UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

UPCommons

Portal del coneixement obert de la UPC

http://upcommons.upc.edu/e-prints

This is an Accepted Manuscript of an article published by Taylor & Francis in *Enterprise information Systems* on 28/03/2019, available online: https://www.tandfonline.com/doi/abs/10.1080/17517575.2019.1593508.

Published paper:

Xhafa, F.; Ip, A. Optimisation problems and resolution methods in satellite scheduling and space-craft operation: a survey. "Enterprise information systems", 2019. doi: 10.1080/17517575.2019.1593508

URL d'aquest document a UPCommons E-prints:

https://upcommons.upc.edu/handle/2117/132523

SURVEY ARTICLE

Optimisation problems and resolution methods in satellite scheduling and space-craft operation: A survey

Fatos Xhafa[a] and Andrew W.H. Ip[b,c]

[a] Universitat Politècnica de Catalunya, Barcelona, Spain; [b] The Hong Kong Polytechnic University, China [c] University of Saskatchewan, Department of Mechanical Engineering, Canada

ARTICLE HISTORY
Compiled May 6, 2018

ABSTRACT

Nowadays, satellite technologies are developing very fast, including the production of small, low cost satellites. This is propelling an important increase in the number of satellite mission planning and operations and the need for intelligent scheduling systems to automatically and optimally handle such mission planning, i.e., the assignment of tasks to space missions through ground station services. Central to satellite mission planning is the satellite scheduling problem, whose resolution is the basis for an optimised allocation of user requests for efficient communication between operations teams at ground and space-craft systems. The aim of this paper is to survey the state of the art in the satellite scheduling problem, analyse its most significant mathematical formulations, seen as a family of time window scheduling problems, as well as examining its multi-objective nature. We particularly stress the computational complexity of the problem, its highly constrained features and the conflicting objectives due to windows accessibility and visibility clashes caused by requirements of different missions to be scheduled. In view of the computationally hardness to solve to optimality, the resolution of the problem is addressed through heuristics and meta-heuristics methods, namely, local search and population-based methods. While, for validating and evaluating the performance of resolution methods, a simulation toolkit in the literature, called STK, is used. Finally, we consider some optimisation problems arising in space-craft design, operation and satellite deployment systems.

KEYWORDS
Satellite scheduling; space-craft operation; optimisation; planning; meta-heuristics; local search; population-based methods; STK; simulation; benchmarking.

1. Introduction

Mission operations arise during the coordination of communications of space-crafts, such as satellites, space stations, probes, etc., with terrestrial ground stations. The communication of operation teams at ground with space-crafts is done by requesting an antenna at a ground station for a specific time window or multiple windows during which the task to be performed by a satellite should be completed. The number of mission requests is each time becoming larger, especially in view of increasing number

CONTACT Fatos Xhafa fatos@cs.upc.edu

of low cost small satellites, which can be afforded by institutions, small and medium size enterprises and universities for research and educational purposes. The challenge, however, in allocating such requests independently submitted by users/costumers is that different requests maybe conflicting by requesting communication with satellite(s) in the same time window, making it thus very complex to manually compute optimised allocation of requests to ground stations for a large number of missions requiring communication of space-crafts with ground stations.

Satellite Scheduling at Large Scale. The optimisation and automation of the satellite mission allocation process is commonplace for missions of European Space Agency (ESA) and NASA, which are among the largest space agencies. Thus, ESA supports its own missions, which are mainly scientific, as well as other missions upon costumers request through several own ground stations. The number of mission planning requests, which as a matter of fact keeps increasing, is larger than the number of available ground stations (Barbulescu et al. 2002), which is rather small.

ESA's ground stations comprise Malargue (Argentina), Perth and New Norcia (Australia), Redu (Belgium), Kourou (French Guiana), Santa Maria (Portugal), Maspalomas, Villafranca and Cebreros (Spain), Kiruna (Sweden). On the other hand there are a number of ESA missions, among which we could distinguish CLUSTER II, GIOVE-A/GIOVE-B, INTEGRAL, METEOSAT-6/METEOSAT-7, and XMM-NEWTON (ESA 2018). Such real life missions are also interesting in simulation phase of scheduling systems, for instance to be used in STK toolkit. ESA uses ESTRACK –the ESA tracking station network (Damiani et al. 2007). ESTRACK is a worldwide system of ground stations providing links between Operations Control Center and satellites in orbit. Current scheduling of operations for space-craft to ground stations communications has shown limitations, for instance, they are not fully automated and still need human coordination and manual labour intensive activity. Clearly, manual computations of satellite mission planning, even when partially employed, are slow, prone to errors and costly. The problem becomes more challenging with the increasing number of submitted requests.

NASA manages the EOSs fleet –Earth Observing Satellites– to assist scientific missions. Another example is the AFSCN –Air Force Satellite Control Network (more than 100 satellites and 16 antennas located at nine ground stations), supports mission planning requests by interested users/costumers (overall typically there are about a few hundreds requests spawning a hundred of time window conflicts per day) (Barbulescu, Howe, and Whitely 2006). Due to existence of such time window conflicts, users can also specify alternative time windows for their requests.

Satellite Scheduling at Small Scale. At a smaller scale, satellite mission planning arises from research projects in institutions, small and medium size enterprises and universities in Europe, USA, China, India, etc. As an example, we could refer to mission scheduling operations at Berkeley (Bester 2003). Again, these smaller scale mission planning need to automate the allocation of ground station services to space-craft missions. In fact, there is a proliferations of applications from many domains (Xuan et al. 2008a,b; Zhao et al. 2011) due to fast advancements in satellite networks (Durresi et al. 2009).

The number of ground stations is smaller compared to the number of mission planning requests, therefore the aim is to achieve a maximum usage of ground stations, which in turn would translate to support as many as possible requests for mission

planning. The ground station usage can be understood as the time during which the ground station is communicating with some space-craft. Alternatively, maximizing the ground station usage is equivalent to minimising its idle time. It should be noted that mission operations projects do have their own particularities but in essence the satellite scheduling targets their allocations to ground stations to ensure tasks completion by space-crafts.

Computational Complexity of Satellite Scheduling. Various types of satellite scheduling formulations have been reported in the literature, such as ground station scheduling (either for just one ground station or multiple ground stations), satellite range scheduling, AFSCN scheduling, LEO satellite scheduling, etc. (Pemberton and Galiber 2000; Barbulescu et al. 2004; Zufferey, Amstutz and Giaccari 2008). A foremost question is therefore if the satellite scheduling problems can be efficiently solved to optimality and thus achieve best ground stations usage and support the largest number of mission planning requests. Unfortunately, as for other time window scheduling problem, satellite scheduling formulations are NP-hard and thus unlikely to solve to optimality (Scherer and Rotman 1994). In fact, due to the high constraint nature of the problem formulations, in some cases could be challenging to even find a feasible solution (i.e. allocate all mission requests). Other scheduling models related to on-air information which uses satellite technology has also been reported in the literature (Waluyo et al. 2011).

