Article

An Efficient Local Search Heuristic for Earth Observation Satellite Integrated Scheduling

Yuning Chen *, Ji Lu, Renjie He and Junwei Ou

College of Systems Engineering, National University of Defense Technology, Changsha 410004, China;
15580814918@163.com (J.L.); renjiehe@nudt.edu.cn (R.H.); junweiou@163.com (J.O.)
* Correspondence: cynnudt@hotmail.com

Received: 1 July 2020; Accepted: 20 July 2020; Published: 13 August 2020

Abstract: Earth observation satellites (EOs) are taking a large number of pictures with increasing resolution which produce massive image data. Satellite data transmission becomes the bottleneck part in the process of EOS resource management. In this paper, we study the earth observation satellite integrated scheduling problem (EOSIS) where the imaging activities and download activities are considered integratively. We propose an integer linear programming model to formulate the problem. Due to the NP-hardness of the problem, we propose an efficient local search heuristic (ELSH) to solve problems of large size. ELSH uses a dedicated local search method to guarantee algorithm performance and efficient constraint handling mechanisms to guarantee algorithm efficiency. Numerical experimental results show that the algorithm demonstrates excellent performance on a set of benchmark instances. The ELSH achieves optimal results for all small-size instances (with 50 targets, two satellites, and three ground stations), and is very robust for large instances with up to 2000 targets. Moreover, the proposed ELSH easily dominates the state-of-the-art algorithm.

Keywords: earth observation satellite; local search; integrated scheduling; satellite data transmission

1. Introduction

Earth observation satellites (EOS) are playing an increasingly important role in economic development, disaster countering, surveillance, and many other fields for both military and civilian purposes [1–3]. Satellites take pictures by their optical devices and send the data back to the ground for various purposes. With the development of technology, the resolution of the satellite camera is getting higher and higher. The massive image data impose a heavy burden on the transmission of data from satellites to the ground. Therefore, it is important to consider data transmission when managing satellite imaging activities.

Satellites need more time to send the data back to the ground, and data transmission management is certainly an important part of the satellite scheduling. In reality, the scheduling of EOSs with separate considerations for imaging and data transmission is quite difficult because the two operations are highly coupled [4]. There are many studies on the scheduling of satellites, but few have considered the data transmission activities integrated with the imaging ones. Most studies consider data transmission and imaging activities individually. The data transmission plan is usually created after the plan of imaging activities is determined. The satellite scheduling problem is actually divided into two subproblems: imaging scheduling given a set of user requests and data transmission scheduling given an imaging plan. Such a scheme will be easily trapped in a local optimum and therefore influences the final outcome of the algorithm.

In this paper, we consider the earth observation satellite integrated scheduling problem (EOSIS), whose output is an imaging plan as well as a data transmission plan. We first provide an integer programming model to formulate the problem. As EOSIS is a very complicated problem and an exact
approach can only solve problem instances of very limited size, we propose an efficient local search heuristic (ELSH) to solve the problem of realistic size. We also present numerical studies to illustrate the excellent performance of the proposed ELSH.

The rest of the paper is organized as follows. Section 2 presents a detailed literature review on relative topics. Section 3 is the problem description. The proposed ELSH is introduced in Section 4 and experiments are presented in Section 5.

2. Literature Review

The earth observation satellite scheduling problem has attracted intensive research attention over the last two decades. They mostly focused on the scheduling of imaging activities. Ref. [5] proved that the scheduling of earth observation satellites is an NP-hard problem and proposed three simple heuristic approaches to solve the problem. Ref. [6] proposed a branch and bound algorithm for onboard satellite scheduling, and Ref. [7] used branch and price to solve the imaging scheduling problem. These exact algorithms can only deal with small-sized problems. Ref. [8] considered a simplified version of the problem and formulated the problem as a MIP model. Ref. [8] introduced several useful cuts to accelerate the solving process.

Due to the complexity and problem scale, most studies investigate the application of heuristics and meta-heuristics to solve the problem, including genetic algorithm [9–12], adaptive large neighborhood search [13], and other heuristics [14–16]. For a review of the planning and scheduling techniques for the EOS scheduling problem, readers could refer to [17].

The satellite data transmission scheduling problem is somewhat less studied. [18] introduced the basic concept and characteristics of the satellite data transmission problem. [19] proposed to use a genetic algorithm to solve the problem, and [20] adopted an ant colony algorithm. [21] studied the case where satellites can communicate and transfer data with each other. [22] studied a multiobjective variation of EOSIS. The solution methods applied to this problem mainly focus on heuristics and metaheuristics, due to its NP-hardness nature.

There are few studies on the EOS integrated scheduling where imaging and data transmission are considered simultaneously. Ref. [23] proposed a two-phase genetic annealing method for the EOSIS problem, where a genetic algorithm and a simulated annealing algorithm are combined to solve the problem. However, general-purpose metaheuristics, without adaptation, are not able to perform well on such a complex problem. Dedicated heuristics are usually needed in order to achieve a satisfactory result.

The ELSH we propose adopts several dedicated operators to improve algorithm performance. An efficient constraint handling mechanism is also incorporated to guarantee algorithm efficiency.

