Computers & Operations Research ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

# A two-phase scheduling method with the consideration of task clustering for earth observing satellites

Guohua Wu*, Jin Liu, Manhao Ma, Dishan Qiu

*Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, 47 Yanzheng Street, Changsha, Hunan 410073, China*

ARTICLE INFO

*Keywords:*
Satellite scheduling
Task clustering
Clique partition
Ant colony optimization
Local search

ABSTRACT

Satellite observation scheduling plays a significant role in improving the efficiency of satellite observation systems. Although extensive scheduling algorithms have been proposed for the satellite observation scheduling problem (SOSP), the task clustering strategy has not been taken into account up to now. This paper presents a novel two-phase based scheduling method with the consideration of task clustering for solving SOSP. This method comprises two phases: a task clustering phase and a task scheduling phase. In the task clustering phase, we construct a task clustering graph model and use an improved minimum clique partition algorithm to obtain cluster-tasks. In the task scheduling phase, based on overall tasks and obtained cluster-tasks, we construct an acyclic directed graph model and utilize a hybrid ant colony optimization coming with a mechanism of local search, called ACO–LS, to produce optimal or near optimal schedules. Extensive experimental simulations demonstrate the efficiency of the proposed scheduling method.



## 1. Introduction

Earth observing satellites (EOSs), orbiting the Earth, are able to collect images of specified areas of the Earth surface at the request of users [1]. They are playing key roles in environment surveillance, intelligence reconnaissance and others, prompting many countries to increase investments to develop EOSs and associated techniques. For instance, China plans to launch four small optical satellites and four small SAR satellites to form a natural disaster-monitoring constellation [2]. To make full use of scarce satellite resources, effective observation scheduling is of great significance [2]. Since users' observation requests usually exceed the capabilities of existing satellite resources, the satellite observation scheduling problem (SOSP) is categorized as an over-subscribed problem. In fact, SOSP can be seen as a kind of multi-dimensional knapsack problem [3], which is NP-hard.

In this study, we will introduce a new strategy called task clustering when addressing SOSP, aiming to improve the scheduling efficiency. As shown in Fig. 1, a satellite sensor generates a dummy observation strip of a certain width and length when passing over targets. The width and length of the strip are determined by the altitude of the satellite, the field of view of the sensor, the slewing angle of the sensor and the observation duration of the sensor. Generally, the altitude of the satellite and the field of view of the sensor are fixed. However, we can rationally tune the slewing (pointing) angle and the observation duration of the sensor to enable an observation strip to cover multiple targets. We view the request of observing a target as an observation task. Then multiple observation tasks can be merged into a composite task and then completed by an observation activity. A composite task is called a cluster-task in this study. For example, in Fig. 1, task $t_6$, $t_7$ and $t_8$ can be combined into a cluster-task and finished by one observation activity. The process that integrates multiple observation tasks into a cluster-task is called task clustering.

In our opinion, considering task clustering in the course of solving SOSP at least has following three advantages.

(1) Task clustering can enable a satellite to finish more tasks at the cost of fewer sensor opening times. In each orbit, times for a satellite turning on its sensor are limited due to electrical constraints and the limitation of onboard energy. A satellite has to first open its sensor before it starts an observation. That is to say, the number of observations in each orbit is limited accordingly. Noticeably, task clustering enables the satellite to finish more tasks with fewer observations (i.e. fewer sensor opening times). For example, as shown in Fig. 1, traditionally, the sensor needs to open for eight times (i.e. eight observations) to execute the eight observation tasks. In contrast, tasks clustering enable the satellite to finish the eight tasks with four sensor opening times.

