Research Article

A Hybrid Genetic Algorithm for Satellite Image Downlink Scheduling Problem

Bingyu Song , Feng Yao , Yuning Chen , Yingguo Chen , and Yingwu Chen

College of Information System and Management, National University of Defense Technology, Changsha, Hunan 410073, China

Correspondence should be addressed to Feng Yao; yaofeng@nudt.edu.cn

Received 5 January 2018; Accepted 16 April 2018; Published 17 May 2018

Academic Editor: Lu Zhen

Copyright © 2018 Bingyu Song et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The satellite image downlink scheduling problem (SIDSP) is included in satellite mission planning as an important part. A customer demand is finished only if the corresponding images are eventually downloaded. Due to the growing customer demands and the limited ground resources, SIDSP is an oversubscribed scheduling problem. In this paper, we investigate SIDSP with the case study of China's commercial remote sensing satellite constellation (SuperView-1) and exploit the serial scheduling scheme for solving it. The idea is first determining a permutation of the downlink requests and then producing a schedule from the given ordered requests. A schedule generation algorithm (SGA) is proposed to assign the downlink time window for each scheduled request according to a given request permutation. A hybrid genetic algorithm (HGA) combined with neighborhood search is proposed to optimize the downlink request permutation with the purpose of maximizing the utility function. Experimental results on six groups of instances with different density demonstrate the effectiveness of the proposed approach.

1. Introduction

Most of the literature on satellite mission planning is divided into two categories: optical satellites [1–4] and Synthetic Aperture Radar (SAR) satellites [5–12]. The satellite image acquisition and image downlink are two main scheduling problems in satellite mission planning. The image acquisition scheduling is a process of selecting observation tasks (shot strips) generated from customer demands. All selected observation tasks consist of a set of requests as the input of SIDSP. The image downlink scheduling is a process of selecting a subset of downlink requests among the candidate set. Only if the image is successfully transmitted can the utility of the corresponding demand be counted. Considering all kinds of constraints, the subset of requests that can eventually be transmitted, their corresponding ground stations and downlink time windows are the output of SIDSP with the purpose of maximizing the utility function. SIDSP is a complex constrained optimization problem.

The growing customer demands and the limited ground resources make SIDSP become an oversubscribed scheduling problem. Not all the requests can be scheduled and there are cases that satellite has sufficient conditions for an imaging but

cannot arrange a downlink for it, which causes the imaging to be removed from the image acquisition plan and the corresponding customer demand has to be shelved. Image downlink may become a bottleneck in the efficiency of the whole satellite system.

Satellite image downlink system is mainly composed of on-board transmission system and ground receiving system. The on-board transmission system includes transmitter and antennas. The ground receiving system includes receiver and antennas. Satellite image downlink mission is to transmit the original or compressed data to the ground station by satellite-ground data link under the certain code rate and bit error rate. Satellites can work in two modes: (1) the antennas can work separately and they can independently downlink different images to one or different ground stations simultaneously; (2) only one antenna can work to downlink at a time. Downlink also can be classified into two ways: (1) the observed images are transmitted to the ground station in real time when the observation target and the ground station are simultaneously visible to the satellite; (2) first store observed images in the on-board memory and then transmit data to the ground station when the satellite is passing.

SIDSP and its variations have been studied by many authors, mainly divided into single-satellite problems [10–12] and multisatellite problems [1, 3, 5–9]. Single-satellite problem deals with the resource contention for satellite internal requests, while multisatellite problem should also take the resource contention for requests from different satellites into consideration. If every satellite can transmit data independently without affecting each other, the multisatellite problem can be transformed into a single-satellite problem. The constraints of SIDSP are not fixed, varying with the specific case. Different from the previous literature, we consider the constraints of satellite memory capacity in our case, which is detailed in Section 2.

The solving methods are mainly divided into three categories: (1) exact solution algorithms (e.g., mixed integer programming [13], dynamic programming [14], and branch and bound algorithm [15]); (2) heuristic algorithms (e.g., a priority-based heuristic [3], an ejection-chain heuristic [11], and a Lagrangian heuristic [16]); (3) metaheuristic algorithms (e.g., simulated annealing [12], tabu search [12, 17], and genetic algorithm [18]). The performance of these solving methods is measured through the instances with varied size and complexity.

In this paper, we investigate SIDSP with the case study of China's commercial remote sensing satellite constellation, SuperView-1. SuperView-1 consists of four optical satellites with the resolution of 0.5 meters. Two of them were launched in December 28, 2016, and the other two were launched in the later stage of the project. The four satellites will network with a 90-degree angle between each other, realizing the global revisit within one day. As a commercial satellite constellation, SuperView-1 will provide global customers with remote sensing data services for national land resources mapping, environmental monitoring, financial insurance, and value-added service in the Internet industry. There are a large number of images that need to be transmitted every day. In addition, China's ground stations are built on the mainland and not scattered around the world. In the project of SuperView-1, three domestic ground stations and one Arctic ground station are used to downlink images. It takes a fee to use the Arctic ground station for downlink in each orbit. These specific situations increase the difficulty of scheduling.