3. Model

3.1. Problem Description

The scheduling of earth observation satellite is an NP-hard problem [5]; therefore, the EOSIS problem is also an NP-hard problem. Basically, satellites need to orbit around the earth and take pictures when they pass by a requested ground target. We call such a time interval when a requested target is visible to a satellite as a visible time window (VTW). Then, satellites download the image data to the ground when they pass by a ground station. Similarly, we define the time interval when a satellite is visible to a ground station as a download window (DW). The problem is defined by the following components:

1. a set of satellite resources \( A \). Each satellite \( a \) is associated with an onboard memory capacity \( m_a \), an imaging data rate \( p_a \), a transmission data rate \( b_a \) and a set of download time windows (DW) \( O_a \).
2. a set of user observation request \( R \). Each request \( r \) is characterized by an observation profit \( p_r \), a minimal observation duration \( d_r \), and a set of visible time window (VTW) \( V_r \).
The EOSIS problem consists of selecting a subset of user requests and arranging observations within their available VTWs. The objective is to maximize observation profit without violating any constraint.

3.2. Mathematical Formulation

An observation activity is to be arranged within a VTW and a download activity within a DW. The basic elements of scheduling are VTWs and DWs. We use \( V \) to represent the union of available VTWs of all user requests and \( D \) to represent the union of all download windows. For simplicity, we define \( N = V \cup D \). A VTW \( i \) is associated with the feature of the corresponding request: \( p_i, d_i \). \( p_i \) is the profit, and \( d_i \) is the minimum duration to accomplish the observation. For a satellite \( a \) to accomplish the request, the total occupied memory is the imaging data rate times imaging duration, \( d_i p_a \). The total time required for downloading the image is \( \frac{d_i p_a}{T_R} \). The start and end of a VTW or a DW \( i \) are denoted as \( s_i \) and \( e_i \).

We use matrix \( H \) to identify the associated user request of a VTW and \( H_{ia} \) equals 1 when VTW \( i \) is associated with request \( r, 0 \) otherwise. Matrix \( U \) indicates the associated satellite and \( U_{ia} \) equals 1 if VTW \( i \) is associated with satellite \( a, 0 \) otherwise. As the VTW and DW are continuous time intervals, we discretize these time windows uniformly into \( K \) time points and we define \( K = \{1, 2, ..., K\} \). \( T_{T_i}^k \) is the time required for a EOS to maneuver from \( i \) to \( j \), starting at the \( k_{th} \) time point of \( i \).

We use the following decision variables to build our MIP model:

1. \( y_i \): is a binary variable that equals 1 if window \( i \) (VTW or DW) is included in the observation plan, 0 otherwise.
2. \( x_{ij} \): is a binary variable which equals 1 if a satellite reaches window \( j \) after \( i \), 0 otherwise.
3. \( z_{ij} \): is a binary variable which equals 1 if a satellite visits window \( j \) after \( i \), leaving \( i \) at the \( k_{th} \) time point of \( i \), 0 otherwise.
4. \( s_i \): indicates the time instant at which the observation or download activity at window \( i \) starts.
5. \( w_i \): indicates the memory level of a EOS at VTW \( i \).
6. \( t_i \): indicates the transmission duration of a download activity \( l \).

Given the above definitions, we present an integer linear programming (ILP) model for the EOSIS problem:

\[
\text{Maximize : } \sum_{i \in I} p_i y_i \quad (1)
\]

subject to:

\[
\sum_{j \in V} x_{ji} - \sum_{i \in V} x_{ij} = 0, \quad \forall i \in N \quad (2)
\]

\[
\sum_{i \in V} H_{ir} y_i \leq 1, \quad \forall r \in R \quad (3)
\]

\[
\sum_{k \in K} (z^k_{ij} T_{T_i}^k) + d_i + s_i \leq s_j + (1 - x_{ij}) M, \quad \forall i, j \in N \quad (4)
\]

\[
l_i \leq s_i \leq u_i, \quad \forall i \in N \quad (5)
\]

\[
0 \leq w_j \leq w_{l} - d_i p_a + (1 - x_{ij}) M, \quad \forall i \in N, j \in V \quad (6)
\]

\[
w_i + t_i b_a - (1 - x_{ij}) M \leq w_j \leq w_{l} + t_i b_a + (1 - x_{ij}) M, \quad \forall i \in N, j \in D \quad (7)
\]

\[
\sum_{k \in K} z^k_{ij} = x_{ij}, \quad \forall i, j \in N \quad (8)
\]

\[
\sum_{j \in N} x_{ij} = y_i, \quad \forall i \in N \quad (9)
\]

The goal of the EOS scheduling problem is to maximize the total profit. Constraint 3 ensures the integrity and validity of the observation schedule. Constraint 4 indicates the fact that each target should only be observed once.
Time constraints is modeled by \( \text{Constraint 5} \) and \( \text{Constraint 6} \). \( \text{Constraint 5} \) models the time consumed by maneuver operation and imaging process. \( \text{Constraint 6} \) is the VTW constraint.

Memory constraints are modeled by \( \text{Constraint 7} \) and \( \text{Constraint 8} \). \( \text{Constraint 7} \) models the memory consumed by imaging process. \( \text{Constraint 8} \) models the change of onboard memory level by download activity.

