Tabu-Based Large Neighbourhood Search for Time-Dependent Multi-Orbit Agile Satellite Scheduling

He, Lei; de Weerdt, Mathijs; Yorke-Smith, Neil; Liu, Xiaolu; Chen, Yingwu

Publication date
2018

Document Version
Final published version

Published in
SPARK 2018

In: Proc. of 11th Scheduling and Planning Applications Workshop (SPARK), Delft, Netherlands, June 2018

Tabu-Based Large Neighbourhood Search for Time-Dependent Multi-Orbit Agile Satellite Scheduling

Lei He[1,2], Mathijs de Weerdt[2], Neil Yorke-Smith[2], Xiaolu Liu[1], Yingwu Chen[1]
[1]College of System Engineering, National University of Defense Technology, 410073 Changsha, China
[2]Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands

Abstract

Agile Earth observation satellite (AEOS) scheduling is complex, due to long visible time windows and time-dependent transitions between observations. We introduce a generic approach suited for scheduling problems characterised by time-dependency and/or sequence-dependency. Our approach is a novel hybridization of adaptive large neighbourhood search (ALNS) and tabu search. We further introduce partial sequence dominance and insertion position ordering operators to the ALNS. Extensive computational results on a real-world multi-orbit AEOS observation scheduling benchmark show that the hybrid ALNS robustly outperforms an improved mixed integer programming model and two recent state-of-the-art metaheuristic methods. The proposed method increases solution quality by more than 10% and reduces calculation time by more than 70% on average.

Introduction

Agile Earth observation satellites (AEOSs) are a new generation of orbital imaging platforms, possessing three degrees of freedom (roll, pitch, yaw), which enables them to observe targets on the Earth's surface before/after an upright pass and next to/along the path (Maillard 2015). This agility greatly enhances the observing abilities of AEOSs.

Scheduling the operation of AEOSs is complex due to long visible time windows (VTWs), and time-dependent transitions. During the VTW the target is visible for the satellite. AEOS VTWs are much longer than the necessary imaging time; target imaging can start anywhere within its VTW. During the transition time, the satellite adjusts its observing angle between two adjacent observations. This transition time is not only sequence-dependent, but also time-dependent because it depends on the observing angles, which differ for different observation start times. In addition, the observation start time also influences the image quality. The best image quality can be acquired when the satellite is at the nadir, i.e., the middle of the VTW. AEOS scheduling is an NP-hard combinatorial optimization problem (Lemaître et al. 2002).

Research on the offline AEOS scheduling problem can be divided into the Maximum Shot Orbit Sequencing Problem (MSOP) and the Maximum Shot Sequencing Problem (MSP) (Lemaître et al. 2002). MSOP aims to select the images with maximum total priority from a single orbit of one satellite and determine the observing sequence and the observing times without violating constraints. MSP is more complex and realistic. In addition to selecting the image-taking tasks, we must also decide which VTWs are chosen out of *several* consecutive orbits. These two decisions are dynamically coupled rather than procedurally separated.

Due to the complexity of the MSP, there exist few exact algorithms. Bianchessi et al. (2007) propose a column generation method to solve a linear programming (LP) relaxation of the problem. Wang et al. (2011) propose a mixed integer programming (MIP) model, where the continuous observation angle is discretized as only three angles. As a result of this approximation, the solution space is reduced and the transition time can be pre-computed. Both methods can only be used in small-size instances. Besides exact algorithms, a variety of metaheuristics and heuristics have been applied to MSP, including tabu search (Lin et al. 2005; Bianchessi et al. 2007), simulated annealing (Dilkina and Havens 2005; Li, Xu, and Wang 2007), genetic algorithms (Wolfe and Sorensen 2000; Li, Xu, and Wang 2007), hybrid differential evolutionary algorithms (Li et al. 2017) and priority-based constructive algorithms (Wolfe and Sorensen 2000; Wang et al. 2011; Xu et al. 2016). However, all the above works neglect the transition time, fix it as a constant value, or simplify it as a sequence-dependent time.

The only works on MSP with time-dependent transition time to date are by Geng et al. (2016) and Liu et al. (2017). Geng et al. (2016) propose a genetic algorithm and only briefly treat the time-dependency. They also ignore the constraints of image quality, onboard memory and onboard energy. Liu et al. (2017) first define a mathematical model that is non-linear due to the time-dependent constraints. Then, in order to use LP, the authors linearize and simplify the MSP into two separate subproblems: all feasible VTW combinations are enumerated, then, for each combination, LP is used to obtain the optimal schedule. The transition time is fixed as a constant, a major approximation, but even so the approach could not solve instances involving more than 12 tasks. Therefore, the authors propose a metaheuristic based on adaptive large neighbourhood search (ALNS) combined with a fast task insertion heuristic. The angles in VTWs are pre-computed and cached. The ALNS method performs well for small-size instances, while when the problem instance

grows in size, the solution quality deteriorates and the computation time grows.

In this paper, we investigate the time-dependent multi-orbit AEOS observation scheduling problem. The major contributions of this paper are summarized as follows:

1. For the first time, a complete MIP model is defined. Compared with the two-stage mixed integer linear programming (MILP) model in Liu et al. (2017), our model avoids enumerating all VTW combinations. The time-dependent onboard energy constraints are also considered. The new MIP model scales better and is more realistic.
2. An improved ALNS is hybridized with tabu search. Our novel hybrid approach provides results with higher quality and robustness and consumes less time compared with state of the art. The tabu mechanism helps the ALNS to avoid searching recently visited solutions.
3. A partial sequence dominance heuristic is proposed, which can help to collect and use the in-process information that is neglected in standard ALNS. It greatly improves the performance of ALNS, especially when the problem instance grows in size.
4. A position ordering heuristic is included in the task insertion algorithm. This strategy explores more insertion positions following an ascending order of possible transition times to save time and energy resources and increase the possibility of successful insertion.

