18^{th} International Conference on Space Operations, Montreal, Canada, 26 – 30 May 2025.

SpaceOps-2025 ID #214

A Scalable Schedule Optimization Solution for Satellite Constellations

Benjamin Altenstein[1], Nicholas Symons[2], Michael Schmeing[3]*

[1] *CGI Deutschland B.V. & Co. KG, Mornewegstr. 30, 64293 Darmstadt, Germany, benjamin.altenstein@cgi.com*
[2] *CGI Deutschland B.V. & Co. KG, Mornewegstr. 30, 64293 Darmstadt, Germany, nicholas.symons@cgi.com*
[3] *CGI Deutschland B.V. & Co. KG, Alte Wittener Straße 56, 44803 Bochum, Germany, michael.schmeing@cgi.com*
*Corresponding author

Abstract

Satellite constellations play an important role in today's economy and society, as they provide a wide range of services that are essential to our lives. Satellite constellations consisting of multiple satellites are capable of performing many tasks that individual satellites are unable to accomplish, such as global satellite navigation, worldwide high-speed, low-latency communication, as well as earth observation services with high spatial resolution and continuous observation of selected target regions.

In satellite constellation operations, scheduling decisions have to be made for several spacecraft and ground stations at a time. As the resources available in a satellite constellation are shared, these decisions interact with one another, even if they impact different spacecraft or ground stations. Therefore, there are many degrees of freedom in the operation of such constellations, and thus many potentially feasible schedules. Employing advanced optimization methods to this problem can lead to significant cost savings and improvements in the fulfillment of mission objectives, while making more efficient use of the constellation's resources. However, these schedule-optimization problems have a large number of parameters, and their feasible solution regions are split into many disconnected subregions, making them not trivially solvable. Several greedy, heuristic and constructive algorithms have been proposed for solving similar scheduling problems. However, these approaches have only been able to find suboptimal solutions.

In this paper, the inherent structure of the constellation scheduling problem is exploited by dissecting it into two subproblems, one of which can be efficiently solved and, in that process, greatly reduces the remaining search space to be considered. A novel schedule optimization algorithm is proposed that implements this approach by combining a genetic algorithm that produces distinct scheduling solutions using a schedule builder, with convex optimization techniques. The proposed algorithm deterministically finds local optima, but is also capable of escaping them, allowing fast convergence to the global optimum of the problem. It is highly parallelizable and efficiently finds near-optimal, feasible scheduling solutions for large satellite constellations within reasonable computing times. A wide range of cost functions is supported, allowing for application of the algorithm to several mission profiles, as is required to operate different types of satellite constellations. Schedule optimization problems involving satellite constellations consisting of several hundred spacecraft, like for example the upcoming European IRIS^2 constellation, have been successfully solved utilizing an implementation of this novel algorithm in CGI's mission planning solution, Pleniter® Plan. This paper discusses the proposed algorithm's performance and adaptability in a variety of example use cases and in several simulated satellite constellation mission scenarios.

Keywords: mission planning, satellite constellations, schedule optimization

1 Introduction

Satellite constellations play a critical role in enabling global communications, earth observation, and navigation systems. Efficient planning of constellations is essential to minimize costs and ensure robust service quality across diverse regions. Especially in the context of sustainability, efficient usage of available resources is an important factor. In this paper, a satellite constellation planning algorithm is introduced which exhibits several favorable properties such as being able to work with a wide variety of cost functions, finding global optima while escaping local ones and being highly parallelizable to achieve performance suitable also for solving planning problems for larger constellations. The algorithm is integrated into CGI's mission planning solution Pleniter® Plan which currently operates the German Federal satellite mission "Heinrich Hertz". A general overview of the concepts of Pleniter® Plan was presented at SpaceOps 2023, see Schmeing and Symons [10], and an overview of the Heinrich Hertz mission can be found in Böcker, Biggins, and Schmeing [2].

This paper is based on and closely follows the work of Altenstein [1]. All figures are taken from this work and more technical details and explanation can be found there.

1.1 Outline

This paper is organized as follows. After discussing related work, we first introduce the problem domain in Section 2. We give an extended example and formalize the problem description. We then describe our contributions in the form of a novel constellation planning algorithm in Section 3. Results are described in Section 4 and Section 5 gives a conclusion and an outlook to future work.

1.2 Related Work

Constellation planning has been investigated by several authors before. In Cho et al. [4], for example, the problem was formulated as two linear programs (assuming smaller constellations with less than 100 spacecraft). The first program maximizes the total time for communication between ground and spacecraft. The resulting contacts are then taken as constraints for the second program, which maximizes the number of tasks executed within their given time budget. Both linear programs enforce exclusive satellite and ground station usage, as well as orbital visibility. The work of Parish [9] aims to find the best sequence of all possible task sequences. For evaluation of the quality of a sequence, a simple first-fit heuristic was used to select each task's start time. This way, each task is in a greedy way scheduled at the first point in time where it meets all its constraints (if possible). The quality score of a specific sequence is then given by the number of successfully scheduled tasks. To guide the search in the space of all possible sequences, a genetic algorithm is utilized.

In general, constellation planning algorithms have been proposed for constellations with specific objectives, such as earth observation, etc. However, since satellite constellations have very different objectives, instead of focusing on one of these objectives, we present in this paper a more generalized approach which can be applied to a wider range of satellite constellations.

