SSC-18-PI-40

Automatic Scheduling for a Ground Segment as a Service Platform Dedicated to Small Satellites

Enrico M. Zucchelli, Ruben Di Battista, Giovanni Pandolfi Bortoletto, Luca Rebellato
Leaf Space
Via Cavour 2, 22074, Lomazzo, Italy
giovanni.pandolfi@leaf.space

ABSTRACT

Together with the development of nano, micro, and small satellite missions and constellations, the necessity for efficient and tailored ground segments is raising. The peculiarities of the market together with the technological developments of the recent years have led to the idea of ground segment as a service. To meet these needs Leaf Space introduced Leaf Line. An essential part of such service consists of scheduling contact windows over the worldwide-deployed network of ground stations. This is an NP-hard problem, which is often solved with methods belonging to the class of operational research. Generally, the orbits of small satellites are very low, characterized by short-timed contact windows. This condition leads to needs way different from those associated to long-lived high-orbit satellites, which most of the literature on scheduling algorithms for telecommunication systems is focused on. Furthermore, a service dedicated to SMEs and NewSpace startups brings additional challenges linked to customer needs. These peculiarities require the development of new, tailored, scheduling algorithms. In the proposed strategy it is assumed to have no information about the state of the satellite (stored data and available energy), and that start and end of contact windows are fixed. In this work, the scheduling is treated as a highly constrained combinatorial optimization problem; various approaches are described and then compared. Such algorithms are iterative, and they all leverage the structure of the problem; specifically, many efforts are made to appropriately reduce the search space. Although optimality cannot be guaranteed, good solutions that are reasonably close to optimal can be obtained. It is found that depending on the problem settings, different algorithms can stand out as the best ones. This paper presents the work done on the scheduling library that is currently powering the Leaf Line network: this platform is offering an easy-to-use, cloud-based and high-availability ground segment service for small satellites operators.

INTRODUCTION

Leaf Line is a ground segment as a service platform dedicated to the monitoring and management of small satellites. Leaf Space is planning to expand its network of ground stations (GSs) considerably in the next few months, in order to enhance the Leaf Line service. A typical ground station is shown in Figure 1. A vital part of the Leaf Line service is the automatic scheduler. Given a list of passages $\boldsymbol{p_{av}}$, the scheduler should provide the optimal schedule $\boldsymbol{s_{opt}}$. Such problem is called Satellite Range Scheduling Problem (SRSP), and it can be treated in many different ways, mostly depending on what its specific features are like. One way consists of seeking the solution in the permutation space. In particular this is possible with high altitude satellites, which have long visibility windows, but need to communicate for a duration that is much shorter. In this case, many possibilities from literature arise, as this is a variation of the problem of late jobs minimization. One example of this is the Air Force Satellite Control Network (AFSCN) scheduling [1]. SRSPs have often been considered very similar to the problem of scheduling satellite observations, which is well described in [2], and approaches that are good for one of the two problems have shown to be effective at solving the other one too. Several methods have been used to solve the SRSP: greedy algorithms [3], squeaky wheel optimization [4], simulated annealing [5], evolutionary algorithms (in particular, Genitor proved to be very successful [6]), hill-climbing, and more. Greedy algorithms [7,8] have been shown to be optimal when a single ground station is available [9]. When multiple ground stations (GSs) and multiple satellites are considered, iterative approaches seem to be necessary. An iterative approach is one in which, at each iteration, one or more new schedules are generated from those available from the previous iteration. An initial guess, or a heuristic to provide one, is necessary. Among these, squeaky wheel optimization was shown to be particularly efficient, because it is capable of looking for new solutions that are relatively far from the current one [10]. In contrast, a hill-climbing method doing only one change per iteration was seen to be not as performing. Nonetheless, Barbulescu et al. [11] showed that the best option for the AFSCN problem was using the Genitor

algorithm, as it seems to be able to exploit hidden patterns in the data. Such result was further confirmed by Barbulescu et al. in 2006 [1] showing that Genitor performed best also for newer versions of the problem.

Figure 1. A Leaf Space GS in Vimercate, Italy.