The process currently in use for SuperView-1 downlink scheduling includes two phases: (1) construction of the schedules with a priority-based heuristic and (2) human intervention, which do not perform very well. To improve the efficiency of the downlink scheduling is significant for the whole satellite system. Adapting to the characteristic of SIDSP, we exploit the serial scheduling scheme involving two key parts: (1) determining a permutation of the downlink requests and (2) producing a schedule from the given ordered requests. A schedule generation algorithm (SGA) is proposed to assign the downlink time window for each scheduled request. In SGA, we use the greedy heuristic rule, scheduling each request to the earliest available time and designing two heuristic rules to improve the utility considering the cost of the Arctic ground station. We propose a hybrid genetic algorithm (HGA) combined with neighborhood search to

optimize the downlink request permutation with the purpose of maximizing the utility function. To validate the effectiveness of the proposed approach, we carry out experiments on several groups of instances with different density and compare HGA with the commonly used construction heuristic algorithms. The experimental results show that HGA and the heuristic rules we design in SGA significantly improve downlink throughput and schedule quality.

The remainder of the paper is organized as follows. Section 2 gives the description of SIDSP in SuperView-1. The solution approach for SIDSP and several key algorithms are introduced in Section 3. Section 4 presents the experimental results of the proposed approach on several groups of instances with different density and the comparison between HGA and the construction heuristic algorithms. Concluding remarks are drawn in Section 5.

2. Problem Description

In this section, SIDSP in SuperView-1 is detailed as the studied case. Each SIDSP has its own restrictions and properties that can be exploited. We hope to discover the commonalities of SIDSP and show the special properties through investigating the case to make preparations for the design of solution approach.

Due to the special structure of the constellation where four satellites operate with a 90-degree angle between each other, every satellite can transmit data independently. For this reason, SIDSP in SuperView-1 can be transformed into a single-satellite problem. Let $R = \{r_1, r_2, \dots, r_n\}$ be a set of n downlink requests. Each request r_i ($r_i \in R$) is associated with the start time of image acquisition ast_i , the end time of image acquisition aet_i , the start time of downlink dst_i , the end time of downlink det_i , the profit p_i , the size s_i , and the deadline dl_i . A downlink activity can be carried out only when the satellite is passing over a ground station. This time interval is called visibility mask of the station. Let $V = \{v_1, v_2, \dots, v_m\}$ be a set of m visibility masks. Each visibility mask v_k ($v_k \in V$) is associated with the start time st_k , the end time et_k , and the cost c_k (only each visibility mask of the Arctic ground station needs a cost; other visibility masks' cost is 0). Moreover, the memory capacity of each satellite is C . Let S_i be the current satellite storage when the image acquisition of request r_i is finished. Let x be $n \times m$ binary matrix such that $x_{ik} = 1$ if the downlink of request r_i is assigned to visibility mask v_k and $x_{ik} = 0$ otherwise. Let R_s be a set of scheduled downlink requests.

We make the following assumptions:

- (a) The image acquisition schedule is given and not to be changed; that is, ast_i and aet_i are already generated.
- (b) The gaps between two consecutive downlinks are ignored.
- (c) The compression ratio of each image is CR (in SuperView-1, $CR = 1/2$); that is, the image acquisition duration is CR times as long as the downlink duration, which can be stated as

$$aet_i - ast_i = CR \cdot (det_i - dst_i), \quad \forall r_i \in R_s. \quad (1)$$

- (d) The satellites work in the mode that only one antenna can work to downlink at a time; that is, any two downlink time windows do not overlap, which can be stated as

$$\text{dst}_i \geq \text{det}_j \vee \text{dst}_j \geq \text{det}_i, \quad \forall r_i, r_j \in R_s \wedge i \neq j. \quad (2)$$

- (e) The downlink way that the observed images are transmitted to the ground station in real time is not considered.
- (f) The storage of each request can be released immediately, once the corresponding downlink finishes.

Our scheduling results must satisfy the following constraints:

- (a) If the downlink of request r_i is assigned to visibility mask v_k , dst_i and det_i must be within visibility mask v_k , which can be stated as

$$\text{st}_k \leq \text{dst}_i < \text{det}_i \leq \text{et}_k, \quad x_{ik} = 1. \quad (3)$$

- (b) The start time of downlink dst_i must be after the end time of image acquisition aet_i and the end time of downlink det_i cannot be after the deadline dl_i , which can be stated as

$$\text{aet}_i < \text{dst}_i < \text{det}_i \leq \text{dl}_i, \quad \forall r_i \in R_s. \quad (4)$$

- (c) Once a downlink starts, it cannot be preempted; that is, an image cannot be split into several fragments, which can be stated as

$$\sum_{k=1}^m x_{ik} \leq 1, \quad \forall r_i \in R. \quad (5)$$

- (d) The satellites cannot observe and downlink at the same time and there must be a gap of at least G units between consecutive image acquisition and downlink, which can be stated as

$$\text{dst}_i \geq \text{aet}_j + G \vee \text{ast}_j \geq \text{det}_i + G, \quad \forall r_i, r_j \in R_s. \quad (6)$$

- (e) The satellite storage cannot exceed the memory capacity C at any time, which can be stated as

$$S_i \leq C, \quad \forall r_i \in R_s. \quad (7)$$

We consider the storage constraint (Constraint (e)) in SIDSP. The limited on-board memory makes us consider how to improve the utilization efficiency by downlink schedule. Image acquisition is a process of loading while image downlink is a process of unloading. There are cases that the satellite cannot acquire image because of the lack of storage. Image downlink can release storage to make it possible for image acquisition to improve the efficiency of the whole satellite system.