Satellite Deployment Systems. Besides solving the satellite scheduling problems, here an important role play satellite deployment systems or simulation systems that enable evaluating the performance of the scheduling solutions. In this regard, in 2015 the Hong Kong Polytechnic University developed a micro-satellite platform and a deployment system. The micro-satellite platform and deployment system was developed to facilitate the implementation of low-cost space experiments. Not only does "Kaituo-1B' belong to the first batch of micro-satellite (CubeSat) launched by China, but it also is the very first micro-satellite platform and deployment system successfully developed by Hong Kong. It was the "20 satellites in one rocket" of "Long March 6" launched on 20 September 2015 (POLYU 2015). Compared to conventional satellites, which weigh from a few hundreds to a few thousands kilograms each, "Kaituo-1B" only weighs around two kilograms, which results in significant cost reduction in the development and production of micro-satellite for carrying small-sized payloads and instruments into space. The space technology can benefit a wide range of industries, including aviation, pharmaceutical industry, advanced materials, and educational sectors. For aviation, the platform provides additional resources to trace air traffic, reducing the difficulties of flight incident investigation. In addition, the university has been working with China Space Agency (POLYU 2017) to continue collaboration with industries and education institutes to develop more space missions for Chang'e projects (Change-3 mission 2013) for outer space explorations (the reader is referred to Sect. 3 and to (Sui-man and Yung 2006; Weiss and Lung 2009; Qian et al. 2017) for details). On the simulation side, we present a benchmark of instances generated by the Satellite Toolkit (STK), which is shown useful to evaluate and judge on the effectiveness of the resolutions methods for the problem.

In this paper, we survey the state of the art in the satellite scheduling problem, analyse its most significant mathematical formulations, seen as a family of time window scheduling problems, as well as its multi-objective nature. We particularly stress the

computational complexity of the problem, its highly constrained features and the conflicting objectives due to windows accessibility and visibility requirements of various mission planning. In view of the computationally hardness to solve to optimality, the resolution of the problem is addressed through heuristics and meta-heuristics methods, namely, local search and population-based methods. Similarly, we discuss optimisation problems arising in space-craft operation, analyse their complexity and discuss resolution methods.

The remainder of the paper is organised as follows. We present and discuss in Section 2 the basic concepts about ground stations, space-crafts/satellites and the ground station scheduling problem. Optimisation problems from space-craft operations are presented in Section 3. We discuss several resolution methods in Section 4. A useful simulation toolkit together with a benchmark of instances is presented in Section 5. We end the paper with some conclusions and future work in Section 6.

2. Satellite Scheduling Problems

In this section we present several formulation variants of satellite scheduling. We analyse in detail the case of Ground Station Scheduling in Subsect. 2.1 and refer to other formulations in Subsect. 2.2.

2.1. *Ground Station Scheduling*

We describe here the problem characteristics, identify the input problem instance and its different optimisation objectives types. Then, the output instance computed from resolution methods is formally given (refer to (Xhafa et al. 2012, 2013a,b) for more details).

Basic Concepts and Terminology. In order to formally define the problem, we give first some basics concepts and terminology about ground stations (GS), satellites and space-crafts (SC). Ground stations are terrestrial terminals designed for extra-planetary communications with space-crafts (SC). SCs are extra-planetary crafts, such as satellites, probes, space stations, orbiters, etc. GSs communicate with a space-craft by transmitting and receiving radio waves in high frequency bands and usually contain more than one satellite dish. In mission planning, a dish is usually assigned to a space mission, however, through the scheduling from control center, dishes are able to handle and switch among mission space-crafts.

Problem Instance in Input. The problem instance in input or the problem input data is the information that defines a concrete instance of the problem. In this case, the input data is given by the values of a series of parameters shown in Table 1.

Optimisation Objectives. In a general setting, scheduling problems are multi-objective optimisation problems, that is, several optimisation objectives can be established. This is also the case for the satellite scheduling for which we can formulate several optimisation objectives. Among these objectives, the most relevant ones are 1) maximizing matching of visibility windows of SCs to communicate with GSs; 2) minimizing the time window clashes of different SCs to one GS; 3) maximizing the communication time of SC with Gs, and, 4) maximizing the usage of GSs. Solving

Table 1. Problem data in input

Data	Meaning
$SC[i]$	List of SCs that participate in the planning
$GS[g]$	List of GSs that participate in the planning
N_{days}	Total number of days for the schedule
$TAOS\_VIS[i][g]$	Visibility times of GSs to SCs
$TLOS\_VIS[i][g]$	Information on timed when a GS looses signal from a SC
$TReq[i]$	List of required communication time for SCs to complete a task

an optimisation problem under various objectives is more challenging than single optimisation objective problems. In particular, multi-objective optimisation problems become more complex when there are conflicting objectives, namely, when trying to optimise an objective goes to the detriment of other objective(s).

Problem Solution in Output. Based on problem objectives, any resolution method to the problem should compute in output the values of the variables that attain the *optimised* objectives. The variables whose values are to be computed are listed in Table 2. It should be noted here that due to high computational complexity of the problem, only near-optimal solutions are expected to be computed by the resolution methods, while optimal solutions can be computed only for limited cases of small instance problems through exact resolution methods.

Table 2. Variables values computed in output

Variable	Meaning
$T_{Start}[i][g]$	Starting time of the communication between $SC[i]$ and $GS[g]$
$T_{Dur}[i][g]$	Duration time of the communication between $SC[i]$ and $GS[g]$
$SC\_GS[i]$	The list of GS assigned to $SC[i]$.
$Fit_{LessClash}$	The fitness function to minimise the collision of two or more SCs to the same GS for a given time period (range 0 to 100).
$Fit_{TimeWin}$	Fitness function to maximise time window access for every pair space-craft to ground stations $GS - SC$ (range 0 to 100).
Fit_{Req}	Fitness function measuring the satisfaction of the communication time requirement (range 0 to 100).
Fit_{GSU}	Fitness function to maximise the usage of all GS for a planning (range 0 to 100).

Multi-fitness Types and Their Combination. As stated earlier, several fitness functions can be formulated for the problem, among which the most relevant are: 1) maximizing matching of visibility windows of SCs to communicate with GSs; 2) minimizing the time window clashes of different SCs to one GS; 3) maximizing the communication time of SC with Gs, and, 4) maximizing the usage of GSs. In such case of presence of multi fitness types the issue is how to compute the total fitness function for the problem based on specific fitness functions. Likewise, one could consider that also the order into which are evaluated the fitness functions is also of relevance.

In the optimisation theory, there have been usually exploited two models: the hierarchical optimisation and the simultaneous optimisation. In the hierarchical optimisation, a priority is established for the fitness functions: $f_1 \succ f_2 \cdots \succ f_n$, namely, f_1 is

considered the most important among objectives and f_n is the least important. In the optimisation procedure, first, f_1 function is optimized, then f_2 is optimised without worsening the value of f_1 and so on with the other objectives until f_n, whose optimisation should *not* worsen any of optimised values of proceeding functions $f_1, f_2, \ldots, f_{n-1}$.

In the simultaneous optimisation all objectives are considered equally relevant and therefore all fitness functions are optimized at the same time. Whenever possible, one can reduce optimising n different fitness functions into optimisation of a single fitness function by summing them up. However, this is not always the case (for instance when fitness functions are in different range values, measure different parameters that cannot be summed together, etc.), therefore, simultaneous optimisations is approached through the general *Pareto-front* model (Chiandussi et al. 2012).

For the case of ground station scheduling we have defined four fitness function and have adopted the simultaneous optimisation approach by summing up the four particular fitness functions into a single fitness function. We describe and define next the four fitness functions and their combination in a unique fitness function for the problem.

Access Window Fitness Function. In satellite scheduling, differently from other types of scheduling, resources (space-crafts, in this case) are not available at all times but only during some time windows, called visibility windows or access windows, during which a GS can establish a communication link with a SC. In order to achieve as many as possible communications between GSs and SCs, the communication links should fall within access windows to enable the communication.

In order to formally define the corresponding fitness function, let $W_{(g,i)}$ denote the set of Access Windows for a GS g and SC i. Furthermore, let $T_{Start}(s)$ and $T_{End}(s)$ be respectively the starting and ending time of each access window. Then, we express $W_{(g,i)}$ in Eq. (1).