* Corresponding author. Tel.: +86 1357 48941 25.
*E-mail address:* guohuawu.nudt@gmail.com (G. Wu).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.02.009

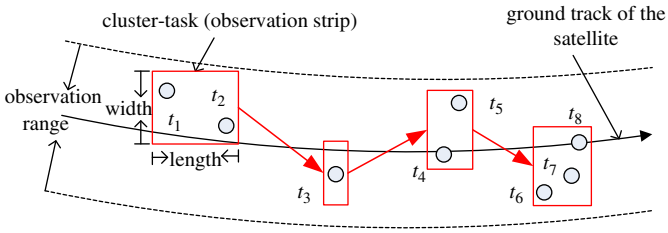


**Fig. 1.** Task clustering of satellite observing.

(2) Task clustering can enable some previously mutual exclusive tasks to be finished simultaneously. When a satellite goes to finish two tasks consecutively, enough setup time is required for the satellite to turn on its sensor and slew its sensor to an appropriate angle to point at a special target. If there is not enough duration between the time-windows of two consecutive tasks, they will be mutual exclusive. However, if these mutual exclusive tasks can be integrated into a cluster-task, they can be completed together. For example, in Fig. 1, we assume that tasks $t_4$ and $t_5$ are mutual exclusive. Then traditionally either $t_4$ or $t_5$ will be scheduled. While after considering task clustering, $t_4$ or $t_5$ can be clustered and finished together. Similar situation can be found in tasks $t_6$, $t_7$ and $t_8$.

(3) Task clustering can enable a satellite to accomplish more tasks while reduce sensor slewing times, resulting in less energy consumption. As shown in Fig. 1, we assume that tasks $t_1$ and $t_2$ can be finished consecutively with slewing angle $\theta_1$ and $\theta_2$, respectively. Traditionally, the satellite have to slew its sensor for two times to accomplish $t_1$ and $t_2$, with total slewing angle being $|\theta_1|+|\theta_2|$. In contrast, after considering task clustering, the satellite only needs to slew its sensor for only one time with angel $\frac{|\theta_1+\theta_2|}{2}$.

It is therefore meaningful to develop a scheduling method integrated with task clustering to address SOSP. Unfortunately, to our knowledge, no study has taken into account task clustering up to now. This work aims to fill this gap.

Another motivation of this work is to apply Ant Colony Optimization (ACO) to the solution of SOSP. At present, except ACO, many heuristic algorithms have been employed to solve SOSP, such as Local Search, Simulated Annealing, Tabu Search and Genetic Algorithm. In fact, ACO has been demonstrated to be very effective in solving complex combinatorial optimization problems. As shown in Fig. 1, scheduling satellite observations amounts to arranging an observation sequence for each orbit of satellites. This process is naturally analogous to the route search process of ants, which motivated us to investigate a specialized ACO to tackle SOSP.

Since SOSP is NP-Hard and the problem of task clustering is complicated as well, it might be intractable to integrate task clustering into the task scheduling process dynamically. To reduce complexities, we propose a two-phase based strategy that first implements task clustering then conducts task scheduling.

The major contributions of this study are summarized as follows:

(1) We put forward a two-phase scheduling strategy with task clustering to solve SOSP.
(2) We analyze the constraints involved in task clustering, and propose effective methods to calculate the slewing angle and the time-window of each cluster-task.
(3) We construct a graph model to formulate task clustering and present a minimum clique partition method to generate cluster-tasks.
(4) We give an integer programming model and an acyclic directed graph model to formulate SOSP, considering overall tasks and cluster-tasks. In addition, based on the models, we propose a hybrid ant colony optimization mixed with local search (ACO–LS) to obtain satisfactory schedules.

The rest of this paper is organized as follows: Section 2 reviews related work in literatures. Section 3 constructs a graph model for task clustering and employs an improved clique partition method to obtain cluster-tasks. Section 4 builds an integer programming model and an acyclic directed graph model for SOSP, and proposes the ACO–LS algorithm generate high quality schedules. Section 5 gives simulation experiments and performance analysis. Section 6 concludes this paper with a summary and direction of future work.