2 Optimizing Constellation Planning

The main type of constraint for satellite constellations is visibility between ground stations and satellites. Typically, the number of ground stations is the limiting resource, so the main objective of a constellation planning facility is to optimize the usage of ground stations. For a set of tasks $\mathcal{T}$ which require visibility to ground stations let t_i denote its start times and d_i its durations. For each satellite and each ground station there exists a visibility window $[s_{ij}, e_{ij}]$ with start time s_{ij} and end time e_{ij}. The start and end times denote the earliest and latest time when a direct communication between satellite and ground station can happen. Several factors contribute to these times, starting from the general visibility of the ground station area from the satellites as computed by flight dynamics, the local conditions at the ground station sites given by their horizon masks and the ramp-up and ramp-down times that ground stations require to slew to and lock on a satellite. The visibility windows $[s_{ij}, e_{ij}]$ are assumed to denote the "net" times, i.e., the time windows that allow for a communication between ground and space and therefore the execution of the task after considering all these effects.

The ith task can be scheduled if it fits into a visibility window for the jth ground station, i.e.:

$$s_{ij} \leq t_i \leq e_{ij} - d_i, \tag{1}$$

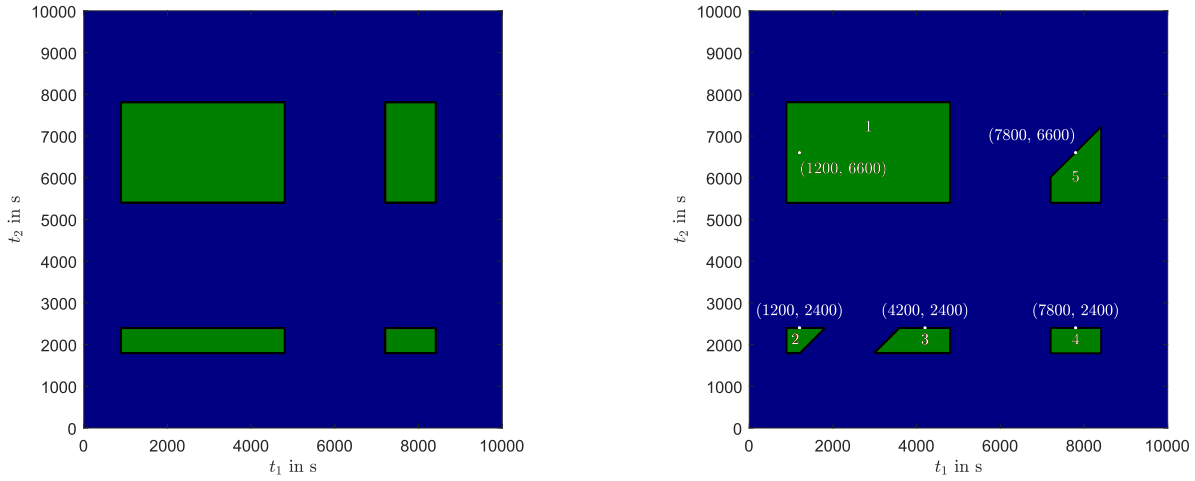
A schedule is therefore feasible when the following sets of conditions are met, i.e., if for each task start time t_i at least one visibility window (and with that implicitly a ground station) can be found such that

$$\begin{gathered}
\underbrace{(s_{0_0} \leq t_0 \leq e_{0_0} - d_0)}_{\text{Visibility Window 0}} \vee \underbrace{(s_{1_0} \leq t_0 \leq e_{1_0} - d_0)}_{\text{Visibility Window 1}} \vee \ \ldots \ \vee \underbrace{(s_{j_0} \leq t_0 \leq e_{j_0} - d_0)}_{\text{Visibility Window } j_0} \vee \ \ldots, \\
(s_{0_1} \leq t_1 \leq e_{0_1} - d_1) \vee (s_{1_1} \leq t_1 \leq e_{1_1} - d_1) \vee \ \ldots \ \vee (s_{j_1} \leq t_1 \leq e_{j_1} - d_1) \vee \ \ldots, \\
\vdots \\
(s_{0_i} \leq t_i \leq e_{0_i} - d_i) \vee (s_{1_i} \leq t_i \leq e_{1_i} - d_i) \vee \ \ldots \ \vee (s_{j_i} \leq t_i \leq e_{j_i} - d_i) \vee \ \ldots, \\
i \in \mathcal{T}, \quad j_i \in \mathcal{V}_i,
\end{gathered} \tag{2}$$

where $\vee$ denotes a logical OR.

In addition to the constraint that each task in $\mathcal{T}$ must be associated with a visibility window, a typical further set of constraints enforces that no two tasks share a visibility to the same ground station at the same time. This is due to the fact that a ground station antenna typically can only connect to a single satellite at a time: even if a second satellite would be in view in principle, communication can only happen with one satellite at a time.

Two tasks with start times t_k and t_l and respective durations d_k and d_l belong to the same *exclusive set* $\mathcal{E}$ if they require the same resource, e.g., a ground station antenna in this context. These tasks cannot



(a) Feasible regions as defined in Equations 4 and 5 are shown in green.

(b) Feasible regions as defined in Equations 4, 5 and 6 are shown in green. In each of the five subregions, an example point with its coordinates is shown.

Figure 1: Feasible Regions for different problems.

overlap, i.e. the following constraints must be fulfilled:

$$\underbrace{(t_k + d_k < t_l)}_{k \text{ before } l} \vee \underbrace{(t_k > t_l + d_l)}_{k \text{ after } l}, \quad k, l \in \mathcal{E} \subseteq \mathcal{T}. \tag{3}$$

Finally, its is assumed that tasks cannot be split into subtasks and that all tasks shall be scheduled. The following sections gives an example to illustrate this situation.