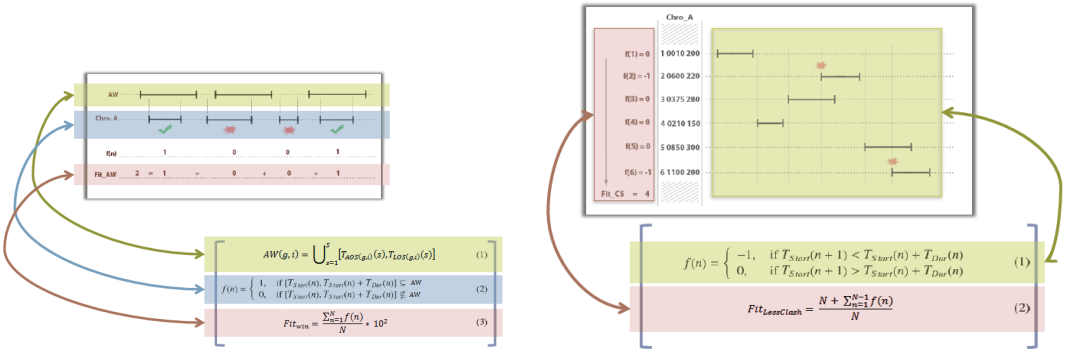
$$AW(g,i) = \cup_{s=1}^{S}[T_{AOS(g,i)}(s), T_{LOS(g,i)}(s)] \tag{1}$$

By using the expression of $W_{(g,i)}$ in Eq. (1), we calculate the total Access Window fitness of the scheduling (or mission planning) solution, denoted by Fit_{AW}, as given in Eq. (3).

$$f_{AW}(n) = \begin{cases} 1, & \text{if } [T_{Start}(n), T_{Start}(n)+ \\ & +T_{Dur}(n)] \subseteq AW(n_g, n_i), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$$Fit_{AW} = \frac{\sum_{n=1}^{N} f_{AW}(n)}{N} \cdot 100, \tag{3}$$

where n corresponds to an event number, N is the total number of events of an entire schedule, g is a GS and i a SC (a graphical representation is shown in Fig. 1(a)). Notice that the fitness of access window is normalized in the range from 0 to 100. This normalization process is useful for later summing up different fitness values.



(a) Access Window Fitness Function.

(b) Communication Clashes Fitness.

Figure 1. Access Window Fitness and Communication Clash Fitness Function.

Communication Clash Fitness Function. Every event or task of the scheduling has a starting and ending communication time. However, as requests for allocating tasks to GSs arrive independently, some classhes can happen, namely, when the starting time of one communication task, say Ta_1 happens before the end of another task Ta_2 on the same GS. Obviously, scheduling solutions that contain clashes are not feasible solutions, therefore, the objective is to minimize the number of communication clashes of different SCs to one concrete GS. In the final scheduling solution, clashes will be removed, therefore the aim is to minimise the number of clashes, which in turn would maximise the number of communications between various SCs and a GS.

In order to formally express the number of clashes, first, we sort SCs according to their starting communication time. A clash is produced when:

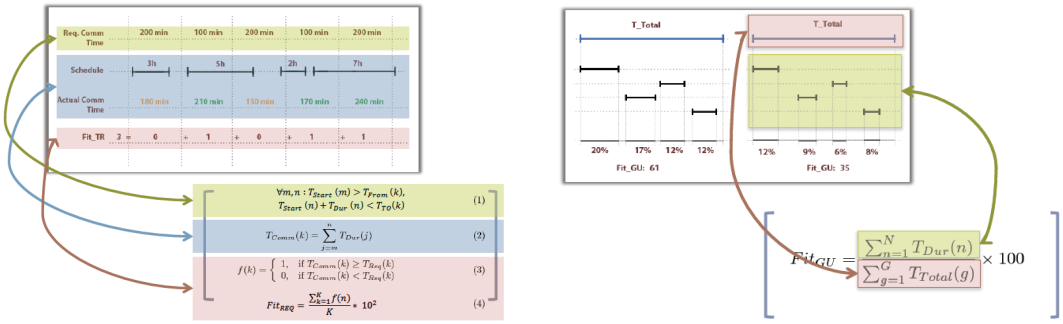
$$T_{Start}(n+1) < T_{Start}(n) + T_{Dur}(n),\ 1 \leq n \leq N-1 \tag{4}$$

where n represent an event number and N is the total number of events in the schedule. As discussed earlier, when a clash happens, to bring the scheduling to a feasible solution, the communication fitness has to be reduced by one (see Eq. (5), and one of the entries that provoked the clash is removed from the solution (a graphical representation is shown in Fig. 1(b)). We compute therefore the total fitness of communication clashes as given in Eq. (6).

$$f_{SC}(n) = \begin{cases} -1, & \text{if } T_{Start}(n+1) < T_{Start}(n) + T_{Dur}(n), \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

$$Fit_{CS} = \frac{N + \sum_{n=1}^{N} f_{SC}(n)}{N} \cdot 100 \tag{6}$$

Communication Time Requirement Fitness Function. When allocating a task to a GS, it is necessary to satisfy some minimum time requirements for TTC (Telemetry, Tracking and Command). This is particularly important for the case of tasks, referred to as data download tasks, when satellites need to download large image data



(a) Communication Time Requirements Fitness.

(b) Ground Station Usage Fitness.

Figure 2. Communication Time Requirements Fitness and Ground Station Usage Fitness Function.

usually require more time for linking/communicating with a GS. Data download tasks are usually periodical tasks (e.g. x hours communication for SC_k each day, y hours data downlink for SC_p every k days, etc.) The information of these requirements is collected in a matrix and is given in input to the scheduling system for computing an optimised solution.

Therefore, we can define a fitness function, here called FIT_{TR} (Eq. 7), aiming to maximize the communication time of SCs with GS under the requirement that each $SC(i)$ will communicate at least $T_{req}(i)$ time with a GS. This fitness function is formally expressed by summing up all the communication link durations of each SC, and dividing them in the required period to check if the allocated times according to the schedule satisfy communication time requirements (for a graphical representation see Fig 2(a)).

$$\begin{aligned} T_{Start}(m) &> T_{From}(k) \\ T_{Start}(n) + T_{Dur}(n) &< T_{TO}(k) \\ T_{Comm}(k) &= T_{Dur}(j) \\ f_{TR}(k) = \begin{cases} 1, & \text{if } T_{Comm}(k) \geq T_{REQ}(k), \\ 0 & \text{otherwise.} \end{cases} & \end{aligned} \tag{7}$$

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

Ground Station Usage Fitness Function. As indicated earlier in this paper, the number of GSs is smaller compared to the number of mission planning requests, therefore the aim is to achieve a maximum usage of GSs, which in turn would translate to support as many as possible requests for mission planning. The ground station usage can be defined as a busy time, i.e. time during which the ground station is communicating with some SC. Alternatively, maximizing the ground station usage is equivalent to minimising its idle time, when GS is not communicating with any SC. (for a graphical representation see Fig. 2(b)).

In order to formally define the Ground Station Usage Fitness Function, it is expressed as the percentage of GSs busy time by the total amount of the possible communication time between SCs and a GS. Clearly, larger usage values for a GS is used, would imply a better schedule because it would mean that a larger communication

time between SCs and a GS is achieved.

$$Fit_{GU} = \frac{\sum_{n=1}^{N} T_{Dur}(n)}{\sum_{g=1}^{NG} T_{Total}(g)} \cdot 100. \tag{8}$$

where n is the task or event number, N represents the number of tasks scheduled, NG is the number of ground stations and $T_{Total(g)}$ is the total available time of a GS.