If satellites are in low Earth orbits (LEO), the availability windows are shorter, and thus they are used in their entirety. Searching a solution in the space of permutations becomes inefficient: in this case a passage can more simply either be scheduled, and thus labeled as a "1", or not scheduled, labeled as "0". This already reduces the search space $\mathcal{S}$ of all schedules s by much, as $|\mathcal{S}| = 2^n$, where n is the number of passage requests; instead, for large n, the number of permutations is approximated by $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ [12], which grows super-exponentially, since there is the term n^n. The approach of considering passages as either 1s or 0s will be used in this work. An alternative would be that of using methods such as mixed integer programming, or linear programming, as in [13]. Such approach however cannot be applied to the problem of this work as it requires good knowledge of the state of the satellites, namely the amount of data collected and their energy availability. While the state of the satellite is certainly important, it is unlikely that all satellite operators would be willing to share those with their ground segment providers. Moreover, enforcing constraints such as the requirement that a satellite communicates with only one GS at a time, and that a GS communicates with only one satellite, makes most of the benefits of using a linear programming approach fade away. Hence, this paper shows how the SRSP can be treated as a constrained combinatorial optimization problem, and how lack in information of the state of the satellite implies additional constraints. Different search spaces are considered, and finally various optimization techniques are described and compared.

PROBLEM DESCRIPTION

This problem can be treated as a constrained combinatorial optimization problem. This is very different from how similar problem have previously been treated in literature. The main constraint is the No Conflict (NC) constraint: it consists of the fact that a ground station cannot communicate with more than one satellite at a time, and a satellite cannot communicate with more than one ground station. Additional constraints are:

- Positioning Time (PT): the minimum time for a GS to start communicating with a satellite after having finished communications with another one
- Min Orbits (O_{min}): the minimum number of revolutions after which a satellite can be communicating with a GS again.
- Max Orbits (O_{max}): the maximum number of revolutions before which a satellite has to be communicating with a GS again.
- Min Passages (P_{min}): the minimum number of passages a satellite has to have (per day).
- Max Passages (P_{max}): the maximum number of passages a satellite can have (per day).

The O_{min} constraint is often caused by energy requirements: a satellite cannot communicate too often, as it would deplete its energy storage, and needs some orbits to refill its batteries. The O_{max} constraint is instead usually driven by data requirements: a satellite is expected to fill its data storage after a certain amount of time. Alternatively, the cause could also be a requirement for maximum latency allowed for the data between its collection time and its communication to Earth. Constraints on number of passages involve instead contractual agreements between the GS provider and the satellite operator, in order to have an indicative range of passages required.

The constraints on positioning time and min orbits are easy to handle: they can be seen as an extension of the NC constraint. A GS that has a scheduled passage cannot communicate with any other satellite for a period of time of PT before the AOS of the given passage to PT after LOS of such passage. Similarly, such satellite cannot communicate with other GSs for a time that goes O_{min} before AOS to O_{min} after LOS. From this point forwards, when referring to the NC constraint, it includes both the O_{min} and the PT constraints. Moreover, from now on, the P_{max} constraint is not considered, as it is very easy to respect (a schedule that has more passages

than P_{max} can easily be "cut" at the end of the scheduling process).

SEARCH SPACES

At this point, there are three different search spaces for a scheduler to work in:

- All schedules $\forall s \in \mathcal{S}$.
- All schedules respecting NC, $\forall s \in \mathcal{NC}$.
- All schedules respecting P_{min} and O_{max}, $\forall s \in \mathcal{P}_{min} \cap \mathcal{O}_{max}$.

Searching in any of these spaces has advantages and disadvantages. Reducing $\mathcal{S}$ to $\mathcal{NC}$ can be enforced by allocating passages one at a time, and removing from the available ones those conflicting with the already allocated ones. Reducing $\mathcal{S}$ to $\mathcal{P}_{min} \cap \mathcal{O}_{max}$ can be enforced by allocating all passages for a given satellite at once: a random number of passages, between P_{min} and P_{max}, is chosen; if then, O_{max} is not satisfied, one or more passage is added. The approaches used for case 2) and 3) may be merged, such that one could end up searching in the space of all and only feasible schedules; nonetheless, it is usually very likely that after scheduling a few satellites, there would not be enough nonconflicting passages to satisfy P_{min} for the remaining satellites. Hence, there is no way to deterministically build a schedule that satisfies all constraints. This means that enforcing all these constraints does not end up in an effective reduction of the search space, as it might lead to a still not satisfactory schedule. Nonetheless, an approach similar to this has still been employed in this work (see the Mar strategy in the next section).