## 2. Related works

Many early studies concentrated on SOSP with single satellite. For instance, Potter and Gasch [4] applied an improved greed algorithm to mission planning and scene acquisition scheduling for the Landsat 7 satellite. Wolfe and Sorensen [5] compared the capability of a dispatch algorithm, a look-ahead algorithm, and a genetic algorithm. They drew the conclusion that the genetic algorithm was most efficient with problem size increasing. Vasquez and Hao [6] translated the scheduling problem into the well-known knapsack model. Moreover, they proposed a Tabu Search algorithm to solve the model. In another paper [7], Vasquez and Hao presented a Partition-based approach to obtain tight upper bounds to evaluate the previous Tabu Search solution. Bensana et al. [3] viewed the scheduling of the Spot 5 satellite as a value constraint-satisfaction problem and utilized exact and approximate methods to solve it, respectively. Gabrel and Vanderpooten [8] proposed an acyclic graph model to formulate the scheduling problem of Spot 5. They generated multiple efficient paths and selected a satisfactory path by a multiple criteria interactive procedure. Lin et al. [9] adopted the mathematical programming method to acquire a near-optimal schedule of the problem. Baek et al. [10] applied a new genetic algorithm to simulations of an actual satellite mission scheduling problem. Miguel et al. [11] presented a strengthened formulation for the daily photograph scheduling of Spot 5, based on valid inequalities arising in node packing and 3-regular independence system polyhedra.

Increasing attention has been directed toward the scheduling of multiple satellites. Wang et al. [12] proposed a multi-objective EOS imaging scheduling method based on Strength Pareto Evolutionary Algorithm 2. Bianchessi et al. [1] investigated the scheduling problem for a constellation of agile satellites. A Tabu search algorithm was devised to produce solutions, which were further evaluated by a column generation algorithm. Frank et al. [13] adopted a constraint-based interval framework to represent the resources of EOSs and proposed a heuristics to guide the solution search process. Lemaître et al. [14,15], Beaumet et al. [16] and Habet et al. [17] investigated the scheduling method for agile Earth observing satellites. Globus et al. [18] employed a stochastic climbing hill, a simulated annealing algorithm, a genetic algorithm, an iterated sampling algorithm and a squeaky wheel optimization to solve the problem, respectively. Their experimental results demonstrated that Simulated Algorithm outperformed the other algorithms. Mansour et al. [19] developed a genetic algorithm for solving the scheduling problem using a new genome representation for maximizing multi-criteria objective including the profit and the number of acquired photographs.

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

Note that most of the current studies are oriented to specified satellite platforms or respective real-world applications. For example, several literatures focused on the scheduling of the Spot series satellites [6,7,19], while some other literatures aimed at PLEIADES constellation [1], COSMO-SkyMed constellation [20], Landsat 7 satellite, ROCSAT-II [9] and FORMOSAT-2 [19], respectively. Most studies took into account optical instruments; however, SAR instruments were considered in [22]. In addition, for simplicity, partial studies ignored some factors when modeling the problem. For instance, Refs. [5,8,23] did not consider the constraints of memory storage capacity and onboard energy capacity. In Refs. [9,13,21,24] weather conditions, especially the cloud, were considered. Emergency tasks were considered in Ref. [25].

Algorithms applied to SOSP can be classified into exact and approximate algorithms. Generally, exact algorithms, such as the Russian Dolls Search and Branch & Bound [3], are feasible in tackling problems with relative small size. And approximate algorithms, such as simulated annealing [18], Tabu search [6] and genetic algorithm [19], are more efficient in coping with problems of large size.

In this paper, we pay attention on the issue of task clustering and its integration with scheduling. We develop a two-phase strategy that consists of a task clustering phase and a task scheduling phase. We assume that sensors of satellites considered in our study are able to slew laterally. This study considers primary constraints concerning maximum sensor opening (slewing) times, memory storage capacity, onboard energy capacity and setup time between consecutive tasks.

## 3. Task clustering model and algorithm

In this section, we investigate the constraints of task clustering, including slewing angle related constraint and time-window related constraint. Then a task clustering graph model is constructed, such that a clique represents a cluster-task. An improved minimum clique partition algorithm is presented to generate cluster-tasks.