Note that SIDSP deals with the problem of finding an image downlink schedule so that a utility function is maximized. The utility function considers the profit of each scheduled request and the cost of each used visibility mask. In order to record which visibility masks are used, we set a

variable u_k such that $u_k = 1$ if the visibility mask v_k is used and $u_k = 0$ otherwise, which can be stated as

$$u_k = \begin{cases} 0, & \sum_{i=1}^n x_{ik} = 0 \\ 1, & \sum_{i=1}^n x_{ik} > 0. \end{cases} \quad (8)$$

Our objective function is maximizing the utility function of solution S , which can be stated as

$$\max U = f(S) = \sum_{i=1}^n \sum_{k=1}^m x_{ik} p_i - \sum_{k=1}^m u_k c_k. \quad (9)$$

To solve SIDSP, we need to determine two points. One is to determine the corresponding visibility mask assigned for each request (i.e., the binary matrix x); the other one is to determine the downlink time window of each request (i.e., the start time of downlink dst_i , the end time of downlink det_i). The requests which cannot be scheduled eventually are removed from the image acquisition plan.

3. Solution Approach

In this section, the solution approach based on serial scheduling is described to deal with SIDSP. The construction heuristic algorithms for comparison are detailed in Section 3.1; the schedule generation algorithm (SGA) is detailed in Section 3.2; the hybrid genetic algorithm (HGA) is detailed in Section 3.3.

Serial scheduling scheme is an effective approach to deal with constrained optimization problems such as Job Shop scheduling [19, 20]. The solver searches in the space of request permutations while a polynomial time schedule generator converts the request permutations into schedules. The schedule generator is a greedy algorithm scheduling the requests in the given order, choosing the earliest available position for each of them considering the constraints. An important property of the schedule generator is that it always generates active schedules; that is, none of the unscheduled requests can be added to it without delaying some scheduled requests [21]. If we do not consider the cost of visibility mask, there exists a request permutation generating an optimal schedule [22]. Our utility function considers the cost of visibility mask so that sometimes delaying or deleting a downlink can improve the solution, which makes us design two other heuristic rules. There also exists a request permutation generating an optimal schedule in our case, which is explained in Section 3.2. With the serial scheduling scheme in mind, SIDSP can be viewed as a problem of optimizing a permutation of the downlink requests. Our objective function also can be maximizing the utility function of permutation P , which is stated as

$$\max U = \phi(P). \quad (10)$$

Note that to evaluate the solution profit of a permutation we need to generate the schedule from that permutation, that is, to apply the schedule generation algorithm. In the following section, let r_i^P be the i th request in P and $x_{ik}^P = 1$

```

Input  $P = \{r_1^P, r_2^P, \dots, r_n^P\}$ : a given request permutation;  $V = \{v_1, v_2, \dots, v_m\}$ : a set of  $m$  visibility masks;
Output  $S$ : a solution;
for  $i \leftarrow 1$  to  $n$  do
   $S \leftarrow \text{Greedy heuristic}(r_i^P, V)$ ; /* Use the greedy heuristic rule to assign request  $r_i^P$  */
  if  $x_{ik}^P = 1 \wedge c_k > 0$  /* Request  $r_i^P$  is assigned to a chargeable visibility mask  $v_k$  */
     $S \leftarrow \text{Delay}(r_i^P, V)$ ; /* Use the delay rule to assign request  $r_i^P$  */
  end if
   $V \leftarrow \text{Clipping}(V)$ ; /* clip the visibility masks after a request is assigned */
end for
 $S \leftarrow \text{Deletion}(S, r_i^P, V)$ ; /* Use the deletion rule to generate a solution */
return  $S$ ;

```

ALGORITHM 1: The schedule generation algorithm.

if the downlink of request r_i^P is assigned to visibility mask v_k and $x_{ik}^P = 0$ otherwise.

3.1. Construction Heuristic. The construction heuristic algorithms are efficient for quick request permutation generation. By using the construction heuristic algorithms reasonably, we can quickly get a schedule with a high profit. Adapted to the characteristic of SIDSP in SuperView-1, the problem can be seen as a variant of multiple knapsack problem: there are n items (analogous to requests) and m knapsacks (analogous to visibility masks). Each item has its own profit (analogous to request profit) and weight (analogous to downlink duration). Each knapsack has its own capacity (analogous to visibility mask duration). We should find a schedule of how to assign each item to maximize the total profit without violating knapsack capacity constraints [23]. The following sorting criterions commonly used in solving multiple knapsack problem are considered,

Criterion 1. Higher profit p_i occurs first.

Criterion 2. Shorter downlink duration $\text{det}_i - \text{dst}_i$ exists first.

Criterion 3. Higher $p_i/(\text{det}_i - \text{dst}_i)$ exists first.

Adapted to our purpose of maximizing the utility, Criterion 1 gives priority to those requests with high profit and may make the total profit of the scheduled requests higher. Adapted to Constraint (a), when it is a tight constraint (i.e., the visibility masks are oversubscribed), Criterion 2 can increase the number of scheduled requests and may have a good performance; Criterion 3 focuses on those requests with high profit per unit time, which can be seen as the balance between Criteria 1 and 2.

3.2. Schedule Generation Algorithm (SGA). SGA is used to produce a schedule from a given request permutation. As the key part in serial scheduling scheme, SGA is applied each time we evaluate the solution profit of a permutation. The general procedure of SGA is described in Algorithm 1, mainly including three heuristic rules and time window clipping.