Constraints \( \text{9} \) and \( \text{10} \) define the relationship between three binary decision variables.

Given the above ILP model, the preliminary experimental results showed that CPLEX is able to solve instances with very limited size given a time limit of five hours. To solve large-sized real-life problems, we resort to a heuristic local search method introduced in the next section.

4. An Efficient Local Search Heuristic for Earth Observation Satellite Integrated Scheduling

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

4.1. A Fast Constraint Evaluation Mechanism

In [24] a fast constraint verification mechanism was proposed by introducing the concept of Valid Observation Time Interval (VOTI). The VOTI allows fast verification of VTW constraints for insertion operation. We extend this idea in the context of memory constraint and constraint violation so that the constraint can be verified and evaluated with \( O(1) \) complexity in case of violation.

4.1.1. VTW Constraint

In [24] the concept of Valid Observation Time Interval (VOTI) was proposed, which allows fast verification of the VTW constraint for insertion neighborhood in the context of satellite imaging scheduling. For each observation, its VOTI is a time interval during which at least one feasible operation and imaging process. Constraint \( 6 \) is the VTW constraint.

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

In [24] the concept of Valid Observation Time Interval (VOTI) was proposed, which allows fast verification of the VTW constraint for insertion neighborhood in the context of satellite imaging scheduling. For each observation, its VOTI is a time interval during which at least one feasible operation and imaging process. Constraint \( 6 \) is the VTW constraint.

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

4.1.1. VTW Constraint

In [24] the concept of Valid Observation Time Interval (VOTI) was proposed, which allows fast verification of the VTW constraint for insertion neighborhood in the context of satellite imaging scheduling. For each observation, its VOTI is a time interval during which at least one feasible operation and imaging process. Constraint \( 6 \) is the VTW constraint.

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

In [24] a fast constraint verification mechanism was proposed by introducing the concept of Valid Observation Time Interval (VOTI). The VOTI allows fast verification of VTW constraints for insertion operation. We extend this idea in the context of memory constraint and constraint violation so that the constraint can be verified and evaluated with \( O(1) \) complexity in case of violation.

4.1.1. VTW Constraint

In [24] the concept of Valid Observation Time Interval (VOTI) was proposed, which allows fast verification of the VTW constraint for insertion neighborhood in the context of satellite imaging scheduling. For each observation, its VOTI is a time interval during which at least one feasible operation and imaging process. Constraint \( 6 \) is the VTW constraint.

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

Given the above ILP model, the preliminary experimental results showed that CPLEX is able to solve instances with very limited size given a time limit of five hours. To solve large-sized real-life problems, we resort to a heuristic local search method introduced in the next section.

4. An Efficient Local Search Heuristic for Earth Observation Satellite Integrated Scheduling

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

4.1. A Fast Constraint Evaluation Mechanism

In [24] a fast constraint verification mechanism was proposed by introducing the concept of Valid Observation Time Interval (VOTI). The VOTI allows fast verification of VTW constraints for insertion operation. We extend this idea in the context of memory constraint and constraint violation so that the constraint can be verified and evaluated with \( O(1) \) complexity in case of violation.

4.1.1. VTW Constraint

In [24] the concept of Valid Observation Time Interval (VOTI) was proposed, which allows fast verification of the VTW constraint for insertion neighborhood in the context of satellite imaging scheduling. For each observation, its VOTI is a time interval during which at least one feasible operation and imaging process. Constraint \( 6 \) is the VTW constraint.

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

In [24] the concept of Valid Observation Time Interval (VOTI) was proposed, which allows fast verification of the VTW constraint for insertion neighborhood in the context of satellite imaging scheduling. For each observation, its VOTI is a time interval during which at least one feasible operation and imaging process. Constraint \( 6 \) is the VTW constraint.

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.

In [24] the concept of Valid Observation Time Interval (VOTI) was proposed, which allows fast verification of the VTW constraint for insertion neighborhood in the context of satellite imaging scheduling. For each observation, its VOTI is a time interval during which at least one feasible operation and imaging process. Constraint \( 6 \) is the VTW constraint.

In this section, we present an efficient heuristic for satellite integrated scheduling. First, we introduce a fast evaluation mechanism where constraint violation can be identified and evaluated with \( O(1) \) complexity. Then, we propose a local search heuristic to solve the EOSIS problem. The fast evaluation mechanism serves as a core component to guarantee the efficiency of the algorithm.
Each time a new VTW is inserted, it is necessary to update the value of $ES(v)$ for each $v$ before $v_1$ and the value of $LS(v)$ for each $v$ after $v_2$. Note that, if a VTW violation is related to a VTW, $v$, then $[ES(v), LS(v)]$ is reduced to a number and $ES(v) = LS(v)$.