2.1 Example

Let's consider a simple situation with two tasks with start times t_1 and t_2. The duration of the first task is $d_1 = 600\,\text{s}$ and the duration of the second is $d_2 = 1200\,\text{s}$. The two tasks are constrained in the following way:

$$(900\,\text{s} \leq t_1 \leq 5400\,\text{s} - 600\,\text{s}) \vee (7200\,\text{s} \leq t_1 \leq 9000\,\text{s} - 600\,\text{s}) \tag{4}$$

$$(1800\,\text{s} \leq t_2 \leq 3600\,\text{s} - 1200\,\text{s}) \vee (5400\,\text{s} \leq t_2 \leq 9000\,\text{s} - 1200\,\text{s}) \tag{5}$$

Each point in Figure 1a corresponds to a set of start times for the two tasks. Points in the green regions correspond to start times that fulfill both constraints in equations 4 and 5.

Figure 1b shows the feasible regions when the two tasks are additionally constrained by an exclusion constraint as defined in Equation 6:

$$(t_1 + 600\,\text{s} < t_2) \vee (t_2 + 1200\,\text{s} < t_1) \tag{6}$$

Here, the set of feasible points consists of five convex feasible subregions. For each of the five points shown in Figure 1b, the respective schedule is shown as a Gantt chart in Figure 2. The white regions surrounded by thin black lines correspond to feasbile points while the black bars correspond to tasks that start at the start points indicated in Figure 1b with their respective durations.

So far, there is no difference between the feasible points in the green feasible regions in Figure 1b. All correspond to feasible schedules but we have not defined any difference between them. The goal of a scheduling algorithm is usually not only to find just a feasible schedule but also one that is optimal in a given sense. This is expressed via a function $f(\vec{t})$ that evaluates on all schedules. Figure 3 shows the values for the simple cost function $f(t_1, t_2) = \frac{1}{2}(t_1^2 + t_2^2)$: The marked points in Figure 3 depict for each feasible subregion the point with the lowest cost. Since the five subregions are convex and the cost function is quadratic with a positive definite Hessian, the unique optimal points (if they exist) can be found within each region with linear programming methods such as *active set*. Finding the globally best feasible schedule is then reduced to the problem of iterating through all feasible subregions, finding the unique optimal point (if existing) and take the global optimum of all these points.

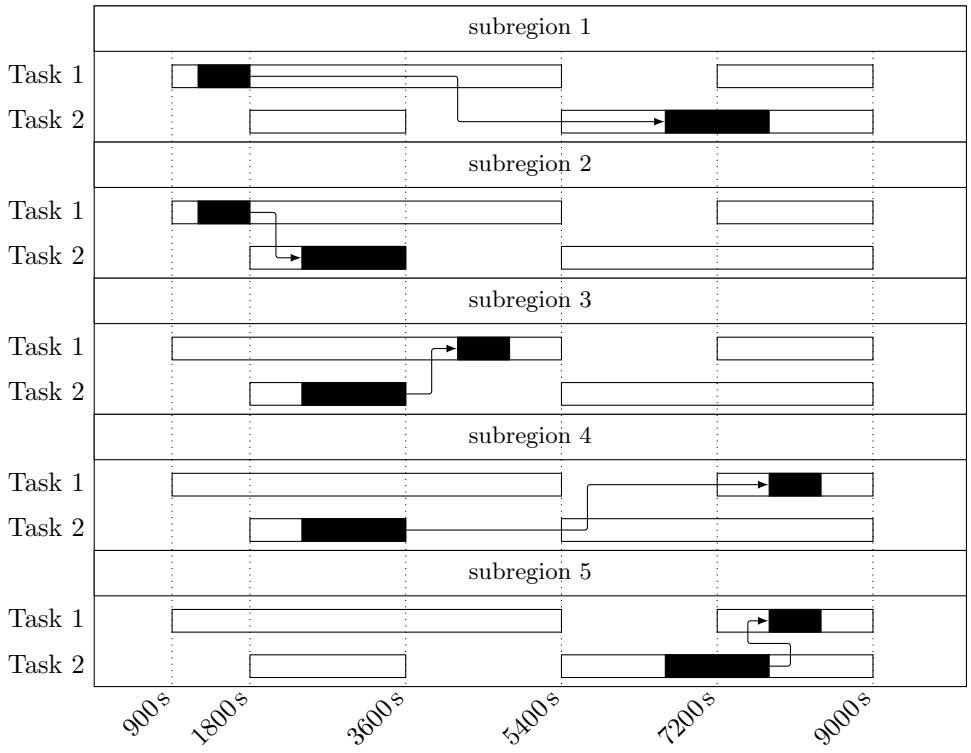


Figure 2: Gantt chart representation of the schedules corresponding to the five points indicated in Figure 1b

.

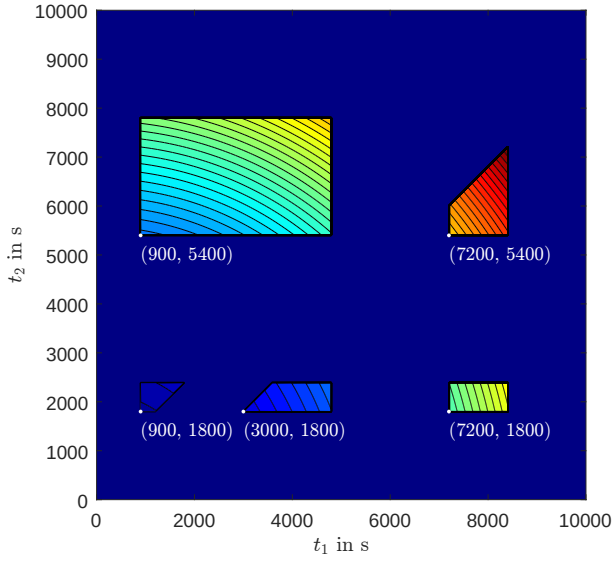


Figure 3: The five feasible subregions described by Equations 4, 5 and 6 shown with cost function $f(t_1, t_2) = \frac{1}{2}(t_1^2 + t_2^2)$. Color scale is from red to blue, denoting high to low cost, respectively.