(1) The Greedy Heuristic Rule. We choose the earliest available position as the downlink time window for each request in the

FIGURE 1: The case that using the delay rule can improve the utility.

given order. When the visibility masks are oversubscribed, making full use of them is an important principle. This rule makes sure that the scheduled requests cannot be advanced and the whole schedule gets more compact. What has been proved is that there exists a request permutation generating an optimal schedule [22].

(2) The Delay Rule. When we consider the cost of the Arctic ground station in the utility function, the inherent combinatorial structure of the problem is changed. Only using the greedy heuristic rule may lead to losing the optimal schedule (e.g., in Figure 1, we assign request r_i to the chargeable visibility mask v_1 only based on the greedy heuristic rule, but if request r_i is delayed and assigned to the free visibility mask v_2 , the utility will be improved), for which we add the delay rule into SGA. To describe the delay rule, if the earliest available position of a request is in a chargeable visibility mask, the request will be delayed and assigned to the earliest available position in other free visibility masks. If there is no available position in other free visibility masks, the request will not be delayed.

Although we adopt the delay rule according to the characteristics of this problem, the opportunity to get an optimal schedule is also kept. Using the delay rule leads to missing the optimal schedule with some special permutations (e.g., in Figure 2, if request r_1 is sorted in front of request r_2 ,

FIGURE 2: The case that using the delay rule leads to missing the optimal schedule.

FIGURE 3: The case that using the delay rule leads to generating the optimal schedule.

first we assign request r_1 to the free visibility mask v_2 based on the delay rule and then request r_2 cannot be scheduled because of no available position). But there also exists a request permutation generating an optimal schedule (e.g., in Figure 3, if request r_2 is sorted in front of request r_1 , first we assign request r_2 to the free visibility mask v_2 and then request r_1 that can be assigned to the chargeable visibility mask v_1).

(3) *The Deletion Rule.* The profits of requests which are assigned to the same visibility mask are added as the total

FIGURE 4: Visibility mask clipping.

profit of that visibility mask. After all the requests are assigned, we check each chargeable visibility mask whether its total profit is higher than its cost. If so, keep the current schedule as the final schedule; if not, we delete all the requests assigned to the unprofitable visibility mask. The deletion rule makes sure that each chargeable visibility mask we used is profitable and improves the utility as the last operation before a schedule generation.

(4) *Visibility Mask Clipping.* After a request is assigned, its image acquisition time window and its downlink time window are determined. Due to Assumption (d) and Constraint (d) (in Section 2), the occupied periods in visibility masks cannot be used to observe and downlink anymore and the gap of G units between consecutive image acquisition and downlink should be reserved. So we need to clip affected visibility masks to make the later assignments satisfy the constraints, which is shown in Figure 4.

3.3. *Hybrid Genetic Algorithm (HGA).* HGA is proposed to optimize the downlink request permutation. The framework of HGA is consistent with GA. In order to strengthen the local optimization ability of GA, we use neighborhood search in mutation operation. The general procedure of HGA is described in Algorithm 2, mainly including the following parts.

(1) *Coding.* The genes of the chromosomes describe the requests, and the order in which they appear in the chromosome describes the scheduling order of requests. In each chromosome, the gene number should be consistent with the number of requests n and the genes cannot be repeatable.

```

Input  $R = \{r_1, r_2, \dots, r_n\}$ : a set of  $n$  downlink requests;  $V = \{v_1, v_2, \dots, v_m\}$ : a set of  $m$  visibility masks;
Output  $S_{best}$ : the best solution found so far;
 $Pop = \{P_1, P_2, \dots, P_{n/5}\} \leftarrow \text{Random\_Init}(R)$ ; /* Initialize a population of  $n/5$  request permutations by randomly sorting */
while stopping condition is not reached do
   $P_{s1}, P_{s2} \leftarrow \text{Selection}(Pop)$ ; /* Select two individuals for reproduction */
   $P_{s1}, P_{s2} \leftarrow \text{Crossover}(P_{s1}, P_{s2})$ ; /* Generate two new individuals by crossover */
   $\phi(P_{s1}) \leftarrow \text{SGA}(P_{s1}, V)$ ; /* Use SGA to calculate the utility of  $P_{s1}$  */
   $\phi(P_{s2}) \leftarrow \text{SGA}(P_{s2}, V)$ ; /* Use SGA to calculate the utility of  $P_{s2}$  */
   $\text{improve\_}P_{s1} \leftarrow 1$ ; /* Initialize improving flag */
   $\text{improve\_}P_{s2} \leftarrow 1$ ;
  while  $\text{improve\_}P_{s1} = 1$  do /* Neighborhood search continues until the utility is not improved */
     $\text{improve\_}P_{s1} \leftarrow 0$ ;
    for  $i \leftarrow 1$  to  $n$  do
      for  $j \leftarrow i + 1$  to  $n$  do
         $P'_{s1} \leftarrow \text{Exchange}(r_i^{P_{s1}}, r_j^{P_{s1}})$ ; /* Use the exchange operator */
         $\phi(P'_{s1}) \leftarrow \text{SGA}(P'_{s1}, V)$ ; /* Use SGA to calculate the utility of  $P'_{s1}$  */
        if  $\phi(P'_{s1}) > \phi(P_{s1})$  then
           $P_{s1} \leftarrow P'_{s1}$ ; /* Preserve the exchange which improves the utility */
           $\phi(P_{s1}) \leftarrow \phi(P'_{s1})$ ; /* Update the utility of  $P_{s1}$  */
           $\text{improve\_}P_{s1} \leftarrow 1$ ;
        end if
      end for
    end for
  end while
  while  $\text{improve\_}P_{s2} = 1$  do /* Neighborhood search continues until the utility is not improved */
     $\text{improve\_}P_{s2} \leftarrow 0$ ;
    for  $i \leftarrow 1$  to  $n$  do
      for  $j \leftarrow i + 1$  to  $n$  do
         $P'_{s2} \leftarrow \text{Exchange}(r_i^{P_{s2}}, r_j^{P_{s2}})$ ; /* Use the exchange operator */
         $\phi(P'_{s2}) \leftarrow \text{SGA}(P'_{s2}, V)$ ; /* Use SGA to calculate the utility of  $P'_{s2}$  */
        if  $\phi(P'_{s2}) > \phi(P_{s2})$  then
           $P_{s2} \leftarrow P'_{s2}$ ; /* Preserve the exchange which improves the utility */
           $\phi(P_{s2}) \leftarrow \phi(P'_{s2})$ ; /* Update the utility of  $P_{s2}$  */
           $\text{improve\_}P_{s2} \leftarrow 1$ ;
        end if
      end for
    end for
  end while
   $Pop \leftarrow \text{Update}(Pop, P_{s1}, P_{s2})$ ; /* Update the population by replacing two of the worst individuals */
end while
 $P_{best} \leftarrow \text{Best}(Pop)$ ; /* Select the request permutation with the highest utility in the population */
 $S_{best} \leftarrow \text{SGA}(P_{best}, V)$ ; /* Use SGA to generate the final solution */
return  $S_{best}$ ;