To give a perspective of the search space size of each approach, the test case that has been used in this work is briefly described:

- Optimal schedule: 14,1 passages per satellite (846 in total)
- Available passages per satellite ($N_{pass,sat}$: 42,7 (2562 in total)
- Number of satellites N_{sat}: 60

Hence, $|\mathcal{S}| = 2^{2562} \approx 10^{770}$ (each passage can either be or not be in the schedule). For the cardinality of $\mathcal{NC}$, some assumptions are required. Every time a passage is allocated, the space of all passages that can be allocated at the next step is reduced by a certain number, which is the number of passages conflicting with the allocated passage (including the passage itself). It is reasonable to assume that such number decreases during the scheduling: when the 440th passage is being allocated, it is reasonable to assume that half of the passages conflicting with it have already been removed. Hence, the number of schedules possible with this approach can be computed as follows:

$$|\mathcal{NC}| = \sum_{|s|=|s|_{min}}^{|s|_{max}} \frac{\prod_{i=1}^{s} |\boldsymbol{p_{av,i}}|}{|\boldsymbol{s}|!}$$

Where $|\boldsymbol{p_{av,0}}|$ is the number of initially available passages, $|\boldsymbol{p_{av,i+1}}| = |\boldsymbol{p_{av,i}}| - \Delta(\mathrm{i})$ is the number of available passages at iteration $i+1$, and $\Delta(\mathrm{i})$ is the number of passages conflicting with i^{th} passage. $|\boldsymbol{s}|$ is the number of scheduled passages. $|\boldsymbol{s}|_{\boldsymbol{min}}$ is the minimum number of passages that can be scheduled when all non-conflicting passages are allocated, where $|\boldsymbol{s}|_{\boldsymbol{max}}$ is the maximum. For simplicity, it is assumed that Δ decreases linearly when i increases (and goes to 0 when $i = |\boldsymbol{s}|$). As a consequence, the function $\Delta(i)$ depends on $|\boldsymbol{s}|$. As an example, for a schedule with 716 scheduled passages, $\Delta_i = \frac{|\boldsymbol{p_{av,i}}|}{200} + 1$, whereas for a schedule with 832 scheduled passages, $\Delta_i = \frac{|\boldsymbol{p_{av,i}}|}{300} + 1$. For a single value of the final length of the schedule (as long as it ranges between 500 and 900), one obtains values of the set size always in the proximity of 10^{125}. Being conservative, one can state that the search space has been reduced to close to 10^{135}.

Concerning the last case, where $\forall s \in \mathcal{P}_{min} \cap \mathcal{O}_{max}$, the search space size can be computed as follows (assuming, for simplicity, that $P_{min} = P_{max} = 14$):

$$\prod_{sat=1}^{N_{sat}} \binom{N_{pass,sat}}{P_{min,sat}}$$

The result is in the range of 10^{640}: while still a huge reduction if compared to the value of $|\mathcal{S}|$, the space is still hundreds of orders of magnitude larger than what can be achieved by enforcing case 2. In this work, it is decided to search in $|\mathcal{NC}|$, because of the very large reduction in combinations that it offers. Obviously, such reduction is problem-dependent; as an example, if, in the previous example, the number of available passages per satellite were equal to P_{min}, then searching in $\mathcal{P}_{min} \cap \mathcal{O}_{max}$ would have been the most efficient: if the solution exists, it is the only one that is in that set.

The reduction in search space does not come without any drawbacks. First, it causes an increase in computational time for the generation of the schedule. Second, it makes representation of the solution more complicated. If the search occurs in $\mathcal{S}$, every passage can be a 0 (not scheduled) or 1 (scheduled); this makes the solution very suitable for methods such as genetic algorithms.

Searching for solutions in $\mathcal{NC}$ with an evolutionary algorithm is instead less straight-forward.

SCHEDULING PROCEDURE

The scheduling procedure involves two phases. The first is the construction of an initial schedule, or the initial guess. The second consists of iteratively modifying said schedule, using a specific operation and one of many methods to decide whether to accept or not the modification.

Initial Guess

The initial schedule is generated starting from the list of all available passages, $\boldsymbol{p_{av}}$. At iteration i, a passage p_i is picked from the list of the remaining available passages $\boldsymbol{p_{av,i}}$ and inserted into the intial schedule $\boldsymbol{s_{0,i}}$; at the same time, all passages that are in conflict with p_i are removed from $\boldsymbol{p_{av,i}}$. This way, the generation of a schedule that is in the space $\mathcal{NC}$ is guaranteed. In this work, the passage p_i is chosen in one of two ways:

- Full Random (FullR): each passage has equal probability of being chosen;
- Margin (Mar): priority is given to passages of satellites that are farther from satisfying the P_{min} constraint.