### 3.1. Task clustering constraints

The precondition for combining multiple tasks into a cluster-task is that these tasks can be finished with the same slewing angle and time-window, which constraints the task clustering process.

- Slewing angle related constraint
  One prerequisite that multiple tasks can be clustered together is that they can be completed by a sensor with the same slewing angle. That is, the corresponding multiple targets can be covered by the same observation strip.
  As shown in Fig. 2, we define that

  $\sigma$ denotes the field of view of the sensor,
  $\theta_i$ denotes the slewing angle of the sensor to observe target $t_i$,
  $\overrightarrow{se}$ denotes the boresight of the sensor,
  $o$ denotes the nadir of the satellite,
  $\phi_i$ denotes the deviation angle from target $t_i$ to nadir $o$.

  A target means a task here. As shown in Fig. 2a, the appropriate slewing angle $\theta_i$ for observing the target $t_i$ equals to $\phi_i$. In fact, we can see that any slewing angle among $[\phi_i-\sigma/2,\ \ \phi_i+\sigma/2]$ is feasible for observing target $t_i$. Then for target $t_k$ and $t_i$, as shown in Fig. 2b, we can choose a slewing angle among

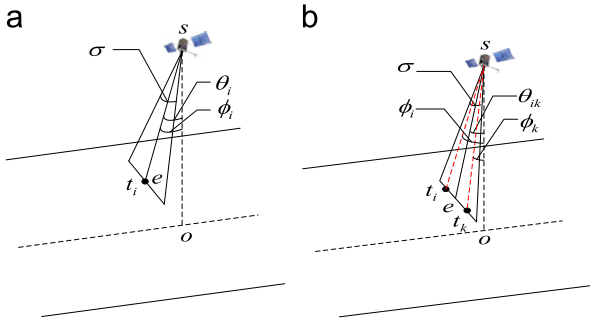


**Fig. 2.** Illustration of the field view angle and the slewing angle of a sensor.

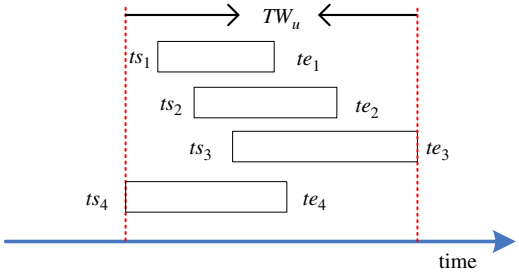


**Fig. 3.** Demonstration of calculating the time-window for a cluster-task.

$\theta_{ik} \in ([\phi_i-\sigma/2,\ \phi_i+\sigma/2] \cap [\phi_k-\sigma/2,\ \phi_k+\sigma/2])$ to observe these two targets simultaneously. Let $\Delta\theta_i=[\phi_i-\sigma/2,\ \ \phi_i+\sigma/2]$, i.e., $\Delta\theta_i$ denotes the feasible sensor slewing angle range for observing target $t_i$. So if task $t_i$ and $t_k$ can be clustered together, they should meet $\Delta\theta_k \cap \Delta\theta_i \neq \emptyset$.
More generally, if multiple tasks, say $t_1,t_2,\ldots,t_n$, can be clustered, they should satisfy

$$\Delta\theta_1 \cap \Delta\theta_2 \cap \cdots \cap \Delta\theta_n \neq \emptyset \tag{1}$$

Suppose that cluster-task $t_u$ is clustered form tasks $t_1,t_2,\ldots,t_n$. Let $\theta_u$ denote the sensor slewing angle for cluster-task $t_u$ and $\Delta\theta_u$ denote the feasible sensor slewing angle range, i.e., $\theta_u \in \Delta\theta_u$. Then we have

$$\Delta\theta_u = \Delta\theta_1 \cap \Delta\theta_2 \cap \cdots \cap \Delta\theta_n \tag{2}$$

Here we simply let $\theta_u$ be equal to the mean value of interval $\Delta\theta_u$, namely

$$\theta_u = \tfrac{1}{2}[\sup(\Delta\theta_u)+\inf(\Delta\theta_u)] \tag{3}$$

- Time-window related constraint
  We should calculate the time-window for each cluster-task, permitting the satellite to finish its component tasks in a common temporal interval.