```

ALGORITHM 2: The hybrid genetic algorithm.

In fact, each chromosome represents a downlink request permutation.

(2) *Initial Population.* The initial chromosomes are obtained by random ordering. The individual number is related to the number of requests n , set to $n/5$.

(3) *Fitness Evaluation.* The utility is computed for each chromosome in the current generation by SGA.

(4) *Selection.* At each iteration, two chromosomes are chosen for reproduction based on roulette wheel selection. For each

chromosome P_i , its utility is $\phi(P_i)$ and the probability that it is selected can be stated as

$$P(P_i) = \frac{\phi(P_i)}{\sum_{i=1}^{n/5} \phi(P_i)}. \quad (11)$$

(5) *Offspring Generation.* The new generation is obtained by changing the order of genes, which includes two operations: crossover and mutation. These operations preserve feasibility of new individuals. At each iteration, two new individuals are generated and replace two of the worst individuals in the population. If new individual is worse than all the original

individuals, it will be abandoned. At the end of each iteration, the updated population is obtained.

In crossover operation, we use the partially mapped crossover (PMX) which includes a two-point crossover and a repair process to ensure the legality of coding.

In mutation operation, we use neighborhood search to strengthen the local optimization ability. An exchange operator which changes the chromosome by exchanging two genes is defined. The operator exchanges two genes in order and the exchange which improves the utility is preserved in each search. The whole search process continues until the utility is not improved with exchanging any two genes. For a detailed description of our neighborhood search see Algorithm 2.

(6) *Stop Criterion.* Fixed number of iterations is reached. If the stop criterion is satisfied, the algorithm ends and the best chromosome, together with the corresponding schedule, is given as output. Otherwise, the algorithm iterates again steps (4)-(5).

4. Computational Experiments

In this section, we present experimental studies on the schedule generation algorithm (SGA) and the hybrid genetic algorithm (HGA).

Test Instances. The instances are divided into six groups according to the density of requests. The requests in each instance are evenly distributed around the world and the density is characterized by the number of requests $n \in \{50, 100, 150, 200, 250, 300\}$. Each group has five instances denoted as “ $n - 1, n - 2, \dots, n - 5$.” The planning horizon is set to 24 h. In our case, three domestic ground stations and one Arctic ground station are used to downlink images. Based on the orbit parameters of number 1 optical satellite in SuperView-1, each domestic ground station is visible 3 to 5 times, and the Arctic ground station is visible 11 times to the satellite during the planning horizon. Each visibility mask lasts for 2 to 10 minutes. URL for instances is <http://forge.info.univ-angers.fr/~yuning/instances/SIDSPinstances.rar>.

Experimental Environment. The proposed algorithm was coded in C++ and compiled using Visual Studio 2012 on an Intel Core i7-4790 processor (3.6 GHz and 8 GB RAM).

Parameter Set. Following an empirical parameter tuning procedure, the parameters in HGA are related to the density of requests. The individual number is set to $n/5$. The number of iterations is, respectively, set to 20, 50, 50, 100, 100, and 100, when n is 50, 100, 150, 200, 250, and 300, correspondingly.

The performance of the algorithms is evaluated by three indexes: (1) the utility U , (2) the number of scheduled requests $|R_s|$, and (3) the visibility mask utilization VU . U is our objective; more $|R_s|$ or higher VU may lead to a better solution. VU can be stated as

$$VU = \frac{\sum_{r_i \in R_s} (\text{det}_i - \text{dst}_i)}{\sum_{k=1}^m (\text{et}_k - \text{st}_k)}. \quad (12)$$

FIGURE 5: The average values of U in HGA for different groups of instances.

FIGURE 6: The average values of $|R_s|$ in HGA for different groups of instances.

The comparison of HGA and the construction heuristic algorithms is shown in Table 1. Table 2 gives the results of HGA under two circumstances, removing the delay rule from SGA and removing the deletion rule from SGA. The average values of U , $|R_s|$, and VU in HGA for different groups of instances are, respectively, shown in Figures 5–7.