Problem Description

The time-dependent multi-orbit AEOS observation scheduling problem aims to select a number of tasks from several consecutive orbits and determine the observing sequence and the observing times without violating technical constraints. We define a task t_i as one image or target to observe from the users' task list T. Each task corresponds to a small area on the Earth's surface that can be observed in one pass.

We account for but simplify the onboard memory and energy constraints. We assume that the memory and energy used during each orbit cannot exceed an upper bound to simulate these orbit-renewable resources.

We firstly introduce notation and provides an angle-fitting method to represent the time-dependent transition time in the MIP model. Then we define the MIP model itself.

Time-dependent transition time

According to Liu et al. (2017), the image quality q_i of task t_i must be higher than its required minimum image quality c_i. The image quality is a function of u_{ij}, the observation start time in the j^{th} VTW w_{ij} of t_i, and can be calculated according to the following equation:

$$q_i = 10 - 9\frac{|\frac{u_{ij}+v_{ij}}{2} - w^*_{ij}|}{\frac{l_{ij}}{2} - \frac{d_i}{2}} = 10 - 9\frac{|2u_{ij} + d_i - 2w^*_{ij}|}{l_{ij} - d_i} \tag{1}$$

where v_{ij} is the observation end time of t_i in w_{ij}, d_i is the observation duration, $v_{ij} = u_{ij} + d_i$, l_{ij} is the length of w_{ij}, and w^*_{ij} is the nadir time of w_{ij}.

In Liu et al. (2017), the quality of an image was treated as a constraint, requiring the image quality to be higher than a user-specified minimum value. In our paper, we use Eq. (1) to prune parts of the VTWs to reduce the solution space and increase the accuracy of angle fitting, which enables us to build our MIP model.

According to (1) and $q_i \geq c_i$, the feasible interval of observation start time is within the original VTW, represented by b^*_{ij} and e^*_{ij}:

$$b^*_{ij} = \max\left(\frac{(l_{ij} - d_i)(c_i - 10)}{18} + w^*_{ij} - \frac{d_i}{2}, b_{ij}\right) \tag{2}$$

$$e^*_{ij} = \min\left(\frac{(l_{ij} - d_i)(10 - c_i)}{18} + w^*_{ij} - \frac{d_i}{2}, e_{ij} - d_i\right) \tag{3}$$

The exact VTWs and the observing angle sequences for tasks are computed before solving the actual scheduling problem. This pre-processing phase takes the satellite's position, the task's position as well as the Earth's rotation and produces VTWs and time-dependent functions of the roll, pitch and yaw angles for every VTW. Although these functions are non-linear, the change of angles can be approximated quite well with a linear function between b^*_{ij} and e^*_{ij} (e.g., on average over 200 tasks, the duration of a VTW is over 300s, while the transition time error is less than 0.5s).

Mixed integer programming model

Objective function The importance of one task t_i is evaluated by its priority $g_i \in [1, 10]$. The objective we consider is to maximize the total priority of all the scheduled tasks:

$$\text{Maximize} \quad \sum\nolimits_{i=1}^{|T|} \sum\nolimits_{j=1}^{|W_i|} x_{ij} g_i \tag{4}$$

where W_i is the VTW set of t_i and x_{ij} is a binary decision variable equal to 1 if and only if w_{ij} is chosen to observe t_i. Another key decision variable is u_{ij}, determining the observation start time for t_i in w_{ij}.

Constraints Constraints (5) are the uniqueness constraints, meaning that each task is observed once at most.

$$\sum\nolimits_{j=1}^{|W_i|} x_{ij} \leq 1 \quad \forall t_i \in T \tag{5}$$

Constraints (6) are the manoeuvring constraints: they ensure that there is sufficient time between the end time v_{ij} and the start time u_{kl} for the transition time $\tau_{w_{ij}w_{kl}}$ between tasks t_i and t_k. These constraints are only enforced for VTWs that are closer to each other than the maximum possible transition time τ_{max}, and if w_{ij} and w_{kl} are selected to observe t_i and t_k respectively, and t_i is the immediate predecessor of t_k, which is encoded by $\rho_{w_{ij}w_{kl}} = 1$ and Constraints (7)–(9).

$$\begin{gathered} v_{ij} + \tau_{w_{ij}w_{kl}} \leq u_{kl} \quad \text{if} \quad \rho_{w_{ij}w_{kl}} = 1 \\ \forall t_i, t_k \in T, i \neq k, w_{ij} \in W_i, w_{kl} \in W_k, \\ b^*_{kl} - e^*_{ij} \leq \tau_{max}, b^*_{ij} - e^*_{kl} \leq \tau_{max} \end{gathered} \tag{6}$$

$$\sum_{k=1}^{|T|} \sum_{l=1}^{|W_k|} \rho_{w_{ij}w_{kl}} + \rho_{w_{ij}w^e} = x_{ij} \quad \forall w_{ij} \in W_i, t_i \in T \tag{7}$$

$$\sum_{k=1}^{|T|}\sum_{l=1}^{|W_k|}\rho_{w_{kl}w_{ij}}+\rho_{w^s w_{ij}}=x_{ij}\quad\forall w_{ij}\in W_i,t_i\in T \quad (8)$$

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|W_i|}\rho_{w_{ij}w^e}=\sum_{i=1}^{|T|}\sum_{j=1}^{|W_i|}\rho_{w^s w_{ij}}=1 \quad (9)$$

In Constraints (7) and (8), w^s and w^e are two dummy nodes representing the first and the last VTW on the satellite. These constraints express that if a VTW is selected, there is a unique selected VTW preceding it, and a unique selected VTW following it.