### 4.1.2. Memory Constraint

The evaluation of memory constraint is relatively easy since it is not related to transition activity. Let $M(v)$ be the memory level at current window $v$ and $MV(v)$ be the violation of memory constraint. For an insertion which inserts $v_2$ after $v_1$, the memory level and violation at $v_2$ can be calculated as

\[
MV(v_2) = MV(v_1) + \Delta MV
\]

\[
M(v_2) = M(v_1) - MC + \Delta MV
\]

where $\Delta MV = \max(0, MC - M(v_1))$. $MC$ is the memory consumed by $v_2$. $MC = d_2 p_a$ if $v_2$ is an observation window. Note that download activity inevitably increases the memory level onboard and $MC = -t_l b_a$ (negative value) if $v_2$ is a download window.

### 4.2. Evaluation of a Schedule

Given the above feature values, we can evaluate a schedule $\sigma$ easily as:

\[
Eva(\sigma, c) = \sum_{v \in \sigma} p_v + c \rho_1 TV(\sigma) + c \rho_2 MV(\sigma)
\]

where $\rho_1, \rho_2$ are penalty parameters and $c$ is a penalty parameter multiplier. The evaluation of a schedule is the sum of the total profit and penalties on constraint violation and resource consumption. In this way, we can evaluate an insertion with $O(1)$ complexity and improve the efficiency of our algorithm.

### 4.3. An Efficient Heuristic for EOS Integrated Scheduling

In this section, we present an efficient heuristic for the EOSIS problem. We first introduce its general framework and then several main components.

#### 4.3.1. General Framework

The proposed heuristic is essentially a stochastic local search algorithm [25]. Starting from an initial solution provided by a greedy construction heuristic, the algorithm alternates between an improvement phase and a perturbation phase. In the improvement phase, the algorithm conducts a directed and focused search by accepting only improving insertions. When a local optimum is reached in the improvement phase, the algorithm triggers a perturbation phase to help the search process escape from the local trap and restarts an improvement procedure in a different zone of the search space. Such an Improvement-Perturbation cycle is iterated for $\pi$ rounds. The best solution is updated each time a new local optimum is attained at the end of the improvement phase. The first half of the iterations are run with a small violation penalty multiplier $c_1$ to diversify the search. While for the other half of the iterations, the algorithm is run with a large violation penalty $c_2$ to guarantee the feasibility of the resulting schedule. The general framework of the SFHA is shown in Algorithm 1.

---

**Algorithm 1. Main Procedure of Our Algorithm**

**Require:** P: An AEOS scheduling problem; $\pi$: the max number of iterations allowed for Improvement-Perturbation;

**Ensure:** The best solution $\sigma^*$

1. $\sigma \leftarrow \text{Greedy-Construction}(P)$
2. $\sigma^* \leftarrow \sigma$
3. $\text{count} \leftarrow 0$

---
4.3.2. Solution Initialization

The initial solution is produced by a greedy construction algorithm. The algorithm first sorts all VTWs (associated with requests) in decreasing order of their profits and in ascending order of their start time. For each VTW, the algorithm arranges observation of the associated request at the earliest possible start time that satisfies all imperative constraints. If the memory level drops below zero at a new VTW, the algorithm finds the nearest download window and inserts a download activity into the schedule. Observation activities that conflict with the download activity are then removed from the schedule to ensure the feasibility. This process is terminated until all requests are examined.

4.3.3. The Improvement Procedure

The aim of the improvement procedure is to attain a local optimum solution (as good as possible) in a given neighborhood. In this paper, we propose to use an Insertion neighborhood, which is induced by a Insertion operator. It can also be regarded as a hill-climbing process [26–28].

- Insertion: insert a window (VTW or DW) into a schedule. Constraint violation is allowed and the amount of violation is evaluated when this occurs.

The pseudo-code of the improvement procedure is shown in Algorithm 2. The algorithm repeatedly considers all possible insertions and chooses the best one. Such a greedy strategy can help the solution achieve better quality at the current stage. The algorithm continues until no improvements can be found (improvement is evaluated according to Equation (17)).

Algorithm 2. The Improvement Procedure

Require: P: An AEOS scheduling problem; σ: a solution for improvement; c: a violation multiplier;
Ensure: The local optimum solution σ*

1: while True do
2: Collect all windows that are not included in σ and denote the set as S;
3: bestScore ← 0;
4: for each window v in S do
5: Insert v into σ and the new solution is σ';
6: if Eva(σ', c) − Eva(σ, c) > bestScore then
7: bestScore ← Eva(σ', c) − Eva(σ, c);
8: end if
9: end for
10: if bestScore > 0 then
11: Insert the window with the best score;
12: else
13: Break;
14: end if
15: end while
16: return $\sigma^*$;

4.3.4. The Perturbation Phase

When the algorithm is trapped in a local optimum, a mechanism is needed to help the search escape from the trap and restart the search from a new starting point. To establish a more global form of diversification, and thereby reinforce the capacity of the algorithm to explore unexplored areas in the search space, we propose a dedicated perturbation mechanism.

The general idea of the proposed perturbation is to remove a set of selected requests within the solution so that new insertions may be possible. To do this, we resort to a set of destroy operators coupled with a tabu mechanism. Specifically, we use four remove operators:

- Random removal: Select $q$ requests from a given solution (schedule) randomly and remove them;
- Priority removal: Iteratively removes the request with the lowest profit until the request bank is full;
- Opportunity removal: Iteratively removes the request with the largest number of VTWs; ties are first broken by profit (smaller profit requests are prioritized) and then randomly;
- Conflict removal: Iteratively removes the request with the largest conflict degree, which is provided by a conflicting measure proposed in [29].