From the experimental results, the following observations can be made:

- (1) From Table 1, we can see that Criterion 3 has a better performance for SIDSP than the other two sorting criterions (18 out of 30) and U obtained by HGA for each instance is much higher than the highest value of U obtained by three construction heuristic algorithms marked in bold. $|R_s|$ and VU of HGA are also better than those of three construction heuristic algorithms. HGA greatly improves the quality of

TABLE 1: The comparison of HGA and the construction heuristic algorithms (the highest value of U for each instance obtained by three construction heuristic algorithms is marked in bold).

Instance	HGA			Criterion 1			Construction heuristic algorithms			Criterion 3		
	U	$ R_s $	VU (%)	U	$ R_s $	VU (%)	U	$ R_s $	VU (%)	U	$ R_s $	VU (%)
50-1	14706	45	13.84	13723	45	13.58	13997	46	13.90	14116	46	13.94
50-2	15726	47	16.98	14663	46	17.08	14942	47	16.90	14942	47	16.90
50-3	14965	45	15.73	13360	43	14.77	13822	44	15.07	13822	44	15.07
50-4	14611	46	17.75	13847	44	16.64	13893	44	16.68	13217	45	17.24
50-5	15336	45	16.48	13920	44	16.38	13911	42	15.13	13911	42	15.13
100-1	33401	92	31.83	28724	81	28.06	28609	83	27.64	28915	83	27.82
100-2	33121	92	29.39	29904	88	28.53	29450	83	25.60	29630	83	25.75
100-3	31388	89	32.64	27368	78	27.64	28804	86	30.56	29330	86	30.62
100-4	33437	94	32.80	29897	84	29.05	29069	88	29.65	29079	88	29.81
100-5	32552	93	31.18	28911	82	26.91	27735	81	25.42	28015	81	25.54
150-1	48064	132	47.12	36246	96	35.29	37852	107	33.40	38523	109	35.25
150-2	47003	131	47.20	38710	110	40.03	37970	110	35.39	38078	111	36.14
150-3	46853	133	48.01	36697	103	35.62	39815	115	37.91	40278	115	38.56
150-4	48539	135	47.18	37473	102	35.52	38662	109	32.82	39252	109	33.12
150-5	47712	133	47.84	38829	109	38.54	36838	104	33.26	37182	103	33.42
200-1	60603	168	54.79	45696	124	39.34	47726	134	36.44	48061	132	36.08
200-2	61509	168	57.47	51462	139	46.11	47966	134	39.71	48269	131	38.88
200-3	61831	166	57.33	47793	127	42.79	48292	135	41.16	52043	140	44.50
200-4	63744	173	56.43	45143	120	38.96	50414	137	38.17	50360	133	36.60
200-5	62700	169	51.19	47155	120	37.57	53696	148	40.95	51551	141	39.04
250-1	70442	191	68.07	47544	133	45.61	50238	141	40.89	51442	139	40.49
250-2	71898	197	64.32	52614	137	44.78	53338	149	41.78	53794	148	41.94
250-3	72696	201	64.85	50987	134	43.37	53740	152	40.01	54214	149	39.34
250-4	72115	192	64.28	56824	147	49.66	54001	149	41.30	56191	151	43.05
250-5	73138	198	65.83	50920	131	44.32	54924	153	41.62	56250	152	42.04
300-1	79244	218	69.16	55277	147	48.67	61203	172	45.21	59412	163	44.36
300-2	81224	221	70.59	57850	152	49.21	61658	173	45.61	61847	170	45.69
300-3	79053	219	72.08	58712	160	53.02	60572	171	44.94	59545	167	44.80
300-4	80740	219	69.14	54049	141	47.76	60325	169	43.67	62398	175	46.74
300-5	80674	222	71.72	56434	151	49.29	59164	167	43.15	61216	166	43.61

TABLE 2: The results of HGA under two circumstances, removing the delay rule from SGA (no delay) and removing the deletion rule from SGA (no deletion) (the utility difference between the normal HGA and HGA under the corresponding circumstance is denoted as δU ; the utility obtained by normal HGA is denoted as U_{normal} ; the instances that U gets higher when the deletion rule is removed are marked in bold).