Constraints (10) ensure that the observation for each task lasts for the required duration.

$$v_{ij}=u_{ij}+d_i\quad\forall w_{ij}\in W_i,t_i\in T \quad (10)$$

Constraints (11)–(15) are used to calculate the transition time between two observations. In Constraints (11), $\theta_{w_{ij}w_{kl}}$ is the total transition angle and a_1–a_4 are four different transition angular velocities for different transition angles. In (12), $\gamma_{ij}^t,\pi_{ij}^t,\psi_{ij}^t$ are observation roll, pitch and yaw angles at t, which are calculated by (13)–(15). The parameters a_{ij}^{γ}, a_{ij}^{π}, a_{ij}^{ψ}, b_{ij}^{γ}, b_{ij}^{π} and b_{ij}^{ψ} are the parameters of the functions of angles and time of the chosen VTW w_{ij}, which are computed in the pre-processing phase of section 'Time-dependent transition time'.

$$\tau_{w_{ij}w_{kl}}=\begin{cases}10+\theta_{w_{ij}w_{kl}}/a_1 & \theta_{w_{ij}w_{kl}}\le 15\\ 15+\theta_{w_{ij}w_{kl}}/a_2 & 15<\theta_{w_{ij}w_{kl}}\le 40\\ 20+\theta_{w_{ij}w_{kl}}/a_3 & 40<\theta_{w_{ij}w_{kl}}\le 90\\ 25+\theta_{w_{ij}w_{kl}}/a_4 & \theta_{w_{ij}w_{kl}}>90\end{cases}$$
$$\forall t_i,t_k\in T,i\ne k,w_{ij}\in W_i,w_{kl}\in W_k \quad (11)$$

$$\theta_{w_{ij}w_{kl}}=\left|\gamma_{ij}^{u_{ij}}-\gamma_{kl}^{u_{kl}}\right|+\left|\pi_{ij}^{u_{ij}}-\pi_{kl}^{u_{kl}}\right|+\left|\psi_{ij}^{u_{ij}}-\psi_{kl}^{u_{kl}}\right|$$
$$\forall t_i,t_k\in T,i\ne k,w_{ij}\in W_i,w_{kl}\in W_k \quad (12)$$

$$\gamma_{ij}^t=a_{ij}^{\gamma}t+b_{ij}^{\gamma}\quad\forall w_{ij}\in W_i,t_i\in T \quad (13)$$

$$\pi_{ij}^t=a_{ij}^{\pi}t+b_{ij}^{\pi}\quad\forall w_{ij}\in W_i,t_i\in T \quad (14)$$

$$\psi_{ij}^t=a_{ij}^{\psi}t+b_{ij}^{\psi}\quad\forall w_{ij}\in W_i,t_i\in T \quad (15)$$

Constraints (16) and (17) are the memory and energy constraints, respectively, where α^M and α^E are two estimated values which measure the percentage of total memory M and energy E available on an orbit, r_{ij}^m is a binary parameter showing whether w_{ij} is on the m^{th} orbit o_m of the orbit set O, and m^o,p^o,p^s,p^a are the consumed memory for observation per second, the consumed energy for observation per second, the consumed energy per observation and the consumed energy for angle transition per degree respectively. The energy constraints are also time-dependent because the energy for satellite transition depends on the total angles the satellite rotates. In Constraints (18), $\theta^*_{w_{ij}w_{kl}}$ is an auxiliary variable to calculate the rotation energy. The value of $\theta^*_{w_{ij}w_{kl}}$ is a piecewise linear function influenced by the value of $\rho_{w_{ij}w_{kl}}$.

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|W_i|}r_{ij}^m d_i m^o\le\alpha^M M\quad\forall o_m\in O \quad (16)$$

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|W_i|}r_{ij}^m(x_{ij}d_i p^o+x_{ij}p^s+\sum_{k=1}^{|T|}\sum_{l=1}^{|W_k|}\theta^*_{w_{ij}w_{kl}}p^a)\le\alpha^E E$$
$$\forall o_m\in O \quad (17)$$

$$\theta^*_{w_{ij}w_{kl}}=\begin{cases}\theta_{w_{ij}w_{kl}} & \text{if } \rho_{w_{ij}w_{kl}}=1\\ 0 & \text{otherwise}\end{cases} \quad (18)$$
$$\forall t_i,t_k\in T,i\ne k,w_{ij}\in W_i,w_{kl}\in W_k$$

Constraints (19)–(21) restrict the domains of the variables. Note that in Constraints (20), the start and end time of the VTW have been cut according to the quality constraints in (2) and (3), so there are no additional quality constraints.

$$x_{ij}\in\{0,1\}\quad\forall w_{ij}\in W_i,t_i\in T \quad (19)$$

$$b_{ij}^*\le u_{ij}\le e_{ij}^*\quad\forall w_{ij}\in W_{ij},t_i\in T \quad (20)$$

$$\rho_{w_{ij}w_{kl}}\in\{0,1\}\quad\forall t_i,t_l\in T,i\ne k,w_{ij}\in W_i,w_{kl}\in W_k \quad (21)$$

To the best of our knowledge, this is the first MIP model proposed for the complete time-dependent multi-orbit AEOS observation scheduling problem. The angle fitting strategy enables the modelling of the time-dependency. Compared with the two-stage MILP model in Liu et al. (2017), we avoid enumerating all the VTW combinations and we consider energy constraints, which are also time-dependent.