Removed requests are taboo and not allowed to be inserted into the solution in the next iteration. Each time when a perturbation is triggered, it is realized by the combined use of a removal operator randomly selected from the four candidates.

5. Experiment

In this section, we present the numerical results. First, we report the algorithm performance on small instances. Then, we report the comparative results with a state-of-the-art algorithm on larger instances.

The performance of the proposed ELSH algorithm relies on a set of correlated parameters. To achieve a reasonable tuning of the parameters and a good algorithm performance, we adopt the iterated F-race (IFR) method, which allows an automated parameter configuration [30]. Table 1 presents the final values for each parameter, together with the range of values determined by preliminary experiments. We use the two-satellite 1000-node instances of area distribution as the training set, and the tuning budget is set to be 1000 runs of ELSH.

| Description                     | Test Range | Value |
|---------------------------------|------------|-------|
| $\pi$ max iteration number     | [50, 300]  | 150   |
| $c_1$ small penalty multiplier  | [0.1, 2.5] | 1     |
| $c_2$ large penalty multiplier  | [1, 25]    | 10    |
| $\rho_1$ penalty for VTW violation | [10, 50] | 15    |
| $\rho_2$ penalty for memory violation | [10, 50] | 25    |
The experiment is coded in C++ and conducted on a Linux cluster system with an AMD Opteron 4184 processor (2.8 GHz and 2 GB RAM) running Ubuntu 12.04. All the results presented in this section is an average of 10 independent runs.

5.1. Problem Instances

We extend the benchmark instances proposed by [13] by including three well-known Chinese ground stations, namely JiaMusi, KaShi, and SanYa, whose precise locations can be found in [23]. In these instances, the VTWs for thousands of targets are provided. The benchmark set consists of two distributions: area distribution and world distribution. Targets are generated randomly in a square area around China in the area distribution. Figure 1 presents an example of area distribution with 200 targets. The world distribution means the targets are selected randomly around the surface of the earth.

![Figure 1. An example for the benchmark instances of area distribution.](image)

For each distribution, ten different scenarios with different numbers of targets are considered. Each target is associated with a set of properties: (i) profit, (ii) minimal duration for observation, (iii) time window, (iv) memory and energy consumption of observation, (v) a set of VTW for each EOS, and (vi) observation angle for every time point in a VTW. The number of targets varies from 100 to 1000 with a step of 100 for area distribution and from 200 to 2000 with a step of 200 for world distribution. The satellite number varies from 2 to 6 for each scenario. This makes 50 instances for each distribution. As CPLEX can only solve small-size instances, we create a set of new instances following the method of [13], each of which contains 50 targets and two satellites for area distribution. The ground stations are still JiaMusi, KaShi, and SanYa.

5.2. Results on Small Instances

First, we solve the model introduced in Section 3 by CPLEX on the small instances we created. The VTWs are discretized with a step of 10 s when using CPLEX (e.g., a VTW of 100 s contains 11 time points). We compare the results with those of the proposed heuristic algorithm, and the results are shown in Table 2.
Table 2. Comparison with CPLEX on the set of 2-satellite, 50-node area distribution instances. Columns CPLEX and ELSH give the optimal solution found by CPLEX and the average result of ELSH, respectively. All the results of these two columns are associated with two numbers. The first number is the total number of targets observed and the second number is the final profit achieved. Columns Dev show the deviation between the best value found by ELSH algorithm and the optimal value found by CPLEX (100% × CPLEX−ELSH

| Instance ID | CPLEX       | ELSH       | Dev(%) |
|-------------|-------------|------------|--------|
| 1           | 47/231      | 47/231     | 0.0    |
| 2           | 50/249      | 50/249     | 0.0    |
| 3           | 48/251      | 48/251     | 0.0    |
| 4           | 43/220      | 43/220     | 0.0    |
| 5           | 49/263      | 49/263     | 0.0    |
| 6           | 47/233      | 47/233     | 0.0    |
| 7           | 50/269      | 50/269     | 0.0    |
| 8           | 49/248      | 49/248     | 0.0    |
| 9           | 44/221      | 44/221     | 0.0    |
| 10          | 47/236      | 47/236     | 0.0    |

As shown in Table 2, the proposed ELSH algorithm can achieve optimal results for all instances. The average running time of CPLEX are 2.1 h while the average running time of the ELSH algorithm is 2.4 s.

From the results, we conclude that the proposed ELSH is able to handle small instances very efficiently and achieve optimal results easily.

5.3. Comparison with the State-of-the-Art Algorithm

To further validate the effectiveness of the proposed ELSH algorithm, we compare the ELSH algorithm with the state-of-the-art two-phase genetic annealing algorithm (TPGA) proposed by [23], which, to the best of our knowledge, is the most recent and best-performing algorithm which was designed to solve exactly the same problem as is studied in this paper.

Tables 3 and 4 present respectively the comparison of observed target number and total profit achieved by the two algorithms on area distribution instances. Tables 5 and 6 show the results for instances of world distribution.