Instance	U	$ R_s $	HGA (no delay)			U	$ R_s $	HGA (no deletion)		
			VU (%)	δU	$\delta U/U_{\text{normal}}$ (%)			VU (%)	δU	$\delta U/U_{\text{normal}}$ (%)
50-1	10939	43	12.75	3767	25.62	14469	47	14.34	237	1.61
50-2	12196	48	17.71	3530	22.45	15678	49	17.85	48	0.31
50-3	11138	45	15.69	3827	25.57	14128	48	16.20	837	5.59
50-4	10528	43	16.52	4083	27.94	14039	47	18.27	572	3.91
50-5	11949	44	16.80	3387	22.09	15262	47	17.59	74	0.48
100-1	29225	89	30.86	4176	12.50	33267	94	32.84	134	0.40
100-2	29805	91	29.49	3316	10.01	32433	93	29.71	688	2.08
100-3	28287	89	32.45	3101	9.88	30465	92	33.68	923	2.94
100-4	29668	92	32.19	3769	11.27	32310	96	33.50	1127	3.37
100-5	28700	91	30.32	3852	11.83	31916	94	31.55	636	1.95
150-1	45131	127	44.76	2933	6.10	48080	135	47.97	-16	-0.03
150-2	44829	131	47.14	2174	4.63	46411	135	48.83	592	1.26
150-3	43910	128	46.76	2943	6.28	46420	134	48.35	433	0.92
150-4	45578	130	45.15	2961	6.10	48132	136	47.52	407	0.84
150-5	44136	130	47.34	3576	7.49	47352	138	49.78	360	0.75
200-1	58228	170	54.71	2375	3.92	60410	170	55.18	193	0.32
200-2	58943	167	57.35	2566	4.17	60892	169	57.74	617	1.00
200-3	59487	169	59.51	2344	3.79	60887	169	58.76	944	1.53
200-4	61330	172	56.73	2414	3.79	63483	174	57.13	261	0.41
200-5	60220	171	52.96	2480	3.96	62845	172	53.38	-145	-0.23
250-1	67979	189	67.97	2463	3.50	70035	192	68.71	407	0.58
250-2	70223	198	65.45	1675	2.33	71467	201	66.56	431	0.60
250-3	70394	200	65.47	2302	3.17	72530	200	65.73	166	0.23
250-4	70937	195	66.24	1178	1.63	72157	199	68.15	-42	-0.06
250-5	71184	196	64.36	1954	2.67	72872	201	65.71	266	0.36
300-1	78439	221	71.13	805	1.02	79553	220	70.69	-309	-0.39
300-2	78622	218	69.94	2602	3.20	80956	221	70.29	268	0.33
300-3	78131	219	70.97	922	1.17	79172	219	72.06	-119	-0.15
300-4	79219	220	69.24	1521	1.88	79948	217	68.11	792	0.98
300-5	78127	217	69.46	2547	3.16	79496	219	68.80	1178	1.46

FIGURE 7: The average values of VU in HGA for different groups of instances.

solution compared with the construction heuristic algorithms.

- (2) In Table 1, from the result of HGA, we can see that there are still some unscheduled requests in each instance, although the visibility masks are not fully used. It illustrates the deadline constraint (Constraint (b)) and the storage constraint (Constraint (e)) may be tight so that some requests cannot be scheduled due to lack of storage or beyond the deadline, which also causes that some periods in visibility masks cannot be used (i.e., the visibility masks are oversubscribed).
- (3) By comparing the results of HGA under different circumstances in Table 2, it can be seen that using the delay rule and the deletion rule in SGA can lead to a better solution, and the effectiveness of the delay rule and the deletion rule is validated. When the delay rule is removed, U is lower than that in normal circumstance for all instances. When the deletion rule is removed, U is lower than that in normal circumstance for most instances except five instances marked in bold. Furthermore, $\delta U/U_{\text{normal}}$ in HGA (no delay) is higher than that in HGA (no deletion). It illustrates that removing the delay rule causes a larger reduction in U than removing the deletion rule and the delay rule has a greater effect on the improvement of solution.
- (4) In Table 2, when the delay rule is removed, $\delta U/U_{\text{normal}}$ is on decline with the density of requests increasing. It illustrates the effect of the delay rule is weakening for the instances with higher density. This is because we have to use the chargeable visibility masks when the free visibility masks are not enough to downlink more requests. The cost of the chargeable visibility masks is calculated into the final utility whether the delay rule is used or not. But the

effect of the deletion rule does not have an obvious change with the density of requests increasing.

- (5) From Figures 5–7, it can be seen that, U , $|R_s|$, and VU step up with the number of requests increasing, while the growth rate steps down. When the number of requests gets higher, the conflict between different requests gets more so that there are more requests that cannot be scheduled due to violating the constraints.

5. Conclusion

In this paper, we investigate the satellite image downlink scheduling problem (SIDSP) with the case study of China's commercial remote sensing satellite constellation (SuperView-1). SIDSP, as an important part in satellite mission planning, should be optimized to improve the efficiency of the whole satellite system.

We give a detailed description of SIDSP in SuperView-1 to make preparations for the design of solution approach. In this case, not only the commonalities of SIDSP but also some special properties should be taken into consideration, such as the cost of some visibility masks and the storage constraint, which changes the inherent combinatorial structure of the problem and increases the difficulty of scheduling.

Adapted to the characteristics of the problem, the serial scheduling scheme is exploited. The main idea of the scheme is that the solver searches in the space of request permutations while a polynomial time schedule generator converts the request permutations into schedules, which makes SIDSP viewed as a problem of optimizing a permutation of the downlink requests. A schedule generation algorithm (SGA) is proposed to assign the downlink time window for each scheduled request according to a given request permutation. A hybrid genetic algorithm (HGA) combined with neighborhood search is proposed to optimize the downlink request permutation with the purpose of maximizing the utility function.

To validate the effectiveness of the proposed approach, we carry out experiments on six groups of instances with different density. From the experimental results, we can see that HGA greatly improves the quality of solution compared with the construction heuristic algorithms. Using the delay rule and the deletion rule in SGA can lead to a better solution, and the delay rule has a greater effect on the improvement of solution than the deletion rule. However, the effect of the delay rule is weakening with the density of instance getting higher.

SIDSP in our studied case is transformed into a single-satellite problem. In addition, our downlink schedule is based on the given image acquisition schedule, which makes the result of downlink schedule affected by the efficiency of image acquisition schedule. In the future work, on the one hand, we would like to extend our approach to multisatellite problems and take the resource contention for requests from different satellites into consideration; on the other hand, we would like to study the integrated scheduling considering image acquisition schedule and image downlink schedule simultaneously.