This problem is NP-hard, and we observe that the runtime of the MIP solver does not scale well with larger problem size. Therefore, in the next section, we propose a metaheuristic approach.

Hybrid ALNS Algorithm

ALNS (Pisinger and Ropke 2007; Liu et al. 2017) provides a flexible framework in which several different operators can be defined according to the problem characteristics. ALNS can be adopted to provide solutions for instances with different characteristics. However, we observe two main drawbacks of ALNS. First, the search efficiency of ALNS can founder due to re-visiting recent solutions. Second, ALNS accepts a new solution depending on the quality of the whole solution sequence. However, during the search process, solutions with some good parts are rejected due to the low quality of the whole sequence – thus neglecting potentially valuable in-process information.

Our ALNS approach is based on the work of Liu et al. (2017). In the following subsections, we first introduce the standard ALNS framework. Then we introduce three new main features of our ALNS approach: tabu search hybridization (TS), partial sequence dominance (PSD), and insertion positions ordering (IPO). The resulting algorithm, called ALNS/TPI, is shown as Algorithm 1. In the fifth subsection we introduce a new definition of conflict degree (CD) which increases the calculation speed.

ALNS framework

ALNS is less sensitive to the initial solution than general local search (Demir, Bektaş, and Laporte 2012), therefore we use a simple greedy search to construct an initial solution. We sort the tasks by an ascending order of start times of

Algorithm 1 Overview of ALNS/TPI

Generate an initial solution S by greedy search;
repeat
  Choose destroy, repair operators D_i, R_i based on weights;
  $S' \leftarrow R_i(D_i(S))$;
  Update tabu attributes of new inserted tasks;
  Produce compound solution S_c from S and S';
  if $f(S_c) > f(S')$ **then**
    $S' \leftarrow S_c$;
  if SA accepts S' **then**
    $S \leftarrow S'$;
  if $f(S) > f(S^*)$ **then**
    $S^* \leftarrow S$;
  if S^* has not improved for many iterations **then**
    $S \leftarrow S^*$;
  Update the weights of operators;
until Terminal condition is met;
return S^*.

their VTWs and we attempt to insert each task under all the constraints stated above.

ALNS updates solutions through destroying and repairing. In the destroying process, some tasks are removed from the current solution by removal operators. The unscheduled and removed tasks are then inserted into the destroyed solution in the repairing process by insertion operators. There are six removal and three insertion operators: removal by random, min priority (tasks with lower priority are removed first), max opportunity (tasks with more VTWs are removed first), max conflict (tasks with higher conflict degree are removed first), cluster 1 (tasks in the orbits with fewest tasks are removed first) and cluster 2 (tasks in the orbits with the lowest priority are removed first); insertion by max priority, min opportunity and min conflict.

At each iteration, a pair of removal and insertion operators is selected according to their weights. The weights are updated adaptively according to the performance of operators in the previous iterations. A simulated annealing (SA) criterion is used to control the acceptance of new solutions.

Tabu search

Žulj, Kramer, and Schneider (2018) propose a method hybridizing ALNS with TS, and apply it to the order-batching problem. Their method combines the diversification capabilities of ALNS and the intensification capabilities of TS. It uses ALNS to search for better solutions and, if a certain number of ALNS iterations have passed, TS is used. Thus ALNS and TS are alternated in a two-stage manner. But since ALNS and TS are used in separate stages, this hybridization does not change the short-term cycling nature of ALNS.

In contrast, we propose a tight integration of ALNS with TS. We declare a removal tabu attribute for each task. Whenever one task is inserted into the current solution, the removal of this task is forbidden for $\sqrt{|T|/2}$ iterations. We use this strategy to prevent the algorithm re-visiting recent evaluated solutions. We compare the two ALNS–TS hybridizations in the experiments below.

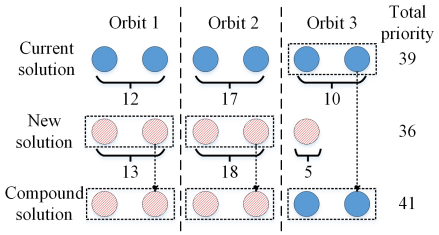

Figure 1: An example of partial sequence dominance

Partial sequence dominance

Due to the time-dependency and sequence-dependency, the quality of a solution is influenced significantly by its partial sequences. Inspired by genetic algorithms, we propose the PSD heuristic. When a new solution is produced, we compare a small part of it with the corresponding part of the current solution. In this paper we use the orbit as the small part. Figure 1 shows one example. In standard ALNS, the new solution is given up. However, Orbit 1 and Orbit 2 of the new solution are better than the current solution. So according to PSD, we keep Orbit 1 and Orbit 2 of the new solution and Orbit 3 of the current solution, and we get the compound solution, which is better than the current solution.

This paper studies a multi-orbit scheduling problem, which means one task might have multiple VTWs on different orbits. Therefore the compound solution might violate the uniqueness constraints (5). When a compound solution is produced, the feasibility is checked and the tasks that violate Constraints (5) are removed. If the repaired compound solution is better than the new solution, it is accepted. We note that for a MSOP problem, this check can be omitted.