2.2 Formal Definition

We assume quadratic cost functions of the following form:

$$\min_{\vec{t}} f(\vec{t}) = \frac{1}{2}\vec{t}^{\top}\mathbf{P}\vec{t} + \vec{t}^{\top}\vec{q} \tag{7}$$

with the Hessian $\mathbf{P}$ and the linear coefficient column vector $\vec{q}$ of $f(\vec{t})$. Equation 7 is constrained by the following inequality and equality constraints, respectively:

$$\mathbf{A}\vec{t} \leq \vec{b} \tag{8}$$

$$\mathbf{C}\vec{t} = \vec{h}, \tag{9}$$

$$\vec{t} \geq \vec{0}.$$

Cost functions of this form can be solved with standard methods like the active set method, see Nocedal and Wright [8]. The active set methods iteratively finds an optimum point by maintaining a set of "active" constraints for an optimum point, i.e., all equality constraints as well as those inequality constraints that are exactly met at the optimum point (inequality becomes equality). The active set method requires a feasible (but not necessary optimal) point as input. To identify whether a given problem has a feasible point, a simplified linear program (called *Phase I problem* by Nocedal and Wright [8]) of the problem is solved with the *simplex method*. This way, either a feasible point is found or it is identified that the problem does not contain a feasible point. Details can be found in Altenstein [1] and in Nocedal and Wright [8].

Note that also a linear combination of different cost functions $f_i(\vec{t})$ is suitable to be solved by the active set method as long as at least one of the Hessians $\mathbf{P}_i$ is positive definite:

$$f(\vec{t}) = \sum_{i=1}^{n} \alpha_i f_i(\vec{t}) \tag{10}$$

$$= \frac{1}{2}\vec{t}^{\top}\sum_{i=1}^{n}\alpha_i\mathbf{P}_i\vec{t} + \vec{t}^{\top}\sum_{i=1}^{n}\alpha_i\vec{q}_i, \quad a_i \in \mathbb{R}^{+}, \tag{11}$$

where each $f_i(\vec{t})$ is constrained by equality and inequality constraints, respectively, analogous to Equations 8 and 9.

To give a better understanding of the type of problems that can be modeled with cost functions of the type of Equation 7, we present in the following three examples.

2.2.1 Smallest time between all pairs of start times

$$f(\vec{t}) = \frac{1}{n(n-1)}\sum_{i=0}^{n}\sum_{j=i+1}^{n}(t_i - t_j)^2 \tag{12}$$

This cost function minimizes the average start time between tasks and with that ensures that all tasks are clustered as much as possible, resulting in a shorter schedule. An example use case is an on-board software update for every satellite of the constellation where the time where there are different software versions deployed in the constellation should be made as short as possible.

This function can be written in the following way in the form of Equation 7:

$$f(\vec{t}) = \frac{1}{2}\vec{t}^{\top}\underbrace{\begin{bmatrix} \frac{2}{n} & -\frac{2}{n(n-1)} & \cdots & -\frac{2}{n(n-1)} \\ -\frac{2}{n(n-1)} & \frac{2}{n} & \cdots & -\frac{2}{n(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{2}{n(n-1)} & -\frac{2}{n(n-1)} & \cdots & \frac{2}{n} \end{bmatrix}}_{\mathbf{P}}\vec{t}, \quad \vec{q} = \vec{0}. \tag{13}$$

2.2.2 Distance to specified start time

$$f(\vec{t}) = \frac{1}{n}\sum_{i=1}^{n}(t_i - z_i)^2 \tag{14}$$

This cost function minimizes the average distance of start times to the specified start times z_i for all or some tasks. An example use case is an earth observing constellation that is supposed to take images from ground locations at the best possible resolution. To achieve this, the z_i would represent the times of lowest height of each satellite over their target region so that the images captured by the constellation are on average captured when the satellites are closest to the requested ground location, maximizing their resolution.

This function can be written in the following way in the form of Equation 7:

$$f(\vec{t}) = \frac{1}{2}\vec{t}^{\top} \underbrace{\begin{bmatrix} \ddots & & \\ & \frac{2}{n} & \\ & & \ddots \end{bmatrix}}_{\mathbf{P}} \vec{t} + \vec{t}^{\top} \underbrace{\begin{bmatrix} \vdots \\ -\frac{2}{n} z_i \\ \vdots \end{bmatrix}}_{\vec{q}} + \frac{1}{n} \sum z_i^2 \tag{15}$$

2.2.3 Distance of selected start times close to specified duration

$$f(\vec{t}) = \frac{1}{n} \sum_{i,j} (t_i - t_j - y_{ij})^2 \tag{16}$$

This cost function aims to space the start times t_i and t_j of pairs of tasks so that their distance is close to a specified value y_{ij}. An example use case is the periodical downlink of satellite telemetry which should happen regularly and evenly spaced for each satellite.