Suppose that cluster-task $t_u$ is clustered form tasks $t_1,t_2,\ldots,t_n$. The time-window of $t_u$ is denoted by $TW_u=[ts_u,\ te_u]$. To calculate $TW_u$ for $t_u$, we let

$$ts_u = \min\{ts_l | l=1,2,\ldots,n\}, te_u = \max\{te_l | l=1,2,\ldots,n\} \tag{4}$$

Fig. 3 gives a demonstration of the time-window calculation process for a cluster-task.

Let $\Delta T$ denote the longest duration allowed for a continuous observation. Therefore, if tasks $t_1,t_2,\ldots,t_n$ can be integrated into a cluster-task $t_u$, time-window $TW_u$ should satisfy

$$te_u - ts_u \le \Delta T, \tag{5}$$

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

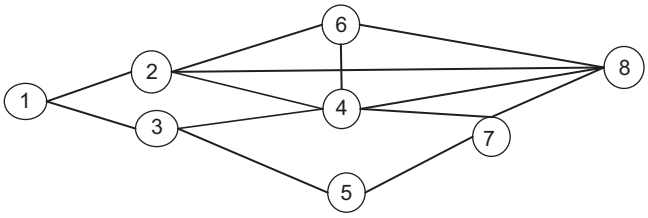


**Fig. 4.** Task clustering graph model.

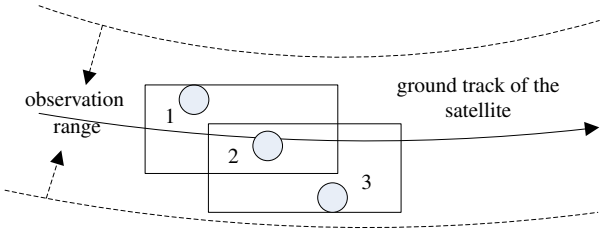


**Fig. 5.** Demonstration of clustering relation.

Obviously, if $TW_u$ dose not satisfy constraint (5), tasks $t_1,t_2,\ldots,t_n$, cannot be clustered into $t_u$.

Note that the priority of a cluster-task is the sum of priorities of its component tasks.

### 3.2. Task clustering model and algorithm

We introduce a graph $G=(V(G),\ E(G))$ to model task clustering, where $V(G)$ is a vertex set and $E(G)$ is an edge set. Each vertex represents a task. As shown in Fig. 4, any two vertices will be linked by an edge if they can be clustered, i.e., they satisfy constraints (2) and (5).

It is clear that clustering relation is reflexive and symmetric. Moreover, in Fig. 5, we can find that clustering relation is non-transitive. As shown in Fig. 5, we assume task 1 and 2 can be combined together, and task 2 and 3 can be clustered together as well. However, we can observe that task 1 and 3 cannot be clustered together. As a result, clustering relation is non-transitive.

Thereby we can come to a conclusion that multiple tasks can be clustered together if and only if any two of them can be clustered together. Since each task is represented by a vertex, a cluster-task can be represented by a clique in which any two vertices are adjacent. For instance, as shown in Fig. 4, vertices 2, 4, 6, and 8 induce a clique, hence corresponding tasks 2, 4, 6, and 8 can be merged into a cluster-task. So do vertices 4, 7, and 8.

As a result, based on the graph model above, to obtain cluster-tasks is equal to obtain cliques. We then can formulate task clustering into a minimum clique partition problem, which is NP-complete [26]. To solve this problem, Tseng [27] provides a primary clique partition algorithm and Kim [28] presented a more efficient algorithm called F_Clique. We improve F_Clique by considering priorities of vertices. The improved version of F_Clique is described as below.