The latter is the solution mentioned in the previous section, which attempts to look for a schedule $s \in \mathcal{NC}$, while trying first to satisfy P_{min} for all satellites. Independently of which of these two ways is chosen, both procedures ensure that the initial schedule is such that no passage that would cause a conflict can be added.

Elementary Operation: Passage substitution

Once the initial schedule is generated, the iterative process begins. The elementary operation that is made over the schedule is the substitution. A random passage, that is not in the schedule, and hence is conflicting with at least one of the scheduled passages, is added to the schedule. The passages that are conflicting with it are removed, and the passages in conflict with the removed ones are considered. Among the last set of passages, those that are not in conflict with any passage iof the current schedule are iteratively added, in the same way the initial guess is generated (hence, with one of the two methods, either FullR or Mar). Also in this case, it is guaranteed that the new schedule generated belongs to $\mathcal{NC}$.

Elementary Operations per Schedule Iteration

Even the number of operations done per iteration may differ: according to Barbulescu et al. [10], it may be beneficial to do more than one operation on the schedule before evaluating again. Such number may be fixed, random, a function of how many iterations have already been carried out (and how many are left), or may depend on how many satellites currently do not satisfy P_{min} (and by how much). A particular option is the "guided" strategy, in which a passage is added, for each satellite that currently does not satisfy P_{min}. In this work, when the "guided" strategy is used, the added passage is, 95% of the times, not in conflict with the passages of the same satellite (but of course will be in conflict with passages of other satellites). It is decided to keep a 5% of cases in which the passage may be in conflict with passages of the same satellite: this is because there is a chance that removing a passage from that satellite might free more than one passage of the same satellite.

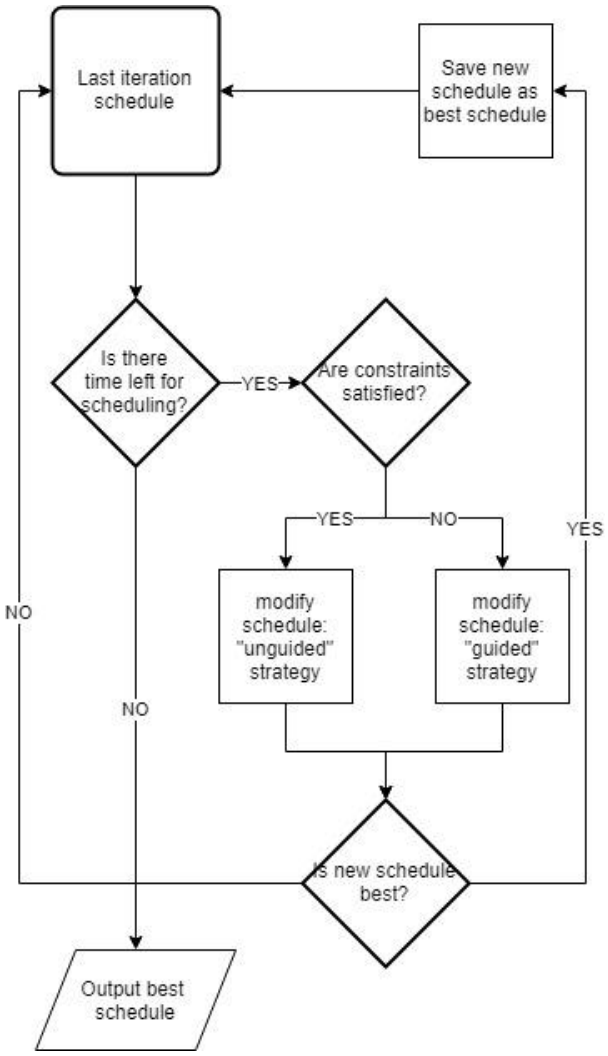

Figure 2. Flowchart of the re-planning decision process.