This function can be written in the following way in the form of Equation 7:

$$f(\vec{t}) = \frac{1}{2}\vec{t}^{\top} \underbrace{\begin{bmatrix} \ddots & & & & \\ & \frac{2}{n} & \cdots & -\frac{2}{n} & \\ & \vdots & \ddots & \vdots & \\ & -\frac{2}{n} & \cdots & \frac{2}{n} & \\ & & & & \ddots \end{bmatrix}}_{\mathbf{P}} \vec{t} + \vec{t}^{\top} \underbrace{\begin{bmatrix} \vdots \\ -\frac{2}{n} y_{ij} \\ \vdots \\ \frac{2}{n} y_{ij} \\ \vdots \end{bmatrix}}_{\vec{q}} + \frac{1}{n} \sum y_{ij}^2 \tag{17}$$

3 Proposed Genetic Algorithm

In the previous section, we have described the type of cost function that we support in this work as well as the structure of the region of feasible points (via the example given in Section 2.1). Solving a constellation planning problems means finding a minimum point for a given cost function. For cost functions of the type described in Equation 7, this in general cannot be done by applying the active set method like described in Section 2.2, because the inclusion/visibility constrains create a non-convex solution space. Active set only works with a convex solution space. In addition, for real world problems with a sufficiently high number of satellites and constraints, the problem size may quickly become infeasible.

For this reason, we propose a two-phase approach by separating the optimization in two phases. The general idea is to apply the active set method only on convex subregions. Here, an optimal point can be quickly found when the region contains a feasible point, which is identified as described in Section 2.2 for this type of cost function. Instead of brute-forcing through all potential subregions, we employ a selection phase based on an genetic algorithm that selects the next subregion to be fed to the active set method finding optimal points in the subregions. The active set method then feeds back information to the selection phase which continues to iterate trough the problem by selecting the next subregion. Figure 4 shows an overview of the proposed algorithm.

3.1 Genetic Algorithms

Genetic algorithms mimic how the evolution found global optima like humans or pandas: specimen are subject to a selection process ("survival of the fittest") and evolve via mechanisms such as mutation or crossover. Genetic algorithms are probabilistic algorithms and can find approximate solutions even for non-linear, high-dimensional problems. If tuned correctly, they can also avoid getting stuck in local optima, see Brucker and Knust [3] and Mitchell [6].

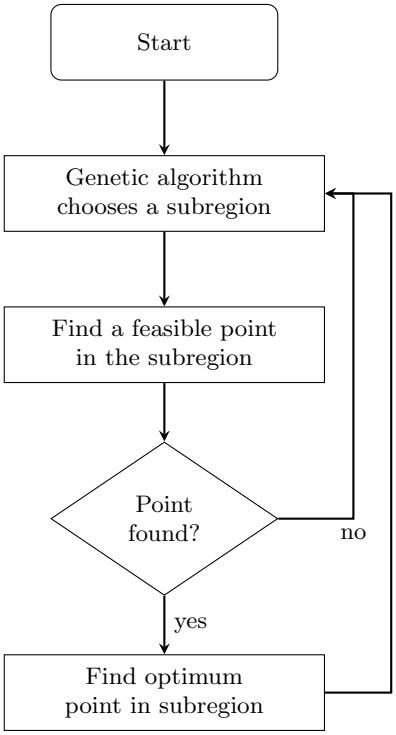


Figure 4: Proposed algorithm.

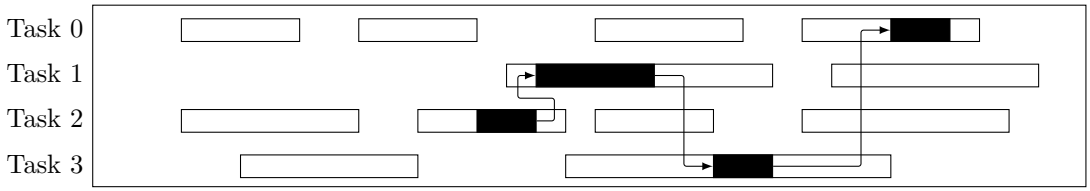
3.2 Our Algorithm

In the following, we describe how we employ a genetic algorithm to select the next subregion to be solved by the active set method. We use SPEA-2 (see Zitzler, Laumanns, and Thiele [11]) as basis for our genetic algorithm. SPEA-2 is able to solve problems with multiple objectives by creating a variety of possible solutions. It uses selection to identify the best solutions from each generation, keeping a balance between high-quality results and maintaining diversity in the population. Through crossover, it combines properties from two parent solutions to create child solutions with potentially better properties. Mutation introduces small random changes to solutions, ensuring a broad search of possibilities and avoiding getting stuck in local minima. The process is iterative and is stopped when a given number of iterations is reached or if the cost function achieves a specified value.

In the following, we describe what the respective genetic concepts map to in our problem domain: specimen, selection, crossover and mutation.

Visibility Windows j_i				Task Sequence			
3	0	1	1	2	1	3	0

(a) A genome representation of a problem domain with 4 tasks. Task 0 is in visibility window 3 of its possible visibility windows (i.e., in the fourth one), task 1 is in its visibility window 0 and so on. Regarding the exclusion constraints, task 2 is the first, followed by task 1, task 3 and task 0. Figure 5b shows a Gantt chart representation of this genome.



(b) The Gantt chart representation of the specimen shown in Figure 5a.

.

Figure 5: Example genome of a specimen.

Visibility Windows j_i				Task Sequence			
3	0	1	1	2	1	3	0

(a) Parent 1 genome.

Visibility Windows j_i				Task Sequence			
2	1	3	0	0	1	2	3

(b) Parent 2 genome.

Visibility Windows j_i				Task Sequence			
3	1	3	1	0	1	3	2

(c) Child 1 genome.

Visibility Windows j_i				Task Sequence			
2	0	1	0	0	1	2	3

(d) Child 2 genome.

Figure 6: Crossover with two parent genomes and two child genomes. Entries with gray background in the child genome originate from the second parent (or are adapted via partially mapped crossover).