Insertion positions ordering

In Liu et al. (2017), two strategies to select the observation start time are used: the earliest start time insertion and the middle start time insertion. According to their experiments, the middle start time insertion strategy works better. However, both of them waste too many insertion opportunities.

We propose an insertion position ordering (IPO) strategy to insert tasks. For every candidate task, we calculate all possible insertion positions. Due to the time-dependency and sequence-dependency, the difference of transition times in different insertion positions can be large. To increase the possibility of successful insertion without increasing the computation time too much, we calculate the possible transition time for each position and insert the task into the positions following an ascending order of possible transition times. The rationale is that time is a valuable resource, especially when we consider energy constraints, and it is better to use time for observation instead of transitioning.

Note because we cannot know the observation start time until we insert the task into the solution sequence, we cannot know the exact transition time. Therefore, we use the angles at the middle of the VTWs to compute an approximate transition time. This value is used to rank the possible positions.

Conflict degree

In Liu et al. (2017), the heuristic conflict degree (CD) is defined as 'the time that one VTW is overlapped with other scheduled tasks'. Since the solution is changed in every ALNS iteration, CD must be updated in each iteration. In order to reduce the computation time, we instead define CD as 'the time that one VTW is overlapped with any other VTWs'. This calculation can be done in the pre-processing phase. We show the quality of solutions is barely affected and the computation time is significantly decreased.

Empirical Results

The aim of the experiments is to assess the effectiveness of the proposed ALNS/TPI hybrid algorithm. Experiments were conducted using Intel Core i5 3.20GHz CPU running Windows 7 with 8GB memory. A time limit of 3600s is used. IBM ILOG CPLEX version 12.8 is used for MIP solving. The results for metaheuristics are the average of 20 runs.

The generation of the problem instances follows the configuration from Liu et al. (2017) except for the required minimum quality. In previous work, the required minimum quality c_i is set at 0 and this makes the quality constraints useless. Therefore in this paper we set it as a uniform random integer in $[5, 10]$ (hence, solution quality for the same-size instance in our experiments is lower than the one in Liu et al. (2017)).

The tasks are generated according to a uniform random distribution over two geographical regions: China and the whole world. For the Chinese area distribution mode, fifteen instances are designed and the number of tasks contained in these instances changes from 50 to 400, with an increment step of 25. For the worldwide distribution mode, twelve instances are designed and the number of tasks contained in these instances changes from 50 to 600, with an increment step of 50. Other parameters of the problem instances are: $M = 2400$, $E = 2400$, $m^o = 1$, $p^o = 1$, $p^s = 2$, $p^a = 1$, $\alpha^M = 0.6$, $\alpha^E = 0.8$, $a_1 = 1.5$, $a_2 = 2$, $a_3 = 2.5$, $a_4 = 3$.

Comparison of different algorithms First, we compare the proposed ALNS/TPI with our improved MIP model and the metaheuristics in Liu et al. (2017) ('old ALNS'), and the coarse ALNS–TS hybrid of Žulj, Kramer, and Schneider (2018) ('ALNS/TS'). The parameters of the ALNS algorithms are as in Liu et al. (2017): the total iteration time is 5000 and the simulated annealing parameter is 0.9975. In ALNS/TS, TS is run for 200 iterations after every 1000 iterations of ALNS. In each TS iteration, 10 neighbourhoods by our removal and insertion operators are examined to find the best local move. The whole process is run four times, for 12000 neighbourhood moves in total. Recently visited solutions are inserted in a tabu list for $\sqrt{|T|/2}$ iterations. Here, we implemented two versions of the modified (Žulj, Kramer, and Schneider 2018) algorithm. The first one is ALNS/TS for MSP without any further improvements. The second, called ALNS/TS/PI, has all the improvement features except the tight TS hybridization.

We compare the solution quality and the CPU time. The solution quality is the percentage of the total priority of scheduled tasks (i.e., the objective value) divided by the total priority of all the tasks in T. Figure 2 shows the comparison of the five different algorithms: our ALNS/TPI, old ALNS, ALNS/TS, ALNS/TS/PI, and our improved MIP. In Figure 2 left (for Chinese area) and middle (for worldwide), black solid lines show the solution quality (left axis) and the blue dash lines show the CPU time (right axis, log scale), showing that the CPU time of the ALNS/TPI increases slowly with the increasing number of tasks. The solution quality is significantly higher than that of the old ALNS and ALNS/TS. ALNS/TS/PI uses more time to produce solutions with worse quality, which proves that our integrated hybridization of ALNS and TS is better than the two-stage hybridization in Žulj, Kramer, and Schneider (2018) for this MSP problem. Furthermore, the implementation of our hybridization is easier than ALNS/TS because we only need to add tabu attributes of tasks to ALNS, while in ALNS/TS, a new TS search process is included. MIP can find optimal solutions for small-size instances but performs badly when the instance size gets large. For the three small instances with optimal solutions by MIP, ALNS/TPI also finds the same optimal solution. Among all the methods, old ALNS performs worst, consuming a long time to produce solutions with the lowest quality. Finally, Figure 2 right shows the anytime quality of different algorithms for instance 600_W with 600 tasks distributed worldwide. The MIP method fails to give a solution within the time limit for this large instance.

Second, in order to compare the improved MIP model with the two-stage MILP model in Liu et al. (2017), we fix the transition time of our model as 20s and remove the energy constraints ('MIP(20s)').