Table 3. Comparison with TPGA on the observed target number on area distribution instances.

| Size  | 2 Satellites | 3 Satellites | 4 Satellites | 5 Satellites | 6 Satellites |
|-------|--------------|--------------|--------------|--------------|--------------|
|       | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA |
| 100   | 80   | 70   | 82   | 72   | 84   | 73   | 85   | 74   | 86   | 74   |
| 200   | 158  | 141  | 167  | 147  | 168  | 146  | 169  | 149  | 170  | 152  |
| 300   | 227  | 200  | 246  | 215  | 254  | 223  | 256  | 225  | 259  | 228  |
| 400   | 282  | 244  | 336  | 288  | 340  | 301  | 344  | 303  | 347  | 305  |
| 500   | 317  | 281  | 400  | 349  | 416  | 368  | 426  | 369  | 438  | 382  |
| 600   | 342  | 304  | 466  | 407  | 508  | 434  | 515  | 442  | 516  | 454  |
| 700   | 351  | 304  | 515  | 455  | 589  | 525  | 590  | 527  | 593  | 538  |
| 800   | 367  | 324  | 552  | 495  | 669  | 586  | 678  | 592  | 686  | 597  |
| 900   | 382  | 336  | 574  | 501  | 748  | 643  | 755  | 651  | 767  | 690  |
| 1000  | 386  | 341  | 601  | 527  | 780  | 689  | 790  | 712  | 875  | 778  |
Table 4. Comparison with TPGA on the total profit on area distribution instances.

| Size  | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA |
|-------|------|------|------|------|------|------|------|------|------|------|------|------|
| 100   | 443  | 387  | 454  | 399  | 464  | 405  | 469  | 413  | 476  | 409  |
| 200   | 883  | 785  | 925  | 814  | 920  | 808  | 937  | 828  | 941  | 842  |
| 300   | 1298 | 1141 | 1350 | 1178 | 1391 | 1222 | 1399 | 1215 | 1419 | 1253 |
| 400   | 1654 | 1434 | 1837 | 1575 | 1856 | 1642 | 1879 | 1630 | 1881 | 1643 |
| 500   | 1945 | 1722 | 2101 | 1923 | 2138 | 2041 | 2363 | 2046 | 2374 | 2116 |
| 600   | 2231 | 1984 | 2429 | 2228 | 2737 | 2483 | 2878 | 2470 | 2883 | 2541 |
| 700   | 2387 | 2066 | 2611 | 2384 | 3012 | 2775 | 3337 | 2930 | 3388 | 2934 |
| 800   | 2522 | 2235 | 2850 | 2492 | 3147 | 2775 | 3532 | 3152 | 3782 | 3345 |
| 900   | 2760 | 2427 | 3060 | 2679 | 3468 | 2975 | 3781 | 3406 | 3925 | 3585 |
| 1000  | 2875 | 2540 | 3270 | 2830 | 3761 | 3141 | 4117 | 3610 | 4287 | 3853 |

From the result of area distribution, we see that ELSH easily dominates TPGA. For both algorithms, the number of the accomplished user requests and the profit increases as the problem instance becomes larger, but ELSH accomplishes more user requests and produces more profit overall instances. The average improvement compared to TPGA is 13.75%.

We draw a similar conclusion on world distribution instances. ELSH shows a great advantage over the TPGA. The average improvement compared to TPGA is up to 13.68%.

In conclusion, the ELSH demonstrates excellent performance on all benchmark instances and easily dominates the state-of-the-art algorithm.

Table 5. Comparison with TPGA on the observed target number on world distribution instances.

| Size  | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA |
|-------|------|------|------|------|------|------|------|------|------|------|
| 200   | 162  | 140  | 165  | 142  | 166  | 144  | 168  | 146  | 170  | 148  |
| 400   | 328  | 291  | 328  | 293  | 329  | 292  | 332  | 291  | 336  | 291  |
| 600   | 488  | 433  | 495  | 434  | 499  | 444  | 509  | 451  | 515  | 455  |
| 800   | 665  | 580  | 672  | 589  | 681  | 593  | 684  | 600  | 691  | 615  |
| 1000  | 820  | 710  | 828  | 715  | 831  | 728  | 840  | 728  | 847  | 759  |
| 1200  | 969  | 859  | 980  | 867  | 1021 | 881  | 1029 | 882  | 1039 | 898  |
| 1400  | 1126 | 1004 | 1153 | 1019 | 1191 | 1039 | 1195 | 1065 | 1225 | 1086 |
| 1600  | 1282 | 1146 | 1325 | 1165 | 1353 | 1174 | 1355 | 1212 | 1401 | 1230 |
| 1800  | 1435 | 1259 | 1502 | 1340 | 1518 | 1345 | 1543 | 1356 | 1574 | 1371 |
| 2000  | 1571 | 1391 | 1689 | 1457 | 1692 | 1471 | 1697 | 1536 | 1717 | 1551 |

Table 6. Comparison with TPGA on the total profit on world distribution instances.