Combination of Fitness Objectives. So far we have defined four fitness functions (Fit_{AW}, Fit_{CS}, Fit_{TR}, Fit_{GU}) covering different requirements for an efficient and optimised schedule. It should be stressed however that these fitness functions can actually be used in the design phase of the scheduler to express/formulate other fitness functions yielding to more fitness functions, which we can group into fitness modules (abbreviated by FM hereafter). On the other hand, the resulting fitness modules can be classified into two types: serial and parallel. Serial fitness modules (serial-FM) are applied in serial fashion due dependencies among them, while parallel fitness modules (parallel-FM) can be applied in parallel computation mode.

All available fitness functions in fitness modules are finally combined into a single fitness function by giving weights to different fitness modules (serial or parallel):

$$Fit = \sum_{i=1}^{n} w_i \cdot Fit_S(i) + \sum_{j=1}^{m} w_j \cdot Fit_P(j) \tag{9}$$

where n, m are the number of fitness modules, resp., w_i, w_j denote the weights assigned to fitness functions, $Fit_S(i)$ from Serial-FMs and $Fit_P(j)$ from Parallel-FMs.

For the purpose of the Ground Station Scheduling, and assuming that only the main four fitness functions (Fit_{AW}, Fit_{CS}, Fit_{TR}, Fit_{GU}) will be used, we combine them as shown in Eq. (10).

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

for some $\lambda > 0$ parameter. Clearly, the coefficients used in Eq. (10) aim to give priority to certain fitness function, for instance for some $\lambda > 1$ value [1] windows fitness would be given more priority in the schedule then other fitness functions.

2.2. *Satellite Scheduling Variants*

There are several satellite scheduling variants, defined under different satellite types and application requirements. A variety of formulations is emerging due to fast development in small satellites technology, which are being used increasingly for civil purposes such as scientific, educational and commercial projects. However, small satellites have shorter periods of communication times with ground stations. Problem formulation therefore seek to maximise total communication time between ground station

[1] We set $\lambda = 1.5$ for the experimental study reported in the references.

and satellites. The size of the problem increases as well due to increase in number of satellites and their communication constraints with ground stations.

All of the satellite scheduling variants remain computationally NP-hard to solve to optimality and therefore heuristics and meta-heuristics can be employed to tackle them. Also, they can be formulated as either single objective or multi-objective optimisation problems. We briefly describe some of them next.

Low-Earth-Orbit Satellite Scheduling. Earth observation satellite scheduling deals with the case of low orbit satellites to collect information about the Earth. (Li, Xing, Chen 2017) addressed its resolution by a hybrid online scheduling mechanism; (Wu et al. 2017) approached the problem with with an adaptive simulated annealing algorithm and a dynamic task clustering strategy. (Tangpattanakul, Jozefowiez, Lopez 2015) categorized the problem as a type of multi-dimensional knapsack problem and present multi-objective optimisation local search heuristic..

Satellite Range Scheduling. This is another variant that arises during GS operations. It aims to schedule satellite requests to GS according to their time windows so that the profit from the scheduled requests is maximized. Again, with the increasing number of requests for satellite services, the problem becomes more intractable. (Luo et al. 2017) developed a conflict-resolution technique for this variant.

Satellite Downlink Scheduling. This variant of satellite scheduling aims at optimising the communication traffic from the satellite to the GSs, which is crucial to achieve the efficiency of the system for acquiring high quality images of the Earth surface. In (Karapetyan et al. 2015), the downlink scheduling is characterised as a highly constrained problem, which is not only hard to solve to optimality but it is even challenging to produce a feasible solution.

Satellite Broadcast Scheduling. This version of the satellite scheduling is formulated for maximising for broadcasting to various GSs under certain constraints and to maximize the number of time intervals used for broadcasting. (Salman, Ahmad and Omran 2015) proposed a Differential Evolution Algorithm, which is shown to be effective and scalable.

Satellite Scheduling Data Downloads. In various application arise the need to efficiently download data from multiple satellites to a ground station network. This leads to the scheduling problem where the objective is to schedule downloads from multiple satellites to a ground station network so that the total amount of data downloaded to Earth is maximised. A greedy scheduling heuristic is presented in (Castaing 2014) and a reasonable approximation obtained by their model is shown. This kind of satellite scheduling shows its usefulness especially for the case of low cost small satellites gathering data from space, in contrast to expensive traditional satellites.

3. Space-craft Optimisation Problems

Various optimisation problems arise as well in satellite deployment systems. Such systems are particularly interesting for evaluating the performance of the scheduling

solutions. A recent micro-satellite platform and a deployment system was developed in 2015 by Hong Kong Polytechnic University, aiming to facilitate the implementation of low-cost space experiments. The space technology can benefit a wide range of industries, including aviation, pharmaceutical industry, advanced materials, and educational sectors. For aviation, the platform provides additional resources to trace air traffic, reducing the difficulties of flight incident investigation. Some research works of interest in this contexts are by (Sui-man and Yung 2006; Yung and Ko 2007; Weiss and Lung 2009; Qian et al. 2017). (Sui-man and Yung 2006) propose the use of a Function Deployment Model (FDM) for the design problem of a multi-function sampling instrument used in the ESA (European Space Agency) Beagle2 Mars Express mission. A Linear Programming (LP) optimization method was used.

(Weiss and Lung 2009) presented a decision support system for landing strategy aiming to match proposed landing sites with three different system architectures (rovers, landing stations, and impacting probes). Through the proposed methodology and based on a review on the objectives and targets, it the system can be used to design landing strategies for the robotic exploration of the Moon.

(Qian et al. 2017) a new proposed and evaluated an evolutionary algorithm for optimisation in layout of a space station.There are identified various objectives such as reducing design complexity by scaling down the computation and facilitating involvement of experts in the solving process. Also, being an evolutionary algorithm, the aim is avoiding getting stuck into local optimums and achieving good algorithm convergence. The algorithm is applied to the problem of layout design of one-space station.

(Yung and Ko 2007) presented the weight optimization of sampling drillbits, designed for European Space Agency Beagle 2 Mars Express mission.

4. Computational Complexity and Resolution Methods

As other time window scheduling problems, satellite scheduling formulations are shown to be NP-hard and thus unlikely to solve to optimality (Scherer and Rotman 1994; Pemberton and Galiber 2000; Barbulescu et al. 2004). In fact, due to the high constraint nature of the problem formulations, in some cases could be challenging to even find a feasible solution. The complexity results suggest the use of heuristics and meta-heuristics to cope with the problem's complexity for efficiently computing high quality solutions (near-optimal solutions, if not optimal) for practical applications.

The search methods is a whole family of methods, which can be classified in various ways. For instance, in Fig. 3 we can see a classification based on the type of the search (random search *vs.* enumerative search). The main difference between this two groups of methods is in efficiency and optimality. While in the first group methods distinguish for being efficient even for large size problem instances because the solution space is partially explored, in the second group large size instances would take a prohibitive time to be solved as full exploration of solutions space (of exponential size) is done. However, random search methods do not guarantee optimal solutions as enumerative methods do.