Next Schedule Acceptance

Once the next schedule has been generated, it is first necessary to evaluate it. Among many, the most

important factors in the schedule evaluation consist of whether the requirements are satisfied, how many passages have been scheduled, what the average elevation at time of closest approach (TCA) of the passages is. After evaluation, it is important to define a way to decide whether the new schedule should be kept or not. Common ways are hill climbing (HC), random walk (RW), or simulated annealing (SA). With HC, the new schedule is kept only if it scores equal or better than the previous one; conversely, in a random walk the schedule is always kept (but the highest scoring schedule is kept in memory); finally, with simulated annealing the new schedule is kept with probability one if it scores better than the previous one, and with probability less than one is it does not. Probability also decreases when the number of iterations increases. The best method to use depends on the shape of the cost function. For example, hill climbing works well when the cost function has no local optima, but only a global one. Simulated annealing is instead a good choice in case the function is relatively smooth but has many local minima.

Parallelization

It should be noted that the building of new schedules may in some cases also be parallelized. For example, a HC procedure can be parallelized, especially in the advanced phases of the search, when only one in hundreds or thousands of modifications results in improvements of the schedule. It is possible to do parallel modifications, and thus generate several new schedules at the same time, knowing that only a very small fraction, if any, of those new schedules will lead to an improvement.

RE-PLANNING

It may happen that a disturbance occurs after the schedule has been communicated to the customers. To the best of the authors' knowledge, the problem of re-planning a satellite contact schedule has never been treated in literature so far; nonetheless, in the more general field of scheduling, some examples of re-planning are available [14,15]. A disturbance may be an urgent request of contact, or a temporary unavailability of a GS. In such case, there are several priorities: 1) a new schedule has to be generated in a very short time; 2) as many requirements as possible have to be respected, but, if necessary, constraints on P_{min} and O_{max} may be relaxed; 3) the re-planner has to delete as few passages as possible. This is a constrained combinatorial multi-objective optimization problem, one of the objectives being the maximization of the quality of the schedule, and the other one being the minimization of the number of passages that were in the original schedule and now have been deleted. A "guided" RW approach is used: this way, many schedules are explored, and the "guided" strategy tends to bring improvements at each iteration. If all constraints are satisfied, then RW is continued, with a random number of changes per iteration (hence, the process is not "guided" anymore). After a certain amount of time has passed, the best schedule is chosen among those that have been explored. Figure 2 illustrates this process. At each iteration, it is decided whether the new schedule is better than the currently best one; this is done according to the following schema:

1. Does it satisfy more or less constraints than the current best schedule? If less, it is is not the best, if more, it is the best; if none of those, go to case 2.
2. Does it have more or less passages deleted than the current best schedule? If less, it is the best schedule, if more, it is not; if none of those, go to case 3.
3. Does it have more or less passages than the current best schedule? If less, it is not the best, if more (or equal), it is the best.

TEST CASE DESCRIPTION

The problem that will be used to test the various algorithms consists of a constellation made of 60 satellites, spread equally over 6 planes. Said planes all have an inclination of 30° and are equally distanced in RAAN. The ground stations are 6, positioned along two belts at 20.5° N and 20.5° S, and equally spaced in longitude, as follows:

- GS 1: 20.5° N, 120° W
- GS 2: 20.5° S, 120° W
- GS 3: 20.5° N, 0° E
- GS 4: 20.5° S, 0° E
- GS 5: 20.5° N, 120° E
- GS 6: 20.5° S, 120° E

Passages whose elevation at TCA is less than 7.5° are not considered. Each satellite has to satisfy $O_{min} = 0.8$: hence, only one passage per orbit is allowed. This problem can easily be solved exploiting symmetries. It is sufficient to split it into 3 different problems, each with planes of satellites opposite to each other, and 2 GSs with same longitude. Hence, the global optimum can be easily found. The problem becomes extremely complex when seen in its entirety, and thus this case is very suitable for testing an algorithm, as it provides a difficult problem whose global optimum is known. Summarizing, the test case consists of 60 satellites having passages over 6 ground stations; the number of total passages is 2562, and the optimal schedule consists of 846 passages. For the time being, no constraints on max orbits are being considered.

