeting a required communication time etc. Due to existence of window times of communication of SCs with GSs, communication clashes can be produced when two or more tasks require communication during the same window time.

From a computational complexity perspective, ground station scheduling, in their general formulations, have been shown computationally hard [2], [12], [14], [22], thus, they are intractable problems within reasonable amount of computation time. Simple heuristics methods can be applied to the problem, however, most successful approaches to highly complex optimization problems are those using meta-heuristics. Local search methods [18] (Hill Climbing, Simulated Annealing and Tabu Search) and population-based methods [19]–[21] (GA, Steady State GA and Struggle GA) have been considered for the study. The performance of these resolution methods was measured through a set of instances of varying size and complexity generated by STK toolkit.

The rest of the paper is organized as follows. In Section II we describe the Ground-Station Scheduling problem requirements and constraints. The different fitness types for the problem are formulated in Section III. The meta-heuristics algorithms considered in this study are given in Section IV

and their 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 (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 and [9]).

Figure 1. ESA Tracking Network.

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.

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

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.

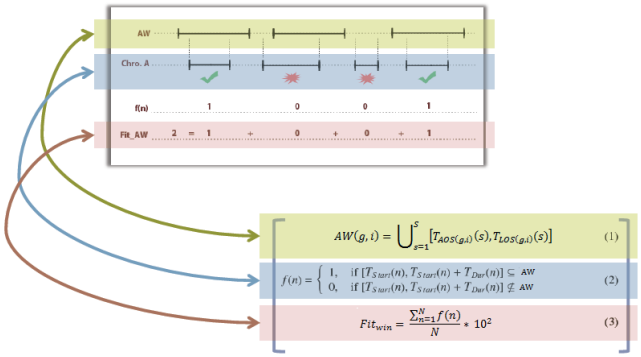

Figure 2. Access Window Fitness.

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:

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

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

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

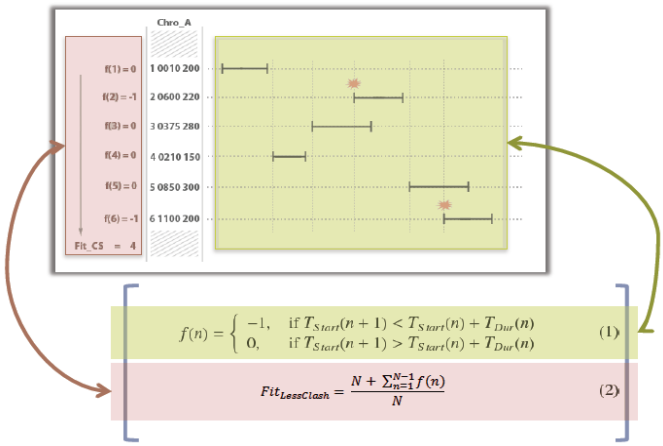

Figure 3. Fitness communication clashes.

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).

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

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

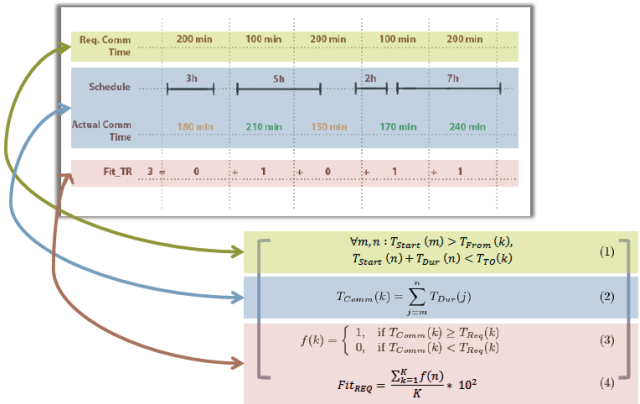

Figure 4. Fitness Requirements .

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).

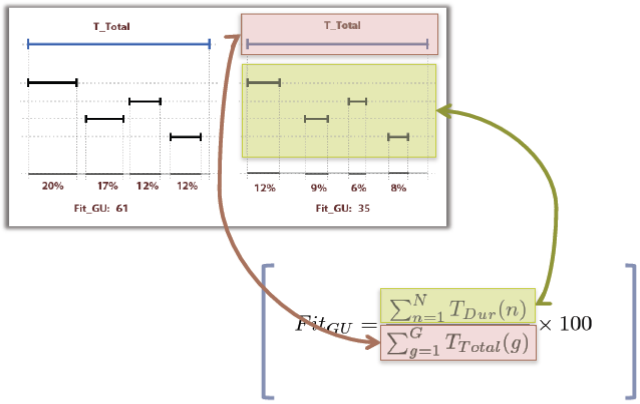

Figure 5. Ground station usage.