Let $V$ be the vertex set and $N(p)$ be the set of neighbors of vertex $p$;

**while** $V!=\emptyset$ **do**
  Pick a vertex $p$ with largest degree from $V$. And if such $p$ are not unique, pick the $p$ with highest priority;
  **while** $N(p)!=\emptyset$ **do**
    Pick a neighbor $q$ of $p$, $q\in N(p)$, such that the number of their common neighbors is maximum. If such $p$ are not unique, pick the $p$ with least edges being deleted.
    Again, if such $p$ are still not unique, pick the $p$ with highest priority;
    Combine $q$ and $p$ into a new $p$;
    Delete edges from $q$ and $p$ that are not connected to their common neighbors;
    Reset neighbor collection $N(p)$ for the new $p$;
  **end while**
  Output the cluster-task denoted by $p$;
  Delete $p$ from $V$;
**end while**

## 4. Scheduling model and algorithm

In this section, we will introduce an integer-programming model and an acyclic directed graph model to describe SOSP, and develop a hybrid ant colony optimization mixed with local search (ACO–LS) to produce effective schedules.

We consider overall tasks and obtained cluster-tasks as candidate scheduling tasks. Indeed, the additional cluster-tasks will increase the problem size. It is noticeable that $n$ tasks can generate at most $n/2$ cluster-tasks. In regard to a problem with $n$ tasks, the number of candidate tasks participating the scheduling may increase to $3n/2$. In fact, experimental simulations demonstrated that the size of cluster-tasks under a general case is relative small (no more than $n/5$). However, task clustering can significantly improve the scheduling efficiency. Hence, considering cluster-tasks into the scheduling is worthy.

### 4.1. Integer programming model for satellite scheduling

Fig. 1 shows that the solution of SOSP aims to arrange an observation sequence for each orbit of satellites to maximize the return, subjecting to corresponding constraints. Different orbits of satellites can be fairly viewed as the same kind of resource with certain observation capabilities. For simplicity, we organize all the orbits together, to translate a SOSP with multi-satellite and multi-orbit into a SOSP with multi-orbit. Let $O=\{o_j|j=1,2,\ldots M\}$ be the collection of orbits, where $M$ denotes the total number of orbits.

Let $T=\{t_i|i=1,2,\ldots N\}$ denote the collection of overall candidate scheduling tasks, including tasks and cluster-tasks. Decision variable $x_{ij}$ indicates whether task $t_i$ is scheduled to be executed in orbit $o_j$. We set

$$x_{ij}=\begin{cases}1, & \text{if task } t_i \text{ is scheduled to be executed in orbit } o_j,\\ 0, & \text{otherwise.}\end{cases}$$

In addition, to facilitate to present the mathematical model, some notations are defined as below.

| | |
|---|---|
| $TW_i=[ts_i,te_i]$ | time-window of task $t_i$ |
| $p_i$ | priority associated with task $t_i$. |
| $y_{ih}$ | 1–0 variable indicating whether task $t_h$ will be executed after $t_i$ |
| $W_j$ | memory storage capacity in orbit $o_j$ |
| $w_j$ | memory consumption rate by an observation in orbit $o_j$ |
| $E_j$ | energy capacity in orbit $o_j$ |
| $eo_j$ | energy consumption rate by an observation in orbit $o_j$ |
| $es_j$ | energy consumption rate by a sensor slewing operation in orbit $o_j$ |
| $v_j$ | sensor slewing velocity in orbit $o_j$ |
| $a_j$ | required time for opening the sensor in orbit $o_j$ |
| $\theta_i$ | slewing angle to observe task $t_i$ in orbit $o_j$ |

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

$c_j$ maximum times for a satellite opening its sensor in orbit $o_j$

The scheduling object is to maximize the profit measured by the sum of priorities of scheduled tasks:

$$\max \sum_{j=1}^{M} \sum_{i=1}^{N} x_{ij} \cdot p_i \tag{6}$$

Complex constraints need to be satisfied in the scheduling process. Refs. [15,29] detailed the primary constraints. The constraints considered in this study are described as below.