Figure 3 (top) shows the number of instances solved within 600s by different methods. The proposed ALNS/TPI, as well as ALNS/TS and ALNS/TS/PI can solve all the problem instances. The old ALNS, however, fails to solve three worldwide instances. MIP(20s) and the improved MIP can only solve small-size instances. If we set the time limit as 3600s (in Figure 3 (bottom)), the old ALNS and MIP(20s) can solve all the instances. The improved MIP can solve eight more instances. Unfortunately, the two-stage MILP model in Liu et al. (2017) fails to solve all the problem instances because of memory overflow. It cannot enumerate all the combinations of VTWs even for our smallest instance: the model can only solve problem with at most 12 tasks.

Comparison of different features of ALNS/TPI Last, in order to understand which features of ALNS/TPI contribute to its superior performance, we perform a factor analysis of features. The results are shown in Tables 1 and 2 (we also include the results of ALNS/TS, ALNS/TS/PI and old ALNS). We compare ALNS without PSD (ALNS/TI), ALNS without TS (ALNS/PI), ALNS without IPO (ALNS/TP) and ALNS with frequent CD update (ALNS/TPI/CD). All the results are compared with ALNS/TPI, so for other algorithms, we calculate the percentage of increase in quality (IQ, higher is better) and increase in time (IT, lower is better).

All the features contribute more to the solution quality for area distribution. Among all these features, IPO contributes most to the solution quality. However, IPO also increases

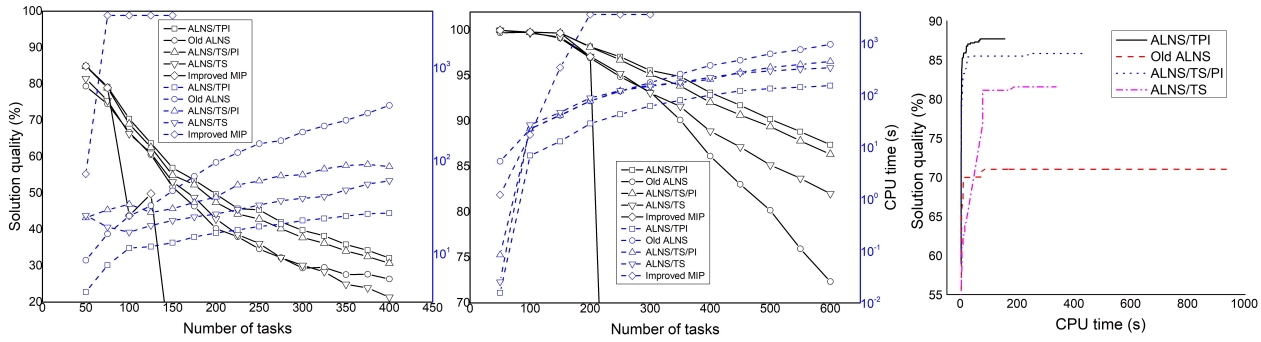

Figure 2: Comparison of algorithms on area distribution (left) and worldwide (middle), anytime quality of different algorithms (right)

Table 1: Results of different ALNSs for area distribution

Instance	ALNS/TPI		ALNS/TI		ALNS/PI		ALNS/TP		ALNS/TPI/CD		ALNS/TS		ALNS/TS/PI		Old ALNS	
	Quality/%	Time/s	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%
50_A	**84.94**	3.86	**0.00**	-7.12	-2.55	-0.43	-3.78	**-33.23**	-0.35	12.38	-4.26	84.91	-0.35	84.02	-7.01	54.34
75_A	**79.24**	7.47	-0.04	-6.19	-1.94	-1.95	-3.18	**-44.29**	-0.28	18.43	-5.12	61.14	-0.41	74.65	-6.38	53.94
100_A	**70.44**	11.42	-0.60	0.48	-3.24	-11.83	-5.14	**-73.69**	-0.56	27.69	-6.25	32.60	-2.04	66.18	-5.55	54.75
125_A	63.75	11.87	-0.07	8.30	-2.68	-11.66	-4.87	**-56.93**	**0.14**	27.07	-4.61	40.50	-1.92	57.04	-5.23	63.63
150_A	**56.95**	13.04	-1.42	5.87	-2.97	-6.76	-6.60	**-55.78**	-0.18	32.89	-7.34	42.23	-3.64	56.94	-10.29	72.23
175_A	53.57	15.05	-1.34	3.73	-2.62	-15.34	-4.62	**-51.04**	**0.24**	39.20	-9.81	39.04	-2.44	57.37	-15.55	77.68
200_A	49.70	16.59	-2.82	3.12	-4.41	-16.37	-6.46	**-44.88**	**0.26**	41.49	-15.94	37.34	-4.76	58.92	-23.49	82.54
225_A	45.73	17.93	-3.30	2.05	-3.45	-15.52	-5.57	**-47.45**	**0.62**	40.86	-18.44	39.51	-3.32	67.09	-20.38	85.19
250_A	**45.39**	19.49	-5.00	2.06	-4.76	-17.63	-9.13	**-40.99**	-0.23	45.70	-25.70	41.16	-5.89	67.58	-31.16	87.13
275_A	42.07	20.67	-3.36	3.78	-3.07	-11.25	-6.98	**-35.90**	**0.36**	52.38	-30.48	43.85	-4.57	69.73	-30.51	87.30
300_A	39.75	22.57	-2.69	0.71	-3.43	-14.01	-8.30	**-37.89**	**0.86**	54.34	-31.99	42.35	-5.32	67.76	-35.44	88.80
325_A	38.26	23.76	-2.62	1.98	-3.38	-11.75	-9.10	**-32.95**	**1.42**	61.09	-34.99	41.96	-5.67	70.65	-29.40	89.85
350_A	35.85	25.26	-2.32	0.73	-2.89	-9.79	-7.64	**-28.85**	**1.48**	65.26	-43.97	47.32	-5.25	71.27	-30.05	90.62
375_A	34.26	26.73	-3.45	-1.92	-2.67	-11.99	-6.95	**-33.75**	**1.70**	68.27	-43.32	51.84	-5.23	70.46	-23.84	91.68
400_A	32.15	27.13	-2.08	-1.20	-1.78	-3.14	-6.30	**-28.61**	**1.82**	68.84	-50.69	55.16	-4.82	68.40	-21.95	93.04
Avg.	51.47	17.52	-2.07	1.09	-3.06	-10.63	-6.31	**-43.08**	**0.49**	43.73	-22.19	46.73	-3.71	67.20	-19.75	78.18