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. Meta-heuristics algorithms

In this section we present several meta-heuristics algorithms used in this study to solve the ground station scheduling problem. Some of them are from local search family, such as Hill Climbing, Simulated Annealing and Tabu Search and others from population-based methods such as Genetic Algorithms and its variants of Steady State GA and Struggle GA.

A. Hill Climbing Algorithm

Hill Climbing (HC) is a local search algorithm and is based on incremental improvements of solutions. 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 (see pseudo-code of HC in Alg. 1).

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

```
Generate an initial solution s_0;
s = s_0; s* = s_0; f* = f(s_0);
repeat
    Movement selection: Choose a movement m = select_movement(s);
    Evaluate & apply movement:
    if δ(s, m) ≥ 0 then
        s' = appply(m, s);
        s = s';
    end if
    Update best solution:
    if f(s') > f(s*) then
        f* = f(s');
        s* = s';
    end if
    return s*, f*;
until (stopping condition is met)
```

B. Simulated Annealing algorithm

Simulated Annealing (SA) consists of a sequence of executions of local search procedure with a progressive decrement of the temperature starting from a rather high temperature, where almost any move is accepted, to a low

temperature, where the search resembles a Hill Climbing. In SA, the solution s' is accepted as the new current solution if $\delta \leq 0$ holds, where $\delta = f(s') - f(s)$. 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 a parameter called the "temperature". 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 uses to be very slow. We have used the *SA template* of Alg. 2 in this study.

Algorithm 2 Simulated Annealing Algorithm.

```
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
```

C. Tabu Search algorithm

Tabu Search [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 [19], 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. 3 for implementing the TS for Ground Station Scheduling.