Data Availability

The instances for SIDSP used in this paper can be downloaded at <http://forge.info.univ-angers.fr/~yuning/instances/SIDSPInstances.rar>.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 71690233, 61473301, 71201171, 71501179, and 71501180).

References

- [1] V. Malyshev and V. Bobronnikov, "Mission Planning for Remote Sensing Satellite Constellation," in *Design & Implementation of Satellite Constellations*, pp. 431–437, Springer, Dordrecht, Netherlands, 1998.
- [2] M. Vazquez and J. K. Hao, "A "logic-constrained" knapsack formulation and a tabu algorithm for the daily photograph scheduling of an Earth observation satellite," *Computational optimization and applications*, vol. 20, no. 2, pp. 137–157, 2001.
- [3] P. Wang, G. Reinelt, P. Gao, and Y. Tan, "A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation," *Computers & Industrial Engineering*, vol. 61, no. 2, pp. 322–335, 2011.
- [4] L. He, X. L. Liu, Y. W. Chen, and L. N. Xing, "Cloud modeling and processing method for agile observing satellite mission planning," *Systems Engineering and Electronics*, vol. 4, article 020, 2016.
- [5] L. Barbulescu, A. E. Howe, J. P. Watson, and L. D. Whitley, "Satellite range scheduling: A comparison of genetic, heuristic and local search," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pp. 611–620, Berlin, Germany, 2002.
- [6] L. Barbulescu, J. P. Watson, L. D. Whitley, and A. E. Howe, "Scheduling space-ground communications for the air force satellite control network," *Journal of Scheduling*, vol. 7, no. 1, pp. 7–34, 2004.
- [7] L. Barbulescu, L. D. Whitley, and A. E. Howe, "Leap before you look: An effective strategy in an oversubscribed scheduling problem," in *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004): Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-2004)*, pp. 143–148, July 2004.
- [8] L. Barbulescu, A. Howe, and D. Whitley, "AFSCN scheduling: How the problem and solution have evolved," *Mathematical and Computer Modelling*, vol. 43, no. 9–10, pp. 1023–1037, 2006.
- [9] N. Bianchessi and G. Righini, "Planning and scheduling algorithms for the COSMO-SkyMed constellation," *Aerospace Science and Technology*, vol. 12, no. 7, pp. 535–544, 2008.
- [10] G. Beaumet, G. Verfaillie, and M.-C. Charneau, "Feasibility of autonomous decision making on board an agile Earth-observing satellite," *International Journal of Computational Intelligence*, vol. 27, no. 1, pp. 123–139, 2011.
- [11] D. Karapetyan, K. T. Malladi, S. Mitrovicminic, and A. P. Punnen, "An ejection-chain heuristic for the satellite downlink scheduling problem: a case study with radarsat-2," *International*, pp. 497–516, 2012.
- [12] D. Karapetyan, S. Mitrovic Minic, K. T. Malladi, and A. P. Punnen, "Satellite downlink scheduling problem: A case study," *OMEGA - The International Journal of Management Science*, vol. 53, pp. 115–123, 2015.
- [13] T. D. Gooley, *Automating the satellite range scheduling process [MSc, thesis]*, Air Force Institute of Technology, 1993.
- [14] S. Damiani, G. Verfaillie, and M. C. Charneau, "An earth watching satellite constellation: How to manage a team of watching agents with limited communications," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 455–462, ACM, 2005.
- [15] E. Bensana, G. Verfaillie, J. C. Agnese, N. Bataille, and D. Blumstein, "Exact & inexact methods for daily management of earth observation satellite," in *Proceedings of the Space Mission Operations and Ground Data Systems-SpaceOps' 96*, vol. 394, article 507, November 1996.
- [16] F. Marinelli, S. Nocella, F. Rossi, and S. Smriglio, "A Lagrangian heuristic for satellite range scheduling with resource constraints," *Computers & Operations Research*, vol. 38, no. 11, pp. 1572–1583, 2011.
- [17] N. Bianchessi, J.-F. Cordeau, J. Desrosiers, G. Laporte, and V. Raymond, "A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites," *European Journal of Operational Research*, vol. 177, no. 2, pp. 750–762, 2007.
- [18] H. Chen, Z. Zhong, J. Wu, and N. Jing, "Multi-satellite data downlink resource scheduling algorithm for incremental observation tasks based on evolutionary computation," in *Proceedings of the 2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*, pp. 251–256, IEEE, 2015.
- [19] J. L. Kim, "Permutation-based elitist genetic algorithm using serial scheme for large-sized resource-constrained project scheduling," in *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, pp. 2112–2118, Washington, DC, USA, December 2007.
- [20] S. Rajeev, S. Kurian, and B. Paul, "A modified serial scheduling scheme for resource constrained project scheduling weighted earliness tardiness problem," *International Journal of Information and Decision Sciences*, vol. 7, no. 3, pp. 241–254, 2015.
- [21] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation," *European Journal of Operational Research*, vol. 90, no. 2, pp. 320–333, 1996.
- [22] E. L. Demeulemeester and W. Herroelen, *Project Scheduling: A Research Handbook*, vol. 102, Springer Science & Business Media, 2002.
- [23] S. Martello and P. Toth, "Heuristic algorithms for the multiple knapsack problem," *Computing*, vol. 27, no. 2, pp. 93–112, 1981.