RESULTS

Simulated Annealing

Simulated annealing has been performing considerably worse than expected. After letting the optimizer work for many hours (all other cases shown here refer to optimization durations of less than 15 minutes), the best schedule found contained 818 passages. This is likely because of a twofold reason: 1) the problem seems to be very flat, and also shows very few local minima; 2) it is difficult to appropriately tune the solver. Statement 1) was proven when using an HC approach with only one passage modification at each iteration: if the modification did not decrease the number of passages, the new schedule was kept. The algorithm managed various times, and very slowly, to schedule as many as 843 passages, which is extremely close to the global optimum. This showed that, if there are any local minima, these are extremely close to optimality, hence making the use of an algorithm such as SA not needed. Moreover, flatness of the function was shown by the fact that even at the end of the optimization, when the score of the schedule was very high, plenty of modifications were being accepted without causing any improvements. This feature was also found by Barbulescu et al. [11]. Nevertheless, these considerations cannot be generalized to other test cases, and may be a consequence of the symmetries of the problem used here. Statement 2) may be solved by using Adaptive Simulated Annealing (ASA) [16], but this was not done for the moment. In fact, it is still likely that even ASA may not perform well, for the reasons related to stamen 2).

Hill Climbing

HC was then evaluated; three cases were considered: 1) only one substitution per iteration; 2) a random amount of substitutions per iteration, uniformly distributed between 1 and 10; 3) the "guided" approach, setting P_{min} =14 for all satellites. While the number of iterations is very different for the different strategies, the runtime is approximately kept the same for all cases (around 15 minutes, without using parallelization). The convergence rate for a single passage modification is illustrated in Figure 3, whereas the one for 10 passage modifications per iteration is illustrated in Figure 4. A few things can be noticed. First, both methods have a large range of logarithmic convergence. Ideally, one would not want to continue the optimization for much longer after this range is over. Unfortunaly, it is a bit hard to estimate when this ends for the 10 substitutions strategy. While the two lines seem to follow similar paths, it is reminded that the two methods are stopped after equal runtime. Hence, simply stated, the 10 substitutions strategy requires almost double the time per itration. Second, the method with one passage substitution performs much better than the one with 10. In fact, not only it provides an average result of 831, compared to 816 of the other strategy, but it scores better even for same number of iterations: after 6000 iterations, the single substitution strategy shows an average of 828 passages. The "guided" case is then considered. It is stopped after 500 iterations (iterations are much more intense computationally, as they require 60 substitutions each, on average). Nonetheless, the guided method turns out less performing, most likely because it does too many changes all at once. A "guided" strategy with a single substitution per iteration may be a valid improvement to the scheduler.

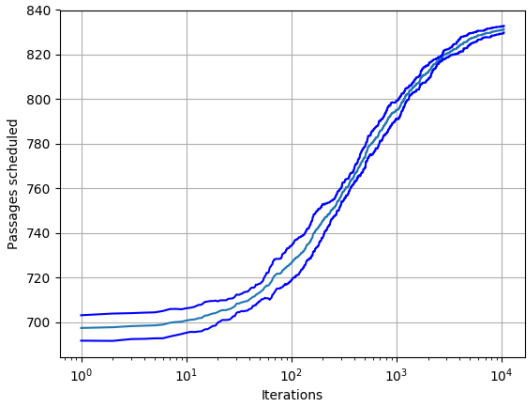

Figure 3. Convergence rate (average and standard deviation) for HC strategy, with one passage substitution per iteration.

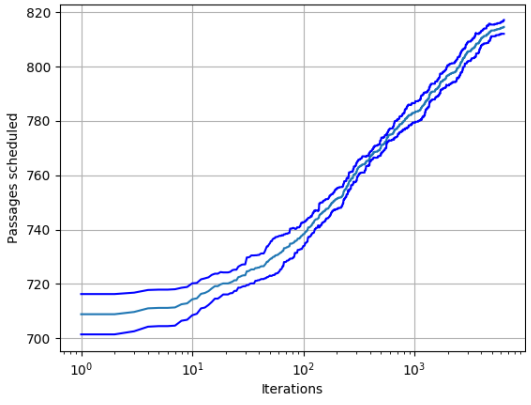

Figure 4. Convergence rate (average and standard deviation) for HC strategy, with one passage substitution per iteration.

Random Walk

Finally, an RW strategy was tested. Nonetheless, it performed poorly. After several hours of running, the algorithm offered a solution which had less than 740 passages when only one passage was changed per iteration, and less than 770 passages for the guided strategy.

CONCLUSIONS

HC strategy is clearly the best performing algorithm in this case. Unexpectedly though, it is found that the strategy with only one change per iteration is better not only than the strategy of having more, randomly picked, modifications, but even than the "guided" approach. Nonetheless, this might be caused by the fact the "guided" approach makes many changes at the time. In line with the fact that a single change at a time works better than the strategy with many modifications per iteration, it might be interesting to look at a "guided" algorithm which only makes one modification per iteration. Moreover, for the HC strategy to properly work, it is important that schedules that score equally are accepted, otherwise the algorithm would get stuck much sooner.