3.2.1 Specimen

Specimen represent convex parts of the feasible region, i.e., subregions. The genome of a specimen, i.e., its "textual" representation, consists of two parts, the first encoding the visibility windows that each task is taking and the second the order of the tasks as shown in Figure 5a.

Since specimen will undergo crossover and mutation, which are merely textual operations on them, we need to be able to generate a subregion from a given specimen, i.e., the constraints analogous to Equations 1 and 3, for which then the optimal point is found via the active set method (if the region contains a feasible point at all). The corresponding subregion is created in the following way:

1. For each task, take the two constraints regarding the start and end time of its corresponding visibility window j_i (see also Equation 1):

$$s_{j_i} \leq t_i \tag{18}$$
$$t_i + d_i \leq e_{j_i}. \tag{19}$$

2. Based on their corresponding visibility window, determine the exclusive sets, i.e., tasks that share the same resource and therefore cannot overlap.

3. For all tasks in such an exclusive set, determine the overlapping corresponding visibility windows.

4. For each set of overlapping corresponding visibility windows, create exclusion constraints between the participating tasks. However, instead of creating exclusion constraints between all pairs of tasks, it is sufficient to only create exclusion constraints between succeeding tasks according to the task sequence encoded in the second part of the genome:

$$t_k + d_k < t_l. \tag{20}$$

5. The subregion is then defined by the constraints from Equations 18, 19 and 20.

3.2.2 Selection

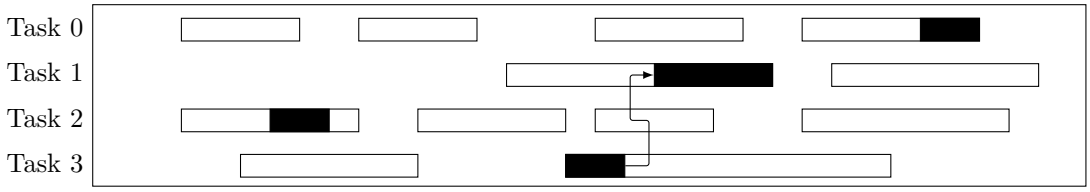
Via the selection step, genetic algorithms decide which specimen survive. We use the binary tournament selector offered by the jMetal framework, see Nebro, Durillo, and Vergne [7]. For this, two specimen are randomly selected and the one with the lower value of $f(\vec{t})$ (lower is better) for the corresponding subregion is selected. If the subregion of one specimen contains no feasible point, the other is selected. If the subregions of both specimens do not contain a feasible point, the specimen with the lower number of violated constraints in the corresponding phase 1 linear program is selected.

Visibility Windows j_i				Task Sequence			
3	0	1	1	2	1	3	0

(a) Original genome.

Visibility Windows j_i				Task Sequence			
3	0	0	1	2	3	1	0

(b) Mutated genome. Mutated locations are denoted with gray background.



(c) Gantt chart representation of the mutated genome.

Figure 7: Genome mutation

3.2.3 Crossover

Crossover of genomes works in the following way. Two parent genomes are taken as input and result in two child genomes. For the first half of a genome that encodes the visibility windows, crossover is implemented as two-point crossover: the child genome is randomly separated into two areas of the same size. The first child then gets one part filled by genome from one parent and the other part of the genome filled from the other parent. For the other child, the process works vice-versa. For the second half of the child genomes, which is a sequence of indices specifying the order of tasks, the partially mapped crossover operator, described in Eiben and Smith [5], is used to avoid duplicates in the child genomes. Figure 6 illustrates the result of one crossover step.

3.2.4 Mutation

Mutation of genomes works in the following way. For the first half of a genome that encodes the visibility windows, one index is randomly changed to a legal value. For the second part of the genome, two randomly chosen entries are swapped. Figure 7 illustrates the result of one mutation step.

4 Results

We investigated the performance of the algorithm for three different constellations.

4.1 Constellations

The first constellation is a small Walker delta pattern constellation with notation $56° : 5/5/1$ and semimajor axis a of 29 599 801 m. For each spacecraft, a single task is scheduled. The following cost function is used:

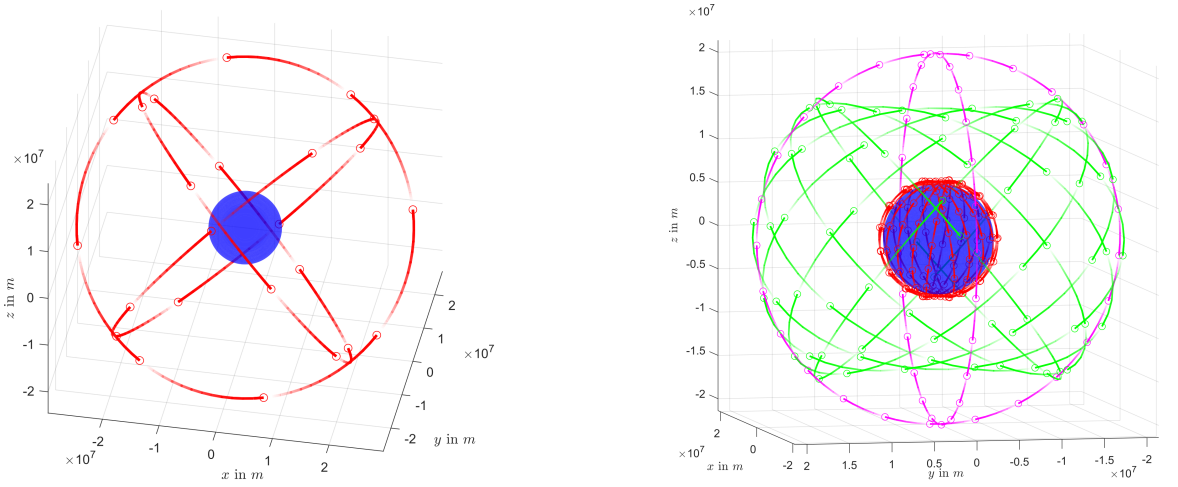
$$f(\vec{t}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} (t_i - t_j)^2 + \frac{1}{100} \frac{1}{n} \sum_{i=1}^{n} t_i^2. \tag{21}$$

This cost function favours equal start times and start times equal to zero. Visibility with ground stations is enforced and each ground station can only talk to one spacecraft at a time.