Algorithm 3 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;
```

D. Genetic Algorithms

From population-based algorithms, we have considered Genetic Algorithms and have used the *GA template* of Alg. 4 as a GA base algorithms. Then, variants of GA such as Steady State and Struggle Strategy are implemented and studied as well.

1) Steady State GA: Steady State GA is an alternative to the traditional GA approach that is based on the partial replacement (usually one or two individuals, $\lambda = 1$ / $\lambda = 2$) of the parent population, instead of the whole population. This approach was popularized by the genitor system. The idea is to reproduce one or two off-springs, evaluate their fitness and reintroduce them directly to the population and removing λ off-springs from the current population according to some criteria (population size is kept constant). The main idea in is Steady State GA is to keep a steady state population where the majority of parents are equal to their children, while very few individuals are different. By keeping such a small replacement scheme, obviously, the advantage over traditional schemes is that it requires less amount of processing and memory, but has the disadvantage that exploration is much more limited due the small number of new individuals introduced in each generation.

2) Struggle GA: In the Struggle strategy [8] a new individual replaces the individual that is most similar to it only in case the new individual obtains a better fitness value than the one to be replaced. This strategy is known for its effectiveness but suffers from a high computational cost.

Algorithm 4 Genetic Algorithm

Generate the initial population P^0 of size μ;
Evaluate P^0;
while not termination-condition **do**
  Select the parental pool T^t of size λ; $T^t := Select(P^t)$;
  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 μ from individuals in P^t and P_m^t;
  $P^{t+1} := Replace(P^t; P_m^t)$
  $t := t + 1$;
end while
return Best found individual as solution;

More precisely, given a new individual, finding a similar individual to it requires comparing against all individuals of the current generation. The definition of effective similarity functions is therefore crucial to Struggle GA. Similarity measures include *Hamming distance*, *Euclidean distance*, *Cosine distance* and *Hash-based* similarity measures.

E. 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:

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

  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.

Solution encoding: An encoding of the scheduling solution consists of (a) encoding the spacecrafts timetable and (b) encoding the pairs of GS-SC (see Fig. 6).

Chromosome A: *Chromosome B*:

$$\begin{bmatrix} SC[1], & T_{Start}, & T_{Dur} \\ SC[2], & T_{Start}, & T_{Dur} \\ \vdots & & \\ SC[i], & T_{Start}, & T_{Dur} \end{bmatrix} \quad \begin{bmatrix} SC[1], & GS[g_1] \\ SC[2], & GS[g_2] \\ \vdots & \\ SC[i], & GS[g_i] \end{bmatrix}$$

Figure 6. Chromosome representations.

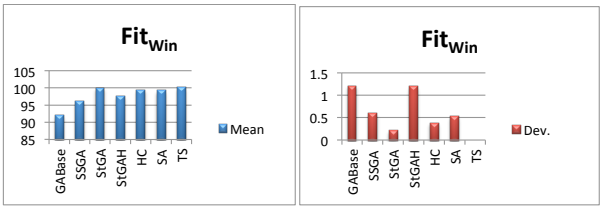

Figure 7. Window fitness for large size instances

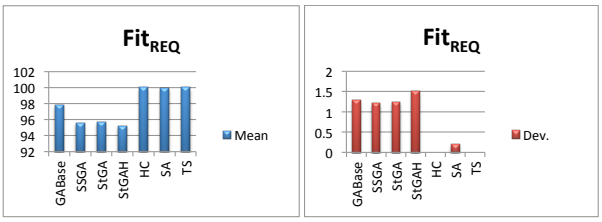

Figure 8. Requirment time fitness for large size instances

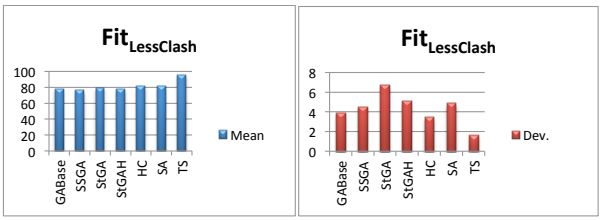

Figure 9. Clash fitness for large size instances

F. 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. Evaluation Study

A. Problem instances

The Satellite Tool Kit [16] is used to generate problem instances[1] of small, medium and large sizes (see Table III).

Table III
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

B. Computational results

A number independent runs of the meta-heuristics were performed for each group of instances (small, medium and large –see Table III), respectively, and averaged results were used to study the performance of different meta-heuristics. We present in Fig(s). 7 – 10 graphical representations for large size instances. The left hand side of the figures shows the averaged values while the right hand side shows the standard deviation. The respective averaged execution times are shown in Fig. 11.

As can be seen from Fig. 7, for the window fitness, Struggle GA and Tabu Search achieved the best results. It should be noted however that TS behaved more consistently (see standard deviation values).

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

Regarding the requirement time fitness, the three local search algorithms performed better, which showed also the smallest standard deviation values (see Fig. 8).

The best fitness clash values were achieved by Tabu Search, while the rest of methods had more or less the same performance (see Fig. 9).

Finally, the usage fitness, that is, the percentage of time the ground station was used during the mission planning, was best maximized by local search algorithms (specifically, SA performed best, see Fig. 10).

It should be noted that GA algorithm and its variants needed longer execution times to converge to good solutions, while local search algorithms were much more efficient

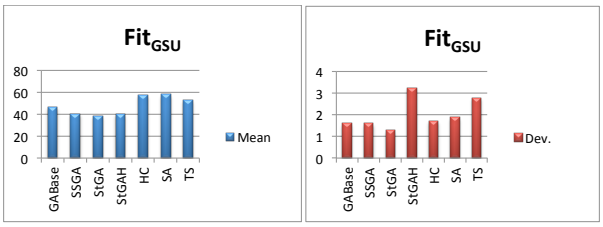

Figure 10. Usage fitness for large size instances

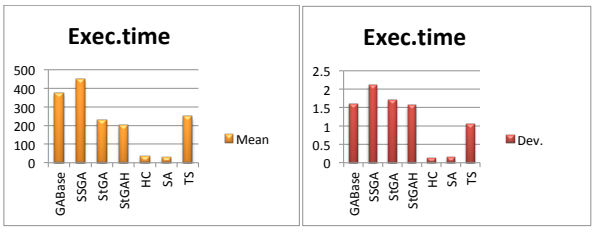

Figure 11. Execution times (in seconds) for large size instances

(see Fig. 11). Among local search algorithms, Tabu Search needed more time, clearly, but it achieved also the best results.

C. Summative evaluation

The computational results for the benchmark of instances showed that, overall, Tabu Search performed best and consistently for all instances. Indeed, it achieved a windows fitness of 100%, meaning that all tasks were planned within the specified window time, with a very small number of clashes, covering all requirement times, and with a satisfactory usage of ground station.

VI. Conclusions and future work

In this paper we have presented the results of a comparative study on the performance of several meta-heuristics methods for ground station scheduling problem, which is known for its high complexity. Local search methods (Hill Climbing, Simulated Annealing and Tabu Search) and population-based methods (GA, Steady State GA and Struggle GA) have been studied. The performance of these resolution methods was measured through a set of 48 instances of varying size and complexity generated by STK toolkit. As a result of the study, it was concluded that Tabu Search showed the best performance in terms of quality of solutions in all four optimization objectives, robustness and efficiency.

In our future work we would like to integrate these resolution methods into scheduling system and evaluate in real life scenarios of mission planning projects.

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. Niézette, 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] T. Grueninger. Multimodal optimization using genetic algorithms. Technical report. Department of Mechanical Engineering, MIT, Cambridge, MA, 1997.

[9] W. Heinen, M. Unal. Scheduling Tool for ESTRACK Ground Station Management, http://arc.aiaa.org, DOI: 10.2514/6.2010-2263 (as of on June 18, 2014)

[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, 2012, pp. 951-957.

[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 (2005) 1079-1096.

[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, X. Herrero, A. Barolli, M. Takizawa: A Simulated Annealing Algorithm for Ground Station Scheduling Problem. NBiS 2013: 24-30

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

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

[21] 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)

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