Recommendations

Recommendations for future work are several. First, it would be necessary to test these scheduling techniques in less symmetrical situations, although this would imply not having knowledge of the global optimum. Additional improvements to the scheduler may be the implementation of Genitor, an evolutionary algorithm that seems to understand and exploit hidden patterns when solving the similar AFSCN problem [11]. Despite using a genetic algorithm would most likely require the method to search the solution in $\mathcal{S}$ instead of in $\mathcal{NC}$, the use of Genitor may still turn out to be beneficial.

A final recommendation concerns robustness. It might in fact be interesting to have a schedule that is not only optimal, but that can also be replanned on-the-fly and with minimum differences in case disturbances occur after the schedule has already been notified to the customers. Sources of disturbances may be temporary unavailability of a GS, or additional requests from a satellite. In such a case, it is important that the scheduler plans a minimum number of changes from the previous schedule, for the customers to have little inconvenience. While we have already developed a software that replans the schedule minimizing the number of changes, it would be interesting to evaluate how taking disturbances (and their probability to occur) into account during the scheduling process itself would affect the process and the overall performance of the system.

References

1. Barbulescu, L., Howe, A. and D. Whitley, "AFSCN scheduling: How the problem and solution have evolved," Mathematical and Computer Modelling, vol. 43, no. 9-10, 2006.
2. Frank, J., Jonsson, A., Morris, R. and D. Smith, "Planning and scheduling for fleets of earth observing satellites", Proceedings of the 6th International Symposium on Aritificial Intelligence, Robotics, Automation and Space, 2001.
3. J.C. Pemberton, "Toward scheduling over-constrained remote-sensing satellites," Proceedings of the 2nd NASA International Workshop on Planning and Scheduling for Space, San Francisco, CA, 2000.
4. Joslin, D.E. and D.P. Clements, "Squeaky wheel optimization," Journal of Artificial Intelligence Research, vol. 10, 1999.
5. Kirkpatrick, J., Gelatt, C.D., Jr. and Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, 1983.
6. D. Whitley, "The GENITOR Algorithm and Selection Pressure," Proceedings of the 3rd International Conference on Genetic Algorithms, San Mateo, CA, 1989.
7. Cormen, T.H., Leiserson, C.E. Rivest, R.L. and C. Stein, Introduction to Algorithms, Third Edition, The MIT Press, Cambridge, MA, 2009.
8. S. Dauzère-Pérès, "Minimizing Late Jobs in the General One Machine Scheduling Problem," European Journal of Operational Research, vol. 81, 1995.
9. S.E. Burrowbridge, "Optimal Allocation of Satellite Network Resources", MSc Thesis, Virginia Polytechnic Institute and State University, 1999.
10. Barbulescu, L., Whitley, D. and A.E. Howe, "Leap Before you Look: An Effective Strategy in an Oversubscribed Scheduling Problem," Proceedings of the 19th National Artificial Intelligence Conference, San Jose, Ca, 2004.
11. Barbulescu, L., Howe, A.E., Whitley, L.D. and M. Roberts, "Trading Places: How to Schedule More in a Multi-Resource Oversubscribed Scheduling Problem,"
12. D. Romik, "Stirling's Approximation for $n!$: The Ultimate Short Proof?," The American

Mathematical Monthly, vol. 107, no. 6, June-July, 2000.

13. J. Castaing, "Scheduling Downloads for Multi-Satellite, Multi-Ground Station Missions," 28th Annual AIAA/USU Conference on Small Satellites, Logan, UT, 2014.
14. Herroelen, W.S., de Reyck, B. and Demeulemeester, E.L., "Project Scheduling under Uncertainty: Survey and Research Potentials," European Journal of Operational Research, vol. 165, no.2, 2005.
15. Gombolay, M.C., Wilcox, R.J. and J.A. Shah, "Fast Scheduling of Multi-Robot Teams with Temporospatial Constraints," Proceedings of Robotics: Science and Systems (RSS), Berlin, Germany, June, 2013.
16. L. Ingber, "Adaptive Simulated Annealing – Lessons Learned," Control and Cybernetics, vol. 25, no. 1, 1996.