In the classification tree of Fig. 3, we stress the importance of two popular groups of methods, namely, local search methods and evolutionary search (also referred to as population-based) methods. To the former group belong methods from simple search such as Hill Climbing and Simulated Annealing to more sophisticated search methods such as Tabu Search, Variable Neighbourhood Search, Scatter Search and Path

Re-linking as well as their hybridisation yielding to higher level search methods (also known as meta-search methods or meta-heuristics). The key mechanism in these methods is neighbourhood exploration, and jumping from one feasible solution to another neighbour one until an arbitrary stopping criteria is met. As shown by many studies in the literature local search methods are very fast but often suffer from premature convergence to some local optima, where they get stuck into and are not able to escape. This kind of search is also known in the literature as "*exploitation*" process because the search is focused on a small part of the search space, i.e. a neighbourhood of the current solution. Various local search methods propose specific strategies such as tabu search or variable neighbourhood search aiming to overcome premature convergence and escaping from local optima towards finding global optimal solutions.

The other branch of search methods comprises Genetic Algorithms (GAs), Memetic Algorithms (MAs), and Evolutionary Algorithms (EAs), in general. Differently from local search methods that deal with one solution at an iteration, the evolutionary algorithms use a set of feasible solutions at the same time at any iteration, known as a population of solutions or a generation. For this reason these search methods are referred to as population-based methods. In contrast to exploitation strategy of local search, for population-based methods the search strategy is an "*exploration*" process – here a larger space comprising a set of solutions is explored. The premature convergence is an issue also for evolutionary algorithms, namely it becomes challenging to know how many generations of populations are needed to converge to optimal solution(s).

From a complexity point of view, it should be noted that while local search is intrinsically sequential and little can be done to take advantage of parallel computing, population-based methods are parallel in nature and therefore several parallel implementation variants are known for evolutionary algorithms.

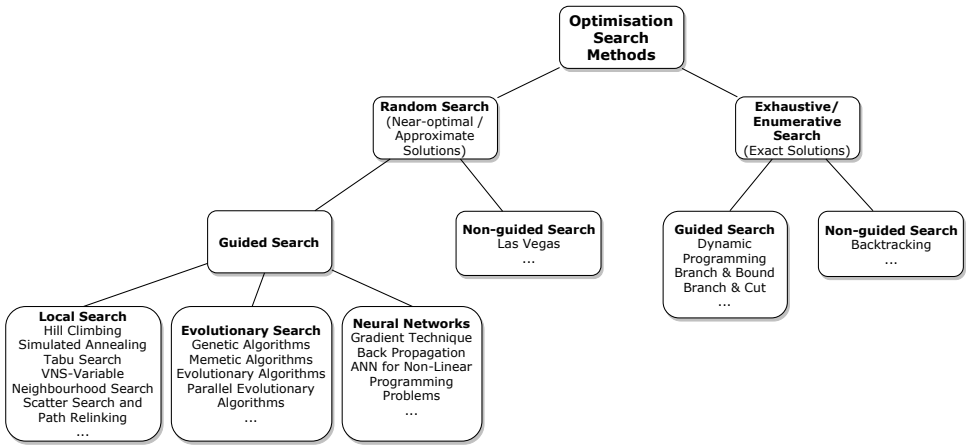


Figure 3. Classification of Search Methods.

Finally, selecting the best suited search method, i.e. local search or evolutionary search or a combination of them into hybrid search, should be addressed while solving single or multi-objective optimisation problems. A priori, there is no theoretical evidence on which of search methods is more suitable for a problem at hand, but there is an abundant literature of concrete empirical studies (see (Xhafa 2011) for a study on exploitation *vs.* exploration for distributed system scheduling, (Črepinšek 2013) for a survey of exploitation *vs.* exploration in evolutionary algorithms and (Lee and Steyvers 2008) for exploration and exploitation in decision-making).

In the following subsections we briefly present some of most paradigmatic local search and population-based search methods.

4.1. *Local Search Methods*

Local search methods distinguish for their efficiency and simplicity. They can find feasible solutions, often of high quality, in very short time. These methods are therefore suitable especially for large size problem instances, that is, when the number of mission planning requests is large and when multi-satellite / multi-ground station scheduling is to be solved.

Hill Climbing (HC) is among the simplest form of local search, while Simulated Annealing (SA) includes some mechanism known as *cooling* process to avoid premature convergence. A more sophisticated local search, yet more effective, is Tabu Search (TS) method.

4.1.1. Hill Climbing

A standard template of the Hill Climbing can be found in Alg. 1.

Algorithm 1 Hill Climbing (maximising fitness function f).

Input: Problem instance
Output: Best found solution and its fitness value

1: **Initial solution**: Compute initial solution s_0;
2: $s := s_0$; $s^* := s_0$; $f^* := f(s_0)$;
3: **repeat**
4: **Move to Next Neighbour Solution**: Compute a move $m = move(s)$;
5: **Evaluate and Apply Move**: ▶ δ function computes fitness value variation
6: **if** $\delta(s, m) \geq 0$ **then**
7: $s' := appply(m, s)$;
8: $s := s'$;
9: **end if**
10: **Update Current 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)
17: **end;**

Initial Solution. An initial solution is computed either at random or by some specific method (Xhafa et al. 2013a). The initial solution serves as the starting point of the path of solutions that HC will build along the search process.

Fitness Function Evaluation. The search process is guided by the value of the fitness function, more precisely, when a new neighbouring solution of better fitness value is computed, that new solution becomes the current solution (steps 6-9 in Alg. 1). For the case of ground station scheduling, the fitness function is defined in Eq. (10).

Neighbouring Solution Selection and Move Types. A neighbourhood of a feasible solution s, denoted $N(s)$, is the set of solutions that are reachable from s by a local move. Clearly, $N(s)$ depends on the move type and there are as many neighbourhoods as move types can be defined. One can see a move as a local small perturbation of solution s. For instance, for the scheduling, a move can be defined by swapping values of two positions in a schedule. It should be noted that no prior computation of $N(s)$ is needed, rather it is explored from one solution to another one; in fact, depending how the selection criteria is implemented, not all $N(s)$ is explored –as soon as a better solution is found the algorithm moves to the new solution and explores its neighbourhood.

Solution Acceptability Criterion. In order to move from current solution to next one, HC uses a solution acceptability criterion, which can be implemented in different ways: 1) *simple ascent* –move to newly found solution if its fitness is simply better than current fitness; 2) *steepest ascent* –move to newly found solution if its fitness is better than current fitness value among all evaluated neighbouring solutions; and 3) *stochastic* –move to a newly computed neighbouring solution at random. A simple ascent acceptability criterion is easy to implement and usually achieves reasonably good performance.

4.1.2. Simulated Annealing

Simulated Annealing (SA) generalises Hill Climbing with the aim to overcome the premature convergence to local optima. Indeed, by moving always to neighbouring solutions of better fitness, HC soon reaches to a solutions where improvements are no more possible. This new mechanism in SA is implemented via a *cooling procedure*, namely, using a temperature parameter according to which for high temperature values almost all candidate neighbouring moves can be accepted, while for low temperature values neighbouring solution selection is more restrictive. The acceptability criterion uses both the δ function and a temperature parameter T. Then, a new move is accepted with probability $\exp(-\delta/T)$ if $\delta > 0$. Clearly, the larger the temperature value T, the larger is the probability of accepting a new move. The algorithms starts with a high value of T and keeps decreasing its value systematically at each iteration of the algorithm according to an a priori established *cooling procedure*. It should therefore be noted that tuning the cooling procedure directly affects the convergence of the SA algorithm. A standard template of SA can be found in Alg. 2.

Initial Solution, Fitness Evaluation and Move Types. As SA is in essence an extended HC, with the difference of being more flexible at accepting a new solution, SA computes similarly an initial solution, uses δ function for fitness evaluation and generates new moves as explained in Subsect. 4.1.1) for HC.