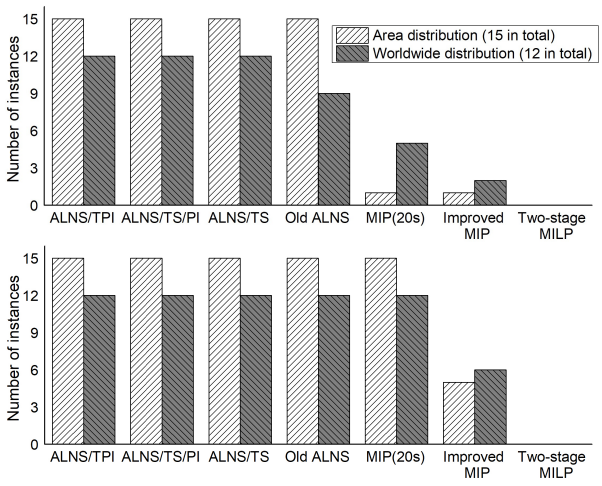

Figure 3: Number of instances solved within 600s (top) and 3600 (bottom)

the CPU time more compared with PSD and TS. TS works better than PSD, especially for the area distribution. This is because for area distribution, the distribution of tasks is dense and the CD of tasks is high. It is then more difficult for the algorithm to find a good solution. TS, which prevents the algorithm from searching recent solutions again, works better in this case. PSD works much better when the problem instance gets larger, which means that PSD performs well when the solution sequences get long. When the solution sequences get long, evaluating a solution only by its total quality gives up too much in-process information of partial sequences. ALNS/TPI/CD works better than ALNS/TPI for the area distribution because CD is an important heuristic influencing the quality for the dense distribution. However, the CPU time is nearly 3 times that of ALNS/TPI. From the results of ALNS/TPI/CD for the worldwide distribution, we can find that the frequent CD updates do not contribute to the solution quality all the time. The previous definition of CD focuses too much on the conflict with scheduled tasks while neglecting the potential conflict with unscheduled tasks.

Table 2: Results of different ALNSs for worldwide distribution

Instance	ALNS/TPI		ALNS/TI		ALNS/PI		ALNS/TP		ALNS/TPI/CD		ALNS/TS		ALNS/TS/PI		Old ALNS	
	Quality/%	Time/s	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%	IQ/%	IT/%
50_W	**100.00**	**0.02**	**0.00**	72.00	**0.00**	79.90	-2.17	99.41	**0.00**	44.74	**0.00**	38.12	**0.00**	81.31	-0.28	99.71
100_W	**99.78**	**6.91**	**0.00**	7.29	**0.00**	11.47	-1.49	10.81	**0.00**	19.30	-0.10	74.21	**0.00**	69.36	**0.00**	69.00
150_W	**99.70**	**12.92**	**0.00**	10.18	-0.20	14.16	-2.26	17.19	**0.00**	51.45	-0.47	72.48	**0.00**	68.92	-0.57	68.41
200_W	98.19	28.46	0.00	**-5.55**	-0.19	-0.14	-2.30	-2.69	**0.09**	45.27	-1.10	67.99	-0.04	63.56	-1.27	63.05
250_W	**97.08**	42.88	**0.00**	**-5.13**	-0.60	2.31	-2.38	-3.99	-0.05	45.47	-1.93	64.99	-0.35	64.56	-2.30	64.21
300_W	**95.58**	61.99	**0.00**	-3.49	-0.56	-8.02	-3.29	**-8.56**	-0.15	43.46	-2.67	58.59	-0.46	60.33	-2.56	64.50
350_W	**94.91**	79.99	**0.00**	-2.78	-1.11	**-22.33**	-4.36	-14.59	-0.19	48.61	-3.58	54.47	-1.09	54.06	-5.30	68.60
400_W	**93.14**	100.12	-0.06	-1.24	-0.99	**-24.80**	-4.90	-21.52	-0.20	51.80	-4.77	54.65	-1.17	50.43	-8.11	73.21
450_W	**91.73**	121.16	-0.23	-1.78	-0.98	-14.79	-5.48	**-25.84**	-0.24	56.07	-5.26	54.66	-1.17	56.20	-10.47	74.27
500_W	**90.22**	134.52	-0.54	1.55	-0.69	-5.23	-5.32	**-28.04**	-0.06	61.52	-5.93	54.19	-0.92	60.93	-12.50	78.54
550_W	**88.84**	142.44	-0.64	5.92	-0.84	-2.87	-4.79	**-20.21**	-0.11	66.28	-6.13	55.35	-1.15	63.52	-16.96	81.12
600_W	**87.41**	153.27	-0.85	6.66	-0.60	0.37	-5.38	**-20.74**	-0.17	69.63	-6.59	54.60	-1.22	65.77	-20.82	83.96
Avg.	**94.71**	73.72	-0.19	6.97	-0.56	2.50	-3.68	**-1.56**	-0.09	50.30	-3.21	58.69	-0.63	63.25	-6.76	74.05

Application in Real World

In this section, we discuss the difference between our simulations and the potential application of our work in the real world.