| Size  | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA | ELSH | TPGA |
|-------|------|------|------|------|------|------|------|------|------|------|
| 200   | 851  | 736  | 866  | 747  | 872  | 770  | 880  | 764  | 892  | 776  |
| 400   | 1753 | 1557 | 1755 | 1569 | 1758 | 1572 | 1790 | 1622 | 1809 | 1618 |
| 600   | 2689 | 2389 | 2727 | 2395 | 2751 | 2463 | 2808 | 2485 | 2841 | 2510 |
| 800   | 3668 | 3198 | 3705 | 3247 | 3752 | 3191 | 3772 | 3306 | 3810 | 3390 |
| 1000  | 4475 | 3945 | 4483 | 3928 | 4555 | 4042 | 4604 | 4098 | 4647 | 4159 |
| 1200  | 5389 | 4856 | 5290 | 4677 | 5511 | 4754 | 5554 | 4762 | 5598 | 4847 |
| 1400  | 6068 | 5515 | 6200 | 5479 | 6403 | 5587 | 6425 | 5727 | 6591 | 5840 |
| 1600  | 7007 | 6261 | 7183 | 6316 | 7331 | 6362 | 7343 | 6569 | 7595 | 6597 |
| 1800  | 7935 | 6963 | 8246 | 7259 | 8254 | 7289 | 8360 | 7337 | 8530 | 7426 |
| 2000  | 8186 | 7168 | 8209 | 7283 | 8430 | 7417 | 8554 | 7745 | 8658 | 7872 |

From the result of area distribution, we see that ELSH easily dominates TPGA. For both algorithms, the number of the accomplished user requests and the profit increases as the problem instance becomes larger, but ELSH accomplishes more user requests and produces more profit overall instances. The average improvement compared to TPGA is 13.75%.

We draw a similar conclusion on world distribution instances. ELSH shows a great advantage over the TPGA. The average improvement compared to TPGA is up to 13.68%.

In conclusion, the ELSH demonstrates excellent performance on all benchmark instances and easily dominates the state-of-the-art algorithm.
6. Conclusions

In this paper, we study the earth observation satellite integrated scheduling problem (EOSIS) which considers the imaging activity and data transmission activity integratively. This is a realistic problem and includes complicated constraints. We develop an integer linear programming model to formulate the problem. With this model, we used CPLEX to solve the problem, and the results show that CPLEX is only able to solve a small-sized problem.

Due to the complexity of the problem, we propose an efficient local search heuristic algorithm (ELSH) to solve instances of large size. The proposed ELSH is characterized by an efficient constraint handling mechanism to improve algorithm speed and a dedicated improvement–perturbation procedure to guarantee the quality of results.

Experiments were conducted to illustrate the excellent performance of the proposed ELSH. In particular, it is able to achieve optimal solution quickly for small instances with less than 50 targets and two satellites, and dominates the state-of-the-art algorithm on large instances with up to 2000 targets.

For future work, we would like to investigate more problem-tailored heuristics and further improve the performance of the algorithm.

Author Contributions: Writing, Y.C. and J.O.; original draft preparation, Y.C.; methodology, Y.C.; software, J.L.; data curation, J.L.; development formal analysis and supervision, R.H.; review and editing, J.O. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Natural Science Foundation of China, Grant No. 71701203 and the National Natural Science Foundation of Hunan Province, Grant No. 2018JJ3618.

Acknowledgments: The authors thank the editor and the anonymous reviewers for their very helpful and constructive comments.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the works reported in this paper.