Acceptability Criterion. The acceptability criterion uses an accepting threshold value. Let t_k be a succession such that $t_k > t_{k+1}$, $t_k > 0$ and $t_k \to 0$ as k tends to ∞. Intuitively, we can use t_k values to select among two solutions s_i and s_j, for instance, if $fitness\_value(s_j) - fitness\_value(s_i) < t_k$, then solution s_j is accepted.

Using this threshold idea, SA implements the acceptance criterion probabilistically as follows:

Algorithm 2 Simulated Annealing.

Input: Problem instance
Output: Best found solution

```
1: t := 0;
2: Initialize T;                                   ▶ Initial temperature value
3: s0 := Initial_Solution()
4: v0 := Evaluate(s0)                              ▶ Initial solution
5: while (not stopping condition) do
6:   while t mod MarkovChainLen = 0 do
7:     t := t+1;
8:     s1 := Generate(s0,T)                        ▶ Compute a new move
9:     v1 := Evaluate(s1)          ▶ Evaluate fitness corresponding to new move
10:    if Accept(v0,v1,T) then
11:      s0 := s1;
12:      v0 := v1;
13:    end if
14:  end while
15:  T := Update(T)          ▶ Update temperature according to cooling schedule
16: end while
17: return s0;
```

- If $fitness\_value(s_j) - fitness\_value(s_i) \leq 0$ then s_j is accepted (in this case s_j is at least as better as s_i).
- If $fitness\_value(s_j) - fitness\_value(s_i) > 0$ then s_j is accepted with probability $e^{[(fitness\_value(s_j) - fitness\_value(s_i))/t_k]}$ (in this case solution s_j is worse than solution s_i, yet it can be accepted with certain probability).

4.1.3. Tabu Search Method

Tabu Search (TS) method is among most sophisticated local search methods, introduced by Glover in various research works in '80s (Glover 1986, 1989, 1990). The aim is to enable an adaptive search and thus perform better than other local search methods. To that end, TS uses some specific mechanisms (which are *per se* heuristics) to 1) be more flexible at selection of next solutions; 2) avoid cycling among previously visited solutions; and, 3) re-launch the search at other parts of the solution space in case of getting stuck into some local optima.

A standard template of TS method can be seen in Alg. 3. There are several heuristics in the Alg. 3 that need to be implemented for a concrete problem –the ground station scheduling in our case.

Neighbourhood Exploration. For a given solution s, neighbour solutions in $N(s)$ are computed as indicated in Subsect. 4.1.1.

Short and Long Term Memory. In order to avoid cycling into previously visited solutions, TS implements two forms of memory: *short term memory* (denoted as *recency*, also referred to as *Tabu List*), which is a data structure to store features of recently visited solutions. This form memory is used by TS to discard recently visited solutions. The other memory form is *long term memory* (denoted as *frequency*),

Algorithm 3 Tabu Search

Input: Problem instance
Output: Best found solution

1: Compute an initial solution s;
2: Set current solution $\hat{s} := s$;
3: Set tabu and aspiration conditions;
4: **while** (not termination-condition met) **do**
5: Choose the best $s' \in N^*(s)$; ▶ $N^*(s) \subseteq N(s)$ s.t. (tabu conditions are *not* violated) or (aspiration criteria hold)
6: $s := s'$;
7: **if** improved$(s', \hat{s})$) **then**
8: $\hat{s} := s'$;
9: **end if**
10: Update recency memory and frequency memory;
11: **if** (intensification condition) **then**
12: intensification_procedure();
13: **end if**
14: **if** (diversification condition) **then**
15: diversification_procedure();
16: **end if**
17: **end while**
18: **return** $\hat{s}$;

which collects statistical information on solutions during the search process such as how many times a solution has been visited, when was the solution visited for the last time, etc. This later information is useful to give chances of exploration to solutions that have not been visited since a considerable number of iterations has past.

Tabu Status. A solution is tagged as *tabu* if it has been explored recently aiming to forbid its selection during next iterations unless some special conditions holds.

Aspiration criteria. These are some logical conditions that serve to the method to waive or not the tabu status of a solution.

Intensification and Diversification Conditions: The intensification condition is aimed to give priority to the search in a certain neighbourhood when there is evidence that such neighbourhood could contain high quality solutions. Intensification can be seen as *drilling* or *exploitation* mechanism. On the other hand, diversification condition aims to relaunch the search at a new area of the solution space and is activated when there is evidence on non-improvement during many iterations. Diversification can be seen as *escaping* or *exploration* mechanism.

4.2. *Genetic Algorithms*

Evolutionary algorithms constitute another large family of optimisation search methods. In this family of population-based methods, Genetic Algorithms (GAs) are the most paradigmatic (Holland 1975). Several variants of GAs have thereafter been for-

mulated including Steady-State GA, Struggle GA, etc., as well as parallel versions of them have been large studied in the literature.

A standard template of GAs is given in Alg. 4.

Algorithm 4 Genetic Algorithm

Input: Problem instance
Output: Best found solution

1: $t := 0$;
2: Compute P^0 –initial population of size μ;
3: Evaluate fitness of individuals in P^0;
4: **while** (not termination-condition met) **do**
5: Select the parental pool T^t of size λ; $T^t := Select(P^t)$;
6: Crossover pairs of individuals in T^t with probability p_c; $P_c^t := Cross(T^t)$;
7: Mutate individuals in P_c^t with probability p_m; $P_m^t := Mutate(P_c^t)$;
8: Evaluate fitness of individuals in P_m^t ;
9: Let P^{t+1} be the new population of size μ from individuals in P^t and/or P_m^t;
10: $P^{t+1} := Replace(P^t; P_m^t)$
11: $t := t + 1$;
12: **end while**
13: **return** Best found solution;

The particularity of GA for the ground station scheduling is briefly described next (full details of implementing GA and its variants for the ground station scheduling can be found in (Xhafa et al. 2012, 2013b)).

Genetic Encoding –The Chromosome. The foremost step in implementing a GA is the chromosome structure, that is, genetic encoding of individuals. Indeed the successful implementation of GAs depends on the ability to efficiently encode and transmit the genetic information, along an a priori number of generations, from parents to offsprings during the search process. The chromosome structure has also a direct impact on the efficiency of the GA implementation as it serves as a basis for implementing genetic operators. In the GA for ground station scheduling it is useful to consider two chromosomes (referred to as *ChromosomeA* and *ChromosomeB* in Fig 4). ChromosomeA uses binary encoding while ChromosomeB uses decimal encoding to encode, for every SC, the starting time and duration of communication with SC.

Fitness Function. Fitness function again is the one defined in Eq. (10)).

Selection Operators. An array of selection operators can be implemented for GAs in order to find the most suitable one. For the case of ground stations scheduling the selection operators such as Linear Ranking Selection, Best Selection and Tournament Selection were implemented according the so-called *implicit fitness re-mapping* technique.