First, although the instances in our test instances are randomly generated, they are very similar to real-world ones. We do not have access to (often classified) instances of task locations. However, in our instances, except for task locations, all parameters are real: satellite's, orbits', Earth's parameters. The calculation of VTWs and transition time is based on the real geographical locations. There is little difference between our and classified instances, since in reality tasks are raised by the users and can be anywhere on the Earth surface. Further, tasks can be dense in a small area: we use the Chinese area distribution to simulate this. The number of tasks can be different on different days: we use different numbers of tasks to simulate this.

The satellite used in the simulation is called AS-01, which is the first AEOS of China. The scheduler of AS-01 was developed by the group of Liu et al. (2017), which uses several simple heuristic operators of the old ALNS to construct the observation schedule. The satellite has now been in orbit for more than two years and the scheduler has worked well until now. However, since the current scheduler is simple and greedy, the solutions generated by it are generally of lower quality than the ones generated by the old ALNS. But since our hybrid ALNS is much more efficient than the old ALNS, we believe it could improve on current operational procedures.

Another difference between our model and real-life satellites is the constraints of specific satellites. For example, for some satellites, the observation time in an orbit is bounded not only by memory and energy, but also by the maximum continuous working time of sensors and the maximum working temperature. If this information is known, it can be added as additional constraints to the proposed models.

Conclusions

We studied time-dependent multi-orbit agile Earth observation satellite scheduling, which is a complex real-world scheduling problem. We developed the first realistic mixed integer programming (MIP) model, and a novel hybridization of adaptive large neighbourhood search (ALNS) and tabu search (TS). As expected, MIP can find optimal solutions only for small-size instances. Extensive empirical results demonstrated that, compared with two state-of-the-art metaheuristic approaches, our proposed ALNS hybrid produces solutions with higher quality in less time. Factor analysis finds the novel insertion position ordering contributes most to the performance, but also consumes the most time. The partial sequence dominance heuristic performs better when the problem instance grows in size. The TS heuristic performs better when the conflict degree is high.

Our work proves that ALNS and TS hybridization is an efficient method for this satellite scheduling problem. Our next step is to evaluate the heuristics in this work on other similar problems. We believe these strategies can significantly improve other algorithms for problems characterized by time- and/or sequence-dependency.

Acknowledgements

We gratefully thank the SPARK 2018 reviewers for their valuable comments.

References

Bianchessi, N.; Cordeau, J.-F.; Desrosiers, J.; Laporte, G.; and Raymond, V. 2007. A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *European Journal of Operational Research* 177(2):750–762.

Demir, E.; Bektaş, T.; and Laporte, G. 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research* 223(2):346–359.

Dilkina, B., and Havens, B. 2005. Agile satellite scheduling via permutation search with constraint propagation. Technical report, Actenum Corporation, `www.cs.sfu.ca/CourseCentral/827/havens/papers/topic`.

Geng, X.; Li, J.; Yang, W.; and Gong, H. 2016. Agile satellite scheduling based on hybrid coding genetic algorithm. In *12th World Congress on Intelligent Control and Automation (WCICA)*, 2727–2731.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6(5):367–381.

Li, G.; Chen, C.; Yao, F.; He, R.; and Chen, Y. 2017. Hybrid differential evolution optimisation for earth observation satellite scheduling with time-dependent earliness-tardiness penalties. *Mathematical Problems in Engineering* 2017:Article ID 2490620.

Li, Y.; Xu, M.; and Wang, R. 2007. Scheduling observations of agile satellites with combined genetic algorithm. In *International Conference on Natural Computation*, 29–33.

Lin, W.-C.; Liao, D.-Y.; Liu, C.-Y.; and Lee, Y.-Y. 2005. Daily imaging scheduling of an earth observation satellite. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 35(2):213–223.

Liu, X.; Laporte, G.; Chen, Y.; and He, R. 2017. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers & Operations Research* 86:41–53.

Maillard, A. 2015. Flexible scheduling for an agile earth-observing satelllite. In *24th International Joint Conference on Artificial Intelligence (IJCAI)*, 4379–4380.

Pisinger, D., and Ropke, S. 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research* 34(8):2403–2435.

Wang, P.; Reinelt, G.; Gao, P.; and Tan, Y. 2011. A model, a heuristic and a decision support system to solve the Earth observing satellites fleet scheduling problem. *Computers & Industrial Engineering* 61(2):322–335.

Wolfe, W. J., and Sorensen, S. E. 2000. Three scheduling algorithms applied to the earth observing systems domain. *Management Science* 46(1):148–166.

Xu, R.; Chen, H.; Liang, X.; and Wang, H. 2016. Priority-based constructive algorithms for scheduling agile earth observation satellites with total priority maximization. *Expert Systems with Applications* 51:195–206.

Žulj, I.; Kramer, S.; and Schneider, M. 2018. A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research* 264(2):653–664.