References

1. Davies, A.G.; Chien, S.; Tran, D.; Doubleday, J. The NASA Volcano Sensor Web, advanced autonomy and the remote sensing of volcanic eruptions: A review. Geol. Soc. 2016, 426, 137–158. [CrossRef]
2. Klumpar, D.; Spence, H.; Larsen, B.; Blake, J.; Springer, L.; Crew, A.; Mosleh, E.; Mashburn, K. FIREBIRD: A dual satellite mission to examine the spatial and energy coherence scales of radiation belt electron microbursts. In Proceedings of the AGU Fall Meeting Abstracts, San Francisco, CA, USA, 14–18 December 2009.
3. Salvi, S.; Stramondo, S.; Funning, G.J.; Ferrettic, A.; Sartid, F.; Mouratidis, A. The Sentinel-1 mission for the improvement of the scientific understanding and the operational monitoring of the seismic cycle. Remote Sens. Environ. 2012, 120, 164–174. [CrossRef]
4. Bianchessi, N.; Righini, G. Planning and scheduling algorithms for the COSMO-SkyMed constellation. Aerosp. Sci. Technol. 2008, 12, 535–544. [CrossRef]
5. Wolfe, W.J.; Sorensen, S.E. Three scheduling algorithms applied to the Earth observing systems domain. Manag. Sci. 2000, 46, 148–166. [CrossRef]
6. Chu, X.; Chen, Y.; Xing, L. A branch and bound algorithm for agile Earth observation satellite scheduling. Discret. Dyn. Nat. Soc. 2017, 1–15. [CrossRef]
7. Wang, P.; Reinelt, G.; Gao, P.; Tan, Y. A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation. Comput. Ind. Eng. 2011, 61, 322–335. [CrossRef]
8. Chen, X.; Reinelt, G.; Dai, G.; Spitz, A. A mixed integer linear programming model for multi-satellite scheduling. Eur. J. Oper. Res. 2019, 275, 694–707. [CrossRef]
9. Barkaoui, M.; Berger, J. A new hybrid genetic algorithm for the collection scheduling problem for a satellite constellation. J. Oper. Res. Soc. 2019, 1–21. [CrossRef]
10. Zheng, Z.G.; Guo, J.; Gill, E. Swarm satellite mission scheduling & planning using Hybrid Dynamic Mutation Genetic Algorithm. Acta Astronaut. 2017, 137, 243–253.
11. Xhafa, F.; Sun, J.; Barolli, A.; Biberaj, A.; Barolli, L. Genetic algorithms for satellite scheduling problems. Mob. Inf. Syst. 2012, 8, 351–377. [CrossRef]
12. Soma, P.; Sonagiri, D.; Santhalakshmi, S.; Bagchi, T.; Kumar. Multi-Satellite Scheduling Using Genetic Algorithms. Available online: https://arc.aiaa.org/doi/pdf/10.2514/6.2004-743-515 (accessed on 13 August 2020).
13. He, L.; Liu, X.; Laporte, G.; Chen, Y.; Chen, Y. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Comput. Oper. Res.* 2018, 100, 12–25. [CrossRef]
14. Wang, P.; Gao, P.; Tan, Y. A model, a heuristic and a decision support system to solve the earth observing satellites fleet scheduling problem. In Proceedings of the Annual Conference on Computers, Troyes, France, 6–9 July 2009. pp. 256–261.
15. Sundar, K.; Qin, J.; Rathinam, S.; Ntaimo, L.; Darbha, S.; Valicka, C. Algorithms for a satellite constellation scheduling problem. In Proceedings of the Conference on Automation Science and Engineering, Fort Worth, TX, USA, 21–25 August 2016; pp. 373–378.
16. Wu, G.; Ma, M.; Zhu, J.; Qiu, D. Multi-satellite observation integrated scheduling method oriented to emergency tasks and common tasks. *J. Syst. Eng. Electron.* 2012, 23, 723–733. [CrossRef]
17. Xhafa, F.; Ip, A.W. Optimisation problems and resolution methods in satellite scheduling and space-craft operation: A survey. *Enterp. Inf. Syst.* 2019, 1–22. [CrossRef]
18. Woodworth, D.J.; Ziegler, F.A.; Grabenstein, J.B. Satellite Data Transmission and Receiving Station. U.S. Patent 4,876,737, 24 October 1989.
19. Xiaoyue, W.U. Application of genetic algorithm in satellite data transmission scheduling problem. *Syst. Eng. Theory Pract.* 2008, 28, 124–131.
20. Zhang, Z.; Hu, F.; Zhang, N. Ant colony algorithm for satellite control resource scheduling problem. *Appl. Intell.* 2018, 48, 3295–3305. [CrossRef]
21. Yunfeng, L.; Xiaoyue, W.U. Model of satellite data transmission scheduling problem based on multi-satellite combined reconnaissance. *J. Beijing Univ. Aeronaut. Astronaut.* 2008, 34, 948–951.
22. Zhang, J.; Xing, L.; Peng, G.; Yao, F.; Chen, C. A large-scale multiobjective satellite data transmission scheduling algorithm based on SVM+NSGA-II. *Swarm Evol. Comput.* 2019, 50, 100560. [CrossRef]
23. Waiming, Z.; Xiaoxuan, H.; Wei, X.; Peng, J. A two-phase genetic annealing method for integrated Earth observation satellite scheduling problems. *Soft Comput.* 2019, 23, 181–196. [CrossRef]
24. Peng, G.; Xing, L. An iterated local search algorithm for agile Earth observation satellite scheduling problem. In Proceedings of the 2018 SpaceOps Conference, Marseille, France, 28 May–1 June 2018. [CrossRef]
25. Tulpan, D.C.; Hoos, H. Choosing the Right Neighbourhood: A Way to Improve a Stochastic Local Search Algorithm; University of British Columbia: Vancouver, BC, Canada, 2002.
26. Vaughan, D.E.; Jacobson, S.H.; Hall, S.N.; McLay, L.A. Simultaneous generalized hill-Climbing algorithms for addressing sets of discrete optimization problems. *Informs J. Comput.* 2005, 17, 438–450. [CrossRef]
27. Sullivan, K.A.; Jacobson, S.H. Ordinal hill climbing algorithms for discrete manufacturing process design optimization problems. *Discret. Event Dyn. Syst.* 2000, 10, 307–324. [CrossRef]
28. Renders, J.M.; Bersini, H. Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways. In Proceedings of the First IEEE Conference on Evolutionary Computation, Orlando, FL, USA, 27–29 June 1994.
29. Liu, X.; Laporte, G.; Chen, Y.; He, R. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Comput. Oper. Res.* 2017, 86, 41–53. [CrossRef]
30. Birattari, M.; Zhi, Y.; Balaprakash, P.; Stitzle, T. F-Race and Iterated F-Race: An Overview; Springer: Berlin/Heidelberg, Germany, 2010; pp. 311–336.