Crossover Operators. These operators are the most important in GAs due to the fact that they directly influence the efficiency of transmission of genetic information. By taking in input two parent chromosomes, a crossover operator produces new offspring(s). The premise is that newly generated individuals contain the best genetic

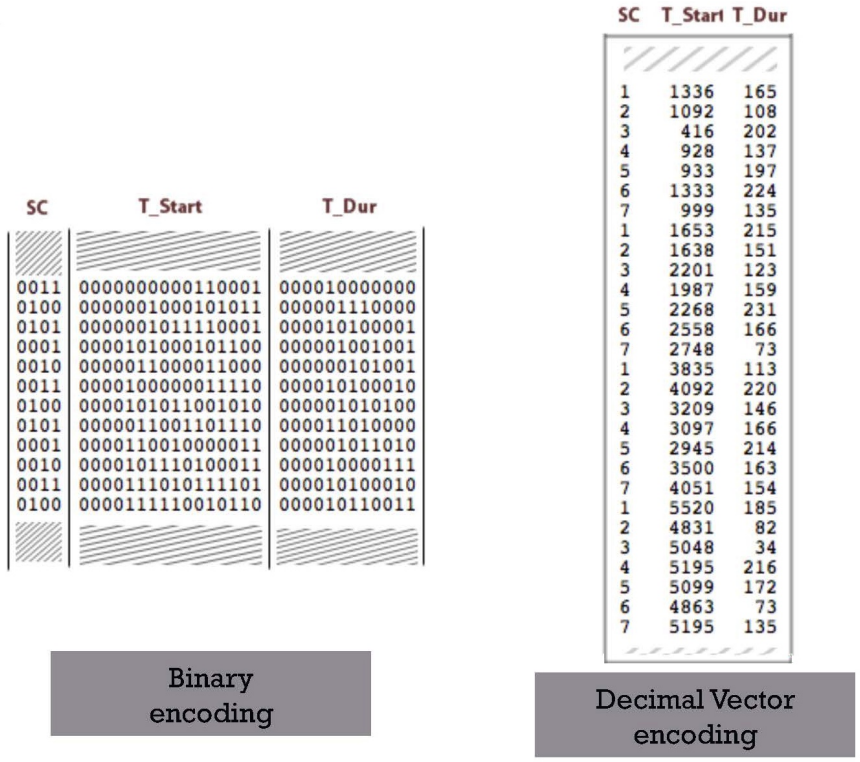


Figure 4. Genetic encoding: ChromosomeA (left) and ChromosomeB (right) (Xhafa et al. 2013b).

features of parents and thus the search progresses towards better individuals (problem solutions). The chromosome representations therefore play a crucial role. It should be also noted that an effective implementation of crossover operators should achieve maintaining the diversity of populations and thus contribute to good convergence of the algorithm.

Mutation Operator. Together with crossover operators, mutation operators aim at efficient transmission of genetic information along generation of the search process. Likewise, mutation operators help maintaining population diversity and contribute to avoiding premature convergence. Mutation operators can be implemented for both binary encoding (ChromosomeA) and decimal encoding (ChromosomeB).

5. Benchmarking and Simulation Platforms

Given that heuristics search methods do not provide any guaranty for computing optimal solution, evaluating their performance becomes of utmost importance in order to find the most suitable search methods for the problem at hand. As every problem has its own particularities and its (unknown) fitness landscape, performance evaluation is useful to know how much a search method can take advantage of problem combinatorics. In this context, the development of benchmarking and simulation tools is crucial to hand to researchers and developers with means for such performance evaluation as well as for reproducing the experimental studies under the same or different parameter setting. Examples of successful benchmarks in the field are test functions

for various search methods.

Motivated by the need of a benchmark of instances, we used the simulator STK toolkit to generate a benchmark of instances (Xhafa et al. 2013a) consisting of different instance sizes (small, medium, large). The instances are formulated in XML to ease their use in reading from various programming environments. Besides, the aim of the benchmark is to capture real features of the problem. By evaluating the search methods presented in previous sections, the benchmarking showed their useful for the experimental evaluations.

Satellite Tool Kit (STK). STK[2] is an analytical tool developed at Analytical Graphics, Inc. as a general purpose simulator for complex and integrated analysis of land, sea, air and space. We have used STK to generate a set of static problem instances for the ground station scheduling problem, which is then used for validating the various resolution methods as well as their performance. While using STK we have selected real satellites/space-crafts and ground stations as part of instances information.

5.1. *XML Instance Format*

A simulation in STK generates different CSV files. Instead, we used an XML format for structuring the instances information, which facilitates efficient reading and processing of instances from various programming environments.

Listing 1 shows the XML instance definition. The description of parameters used in XML instance are given in Table 3.

Listing 1 XML Instance Definition

```
<?xml version="1.0" encoding="utf-8"?>
<problem>
<!--  Basic problem information -->
<basic nGS="" nSC="" nDays="" />
<!-- Satellite visibility -->
<visibility>
<timewindow GS="" SC="" tAos="" tLos="" />
</visibility>
<!-- Satellite communication requirements -->
<requirements>
<communication SC="" tBeg="" tEnd="" tReq="" />
</requirements>
</problem>
```

5.2. *The Design of Problem Instances*

Aiming to cover the problem complexity, a total of 48 simulation scenarios of small, medium and large sizes were considered and for each of them a static problem instance was generated[3]. The characteristics of the instances are given in Table 4.

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

[3]Benchmark of instances can be accessed at: http://www.cs.upc.edu/∼fatos/GSSchedulingInputs.zip

Table 3. Parameter description of XML instance format

TAG	**TAG mother**	**Description**	**Attributes**	
			Name	Description
Problem	-	Principal TAG	-	-
Basic	Problem	Basic Problem Information	nGS nSC nDays	Number of GSs Number of SC Number of days
Visibility	Problem	Visibility List of GS-SC.	-	-
timeWindow	Visibility	Visibility list element	GS SC tAos tLos	ID of the GS ID of the SC Starting visibility time Ending visibility time
Requirements	Problem	List of required communication time of SCs.	- -	- -
Communication	Requirements	Communication list element	SC tBeg tEnd tEnd tReq	ID of SC Starting time of communication requirement Ending time of communication requirement Required communication time

Table 4. Instance characteristics description

Instance size	**Number of Ground Stations** (nGS)	**Number of Space-crafts** (nSC)	**Number of days** (nDays)
Small	5	10	10
Medium	10	15	10
Large	15	20	10

6. Conclusions and Future Work

In this paper we have surveyed some relevant formulations of satellite scheduling problem arising in mission planning of communicating space-crafts with terrestrial ground stations. The problem is in itself a whole family of time window scheduling problems, is attracting attention to the community in the field due to the fast development of satellite technologies for producing low-cost small satellites. The availability of more affordable satellites is propelling various research and educational projects at institutions and universities, which, in turn, generate each time more satellite mission planning requests, given the rather small number of ground stations. Various satellite scheduling problems are formulated and their complexity is discussed. Then, several resolution methods, namely heuristics methods, are presented to tackle with complexity and highly constrained nature of the problem variants. Likewise, we stressed the importance of solving optimisation problems arising in space-craft design, operation and satellite deployment systems.

The development of integrated tools and platforms for optimisation, simulation, design and satellite deployment systems is an interesting field of future research work.

Acknowledgement

The work in the paper was partially supported by a grant from the Department of Industrial and Systems Engineering of the Hong Kong Polytechnic University (H-ZG3K).

References

Barbulescu, Laura, Howe, Adele E., Watson, Jean-Paul, Whitley, Darrell. 2002. Satellite Range Scheduling: A Comparison of Genetic, Heuristic and Local Search. *Parallel Problem Solving from Nature –PPSN*, VII: 611-620, 2002.

Barbulescu, Laura, Watson, Jean-Paul, Whitley, Darrell and Howe, Adele E. 2004. Scheduling space-ground communications for the air force satellite control network. *Journal of Scheduling*, 7(1), 7-34.

Barbulescu, Laura, Howe, Adele E., Whitley, Darrell. 2006. AFSCN scheduling: How the problem and solution have evolved. *Mathematical and Computer Modelling*, 43(9-10): 1023-1037.