The second constellation is the Galileo reference constellation with notation $56° : 24/3/1$ and a semimajor axis a of 29 599 801 m. For each spacecraft, three tasks with durations 60 min, 90 min and 150 min are scheduled and ground station visibility is enforced. As cost function, Equation 21 is used. The constellation is visualized in Figure 8a.

The third constellation is a stitched Walker constellation and consists of 270 satellites in multiple shells and is based on the general concept of the IRIS^2 constellation. The inner shell has notation $80° : 150/10/1$ at $a = 500\,000$ m. The second shell has notation $50° : 70/7/1$ at $a = 15\,000\,000$ m and the third shell has notation $90° : 50/2/1$ also at $a = 15\,000\,000$ m. For each satellite, a single task of 5 minutes duration is scheduled and ground station visibility is enforced. The cost function is again Equation 21. The constellation is visualized in Figure 8b.

For all constellations, we use the six Galileo ground stations Kiruna, Kourou, Noumea, Reunion, Redu and Papeete.



(a) Galileo reference constellation.

(b) Complete stitched Walker constellation.

Figure 8: Analyzed large constellations.

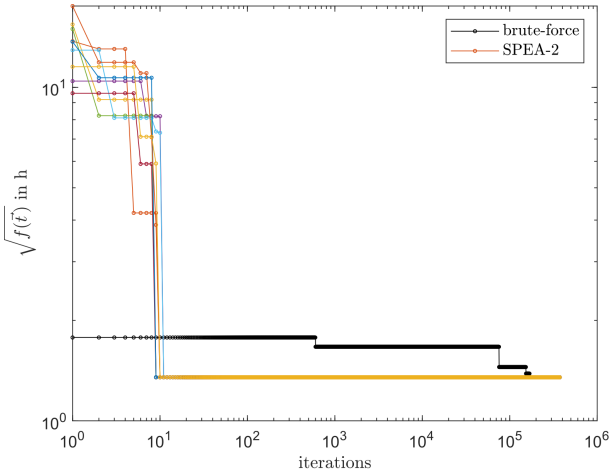


Figure 9: Development of solution quality over time for a small Walker delta pattern constellation. Exhaustive search (black) and multiple runs of proposed algorithm (other colors). Each point shows the cost of best solution found after the specified number of iterations. Exhaustive search eventually converges to optimal solution. Our algorithm finds it significantly quicker in every run.

4.2 Discussion

With the first, small constellation, we investigate whether the proposed algorithm is able to find an optimal solution. The constellation is so small that an exhaustive, brute force search is actually feasible, allowing for a comparison with the proposed algorithm. Figure 9 shows the results for several runs of our algorithm. It always finds the optimal solution and is significantly quicker in doing so than the brute force approach.

The planning problems for the Galileo and the stitched Walker constellation are too big to be completely solved with a brute force approach. Figure 10 shows the temporal development for the solution quality for the Galileo problem. It can again be seen that the solution quality, i.e., the value of the cost function (lower is better) improves quickly using our proposed algorithm. The longer it runs, though, the better the found solution. After 24 000 iterations, the square root of the solution of the proposed algorithm is 7.75(9) h and 30.92(25) h for the brute force approach. Figure 11 shows how the solution quality and the number of constraint violations evolves over time (no comparison with brute force is shown). The algorithm quickly finds a feasible solution while the solution quality continues to improve. A longer runtime would probably result in a better solution.

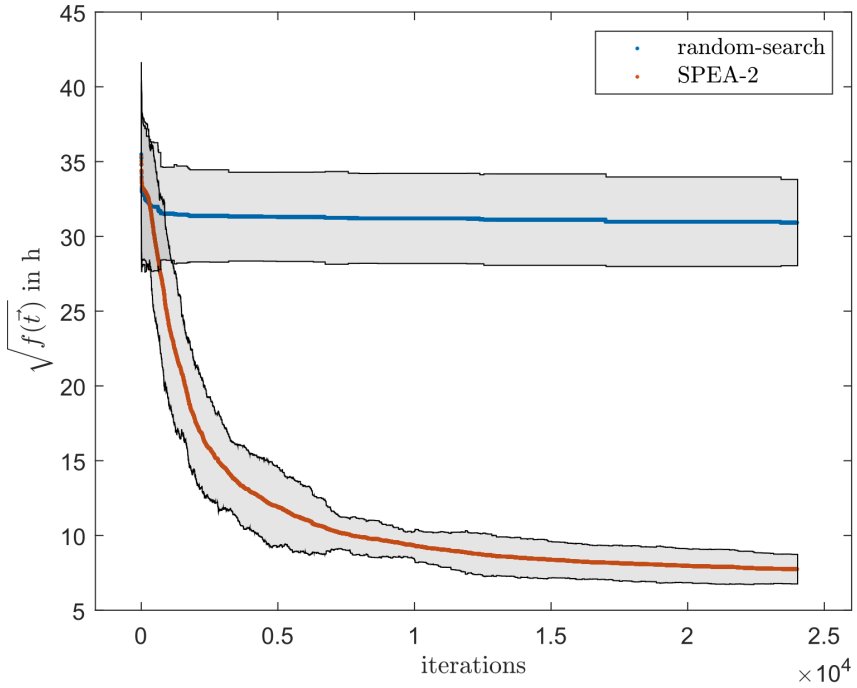


Figure 10: Development of solution quality over time for Galileo reference constellation. Random-search (blue) and proposed algorithm (red) shown as average best solutions after a number of iterations over 15 independent runs. Gray areas show 3σ confidence intervals.