Bester, Manfred. 2009. Automated multi-mission scheduling and control center operations at UC Berkeley. *IEEE Aerospace conference*. DOI: f10.1109/AERO.2009.4839694

Castaing, Jeremy. 2014. Scheduling Downloads for Multi-Satellite, Multi-Ground Station Missions *28th Annual AIAA/USU Conference on Small Satellites*.

Change-3 mission. 2013. Chinese Academy of Sciences, China National Space Administration, The Science and Application Center for Moon and Deepspace Exploration.

Damiani, Sylvain, Dreihahn, J. Noll, Nizette, Marc, Calzolari, Gian Paolo. 2007. A Planning and Scheduling System to Allocate ESA Ground Station Network Services. *The Int'l Conference on Automated Planning and Scheduling*, USA.

Durresi, Arjan, Durresi, Mimoza, Barolli, Leonard, Xhafa, Fatos. 2009. MPLS Traffic Engineering for Multimedia on Satellite Networks. *Journal of Mobile Multimedia*, Vol. 5, No. 1, 3-11.

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

Chiandussi, Giorgio, Codegone, Marco, Ferrero, Simone and Varesio, Stefano. 2012. Comparison of multi-objective optimization methodologies for engineering applications. *Comput. Math. Appl.* 63, 5, 912-942. DOI: http://dx.doi.org/10.1016/j.camwa.2011.11.057

Glover, Fred. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*. 13 (5): 533-549. DOI: 10.1016/0305-0548(86)90048-1.

Glover, Fred. 1989. Tabu Search – Part 1. *ORSA Journal on Computing*. 1 (2): 190-206. DOI: 10.1287/ijoc.1.3.19010.1287/ijoc.1.3.19010.1287/ijoc.1.3.190.

Glover, Fred. 1990. Tabu Search Part 2. *ORSA Journal on Computing*. 2 (1): 4-32. DOI: 10.1287/ijoc.2.1.4.

Holland, John H. 1975. Adaptation in Natural and Artificial Systems. MIT Press

Karapetyan, Daniel, Mitrovic Minic, Snezana, Malladi, Krishna T., Punnen, Abraham P. 2015. Satellite downlink scheduling problem: A case study. *Omega*, Vol. 53, 115-123.

Ko, Sui-man and Yung, Kai-Leung. 2006. Function Deployment Model for Continuous and Discontinuous Innovation Product Development. *International Journal of Innovation and Technology Management*, Vol. 3, No. 11, 107-128, World Scientific Co.

Lee, Michael D. and Steyvers, Mark. 2008. Final Report: Exploration and Exploitation in Structured Environments. *Report* of AFOSR sponsored research (FA9550-09-1-0082), Department of Cognitive Sciences University of California, Irvine, USA.

Li, Guoliang, Xing, Lining, Chen, Yingwu. 2017. *Acta Astronautic*a, Vol. 140, 2017, 308-321.

Luo, Kaiping, Wang, Haihong, Li, Yijun, Li, Qiang. 2017. High-performance technique for satellite range scheduling. *Computers & Operations Research*, Vol. 85, 2017, 12-21.

Pemberton, Joseph C. and Galiber,III, Flavius. 2000. A constraint-based approach to satellite scheduling. *DIMACS workshop on Constraint programming and large scale discrete optimization*, 101-114.

POLYU. 2015. Media Releases. PolyU Participates in Development of Microsatellite Platform and Deployment System [Online], 10th December.

POLYU. 2017. Media Releases. The Hong Kong Polytechnic University- The SPace Exploration Journey [Online], 24th November.

Qian, Zhiqin, Bi, Zhuming, Cao, Qun, Ju, Weiguo, Teng, Hongfei, Zheng, Yang and Zheng, Siyu. 2017. Expert-guided evolutionary algorithm for layout design of complex space stations. *Enterprise Information Systems*, 11:7, 1078-1093.

Salman, Ayed A., Ahmad, Imtiaz, Omran, Mahamed G.H. 2015. A metaheuristic algorithm to solve satellite broadcast scheduling problem. *Information Sciences*, Vol. 322, 72-91.

Scherer, William T. and Rotman, Frank. 1994. Combinatorial optimization techniques for space-craft scheduling automation. *Annals of Operations Research*, 50(1):525-556.

Črepinšek, Matej and Liu, Shih-Hsi and Mernik, Marjan. 2013. Exploration and Exploitation in Evolutionary Algorithms: A Survey *ACM Comput. Surv.* 45(3).

Tangpattanakul, Panwadee, Jozefowiez, Nicolas, Lopez, Pierre. 2015. A multi-objective local search heuristic for scheduling Earth observations taken by an agile satellite. *European Journal of Operational Research*, Vol. 245:2, 542-554.

Waluyo, Borgy, A., Rahayu, Wenny, Taniar, David, Srinivasan, Bala. 2011. A Novel Structure and Access Mechanism for Mobile Broadcast Data in Digital Ecosystems. *IEEE Transactions on Industrial Electronics*, 58(6): 2173 - 2182.

Weiss, Peter and Yung, Kai Leung. 2009. Mission architecture decision support system for robotic lunar exploration. *Planetary and Space Science*, Vol. 57:12, 1434-1445.

Wu, Guohua, Wang, Huilin, Pedrycz, Witold, Li, Haifeng, Wang, Ling. 2017. *Computers & Industrial Engineering*, Vol. 113, 2017, 576-588.

Fatos Xhafa, Bernat Duran, Joanna Kolodziej, Leonard Barolli, Makoto Takizawa. 2011. On Exploitation vs Exploration of Solution Space for Grid Scheduling. *Third International Conference on Intelligent Networking and Collaborative Systems (INCoS 2011)*, 164-171, IEEE Computer Society

Xhafa, Fatos, Sun, Junzi, Barolli, Admir, Biberaj, Aleksander, Barolli, Leonard. 2012. Genetic algorithms for satellite scheduling problems, *Mobile Information Systems*, 8(4), 351-377.

Xhafa, Fatos, Herrero, Xavier, Barolli, Admir and Takizawa, Makoto. 2013. Using STK Toolkit for Evaluating a GA Base Algorithm for Ground Station Scheduling. *CISIS 2013*: 265–273, IEEE CPS.

Xhafa, Fatos, Barolli, Admir, and Takizawa, Makoto. 2013. Steady State Genetic Algorithm for Ground Station Scheduling Problem. *IEEE 27th International Conference on Advanced Information Networking and Applications (AINA-2013)*, IEEE CPS, 153-160.

K. Xuan, G. Zhao, D. Taniar, B. Srinivasan. 2008. Continuous Range Search Query Processing in Mobile Navigation, *Proceedings of the 14th International Conference on Parallel and Distributed Systems* (ICPADS 2008), IEEE, pp: 361-368.

Xuan, Kefeng, Zhao, Geng, Taniar, David, Safar, Maytham, Srinivasan, Bala. 2011. Voronoi-based multi-level range search in mobile navigation. *Multimedia Tools Appl.*, 53(2): 459-479.

Yung, Kai-Leung and Ko, Sui-man. 2007. Weight optimization of sampling instruments for ESA Mars Express mission. *Engineering Computations*, Vol. 24 Issue: 1, 97-109.

Zhao, Geng, Xuan, Kefeng, Rahayu, Wenny, Taniar, David, Safar, Maytham, Gavrilova, Maria L., Srinivasan, Bala. 2011. Voronoi-Based Continuous k Nearest Neighbor Search in Mobile Navigation, *IEEE Transactions on Industrial Electronics*, 58(6): 2247-2257.

Zufferey, Nicolas, Amstutz, Patrick, Giaccari, Philippe. 2008. Graph Colouring Approaches for a Satellite Range Scheduling Problem. *Journal of Scheduling*, 11(4):263-277.