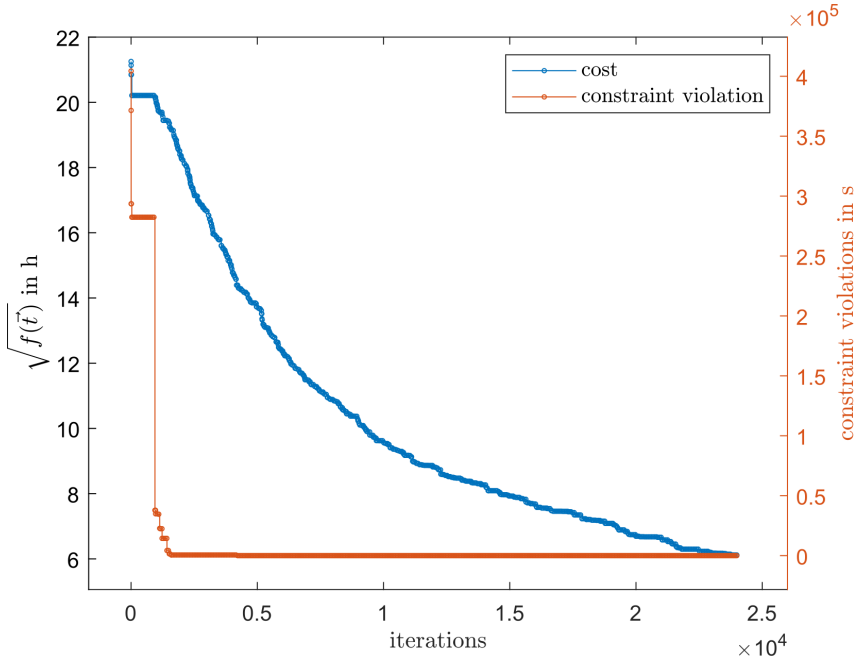


Figure 11: Development of solution quality over time for stitched Walker constellation. Cost of best found solution (blue) and degree of constraint violation (red) for a single run of the proposed algorithm.

5 Conclusion

In this paper, we proposed a constellation planning algorithm that can find optimal solutions for quadratic cost functions. The two-phase approach allows the algorithm to find guaranteed optimal solutions in subregions and avoids finding only local optima via the usage of a genetic algorithm for subregion selection. The genetic algorithm also allows the evaluation of subregions to be parallelized. The algorithm is capable of working on small constellations as well as on larger ones like the upcoming IRIS^2 constellation. The algorithm is integrated into CGI's mission planning system Pleniter® Plan.

In the future, we will continue to improve the performance of the implementation to allow for more iterations and therefore increased solution quality in the same time frame. Potentially, an improved genome representation could lead to a better solution quality in less iterations. Lastly, it should be investigated if an even bigger class of cost functions could be supported.

References

[1] Benjamin Altenstein. "A scalable Schedule Optimization Solution for Satellite Constellations". Supervisor: Martin Tajmar. Diploma Thesis. Dresden, Germany: TU Dresden, 2024.

[2] Marvin Böcker, Ralph Biggins, and Michael Schmeing. "Lights-Out: An Automated Ground Segment for unstaffed Satellite Operations". In: 18^{th} International Conference on Space Operations. Montreal, Canada, 2025.

[3] Peter Brucker and Sigrid Knust. *Complex scheduling*. In collab. with Gesellschaft für Operations Research. 2nd ed. GOR publications. Heidelberg; New York: Springer, 2012. 340 pp. ISBN: 978-3-642-23928-1 978-3-642-06734-1.

[4] Doo-Hyun Cho et al. "Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation". In: *Journal of Aerospace Information Systems* 15.11 (Nov. 2018), pp. 611–626. ISSN: 2327-3097. DOI: 10.2514/1.I010620. (Visited on 04/07/2024).

[5] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. ISBN: 978-3-662-44873-1 978-3-662-44874-8. DOI: 10.1007/978-3-662-44874-8. (Visited on 04/24/2024).

[6] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1998.

[7] Antonio J. Nebro, Juan J. Durillo, and Matthieu Vergne. "Redesigning the jMetal Multi-Objective Optimization Framework". In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO '15: Genetic and Evolutionary Computation Conference. Madrid Spain: ACM, July 11, 2015, pp. 1093–1100. ISBN: 978-1-4503-3488-4. DOI: 10.1145/2739482.2768462. (Visited on 04/24/2024).

[8] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. 2nd ed. Springer series in operations research. New York: Springer, 2006. 664 pp. ISBN: 978-0-387-30303-1.

[9] Donald A. Parish. "A Genetic Algorithm Approach to Automating Satellite Range Scheduling". Supervisor: James W. Chrissis. Master's Thesis. Wright-Patterson, USA: Air Force Institute of Technology, 1994.

[10] Michael Schmeing and Nicholas Symons. "Unsupervised Hierarchical Planning for Geostationary Satellite Missions". In: 17^{th} International Conference on Space Operations. Dubai, United Arab Emirates, 2023.

[11] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm". In: *ETH Library* (2001).