Each task requires only one observation.

$$\sum_{j=1}^{M} x_{ij} \le 1, \quad i = 1,2,\ldots,N \tag{7}$$

In each orbit $o_j$, the energy to be consumed should not exceed the maximum capacity. The energy is mainly consumed by observation and sensor slewing activities.

$$\sum_{i=1}^{N} x_{ij} \cdot eo_j \cdot (te_i - ts_i) + \sum_{i=1}^{N} \sum_{h=1}^{N} x_{ij} \cdot x_{hj} \cdot y_{ih} \cdot es_j \cdot (|\theta_i - \theta_h| / v_j) \le E_j, \quad j = 1,2,\ldots,M \tag{8}$$

In each orbito$_j$, the memory storage to be consumed should not exceed the maximum capacity.

$$\sum_{i=1}^{N} x_{ij} \cdot w_j \cdot (te_i - ts_i) \le W_j, \quad j = 1,2,\ldots,M \tag{9}$$

Enough setup time is required for a satellite to open its sensor and calibrate the pointing angle of its sensor.

$$ts_h - te_i \ge a_j + |\theta_h - \theta_i| / v_j, \ i,h = 1,2,\ldots,N, \quad y_{ih} = 1, \quad j = 1,2,\ldots,M, \tag{10}$$

A satellite needs to open (i.e. power on) its sensor before performing each observation task (i.e. one sensor opening time is corresponding to one observation task). In each orbit $o_j$, the number of sensor opening times (denoted by $c_j$) for a satellite is limited due to electrical constraints and the limitation of onboard energy. That is to say, limited number of observation tasks can be executed in each orbit. As a result, in each orbit, the number of execution tasks should not be larger than the number of sensor opening times.

$$\sum_{i=1}^{N} x_{ij} \le c_j, \quad j = 1,2,\ldots,M \tag{11}$$

### 4.2. Acyclic directed graph model for satellite scheduling

Based on possible observation orders of tasks, we construct an acyclic directed graph model for each orbit $o_j$. As shown in Fig. 6, $G_j = (V_j, E_j)$ denotes the graph model for orbit $o_j$, where $V_j$ is the node set and $E_j$ is the edge set. A node in $V_j$ represents a task possessing a time-window in orbit $o_j$. Any two tasks that satisfy setup time constraint (10) will be linked by a directed edge (i.e., indicate that one task can be executed successively after another). To describe the problem more clearly, we add to each $G_j$ two dummy tasks $b_j$ and $e_j$, which represent the source node and sink node of the graph. Dummy tasks do not consume any resource.

Note that the same node may belong to distinct sub-graphs. This is because one task may has time-windows in multiple different orbits. For example, in Fig. 6, task $t_i$ has time-windows in both orbit $o_1$ and $o_2$, so it belongs to both $G_1$ and $G_2$ simultaneously. The latter scheduling algorithm determines task $t_i$ to be executed whether in orbit $o_1$ or $o_2$.

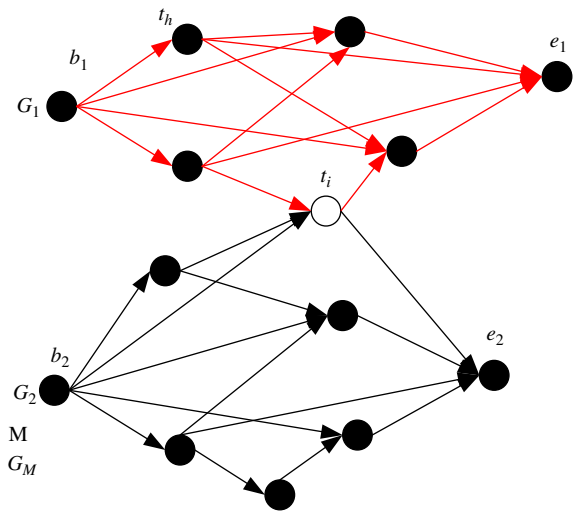


**Fig. 6.** Acyclic directed graph model of multi-satellite observation scheduling.

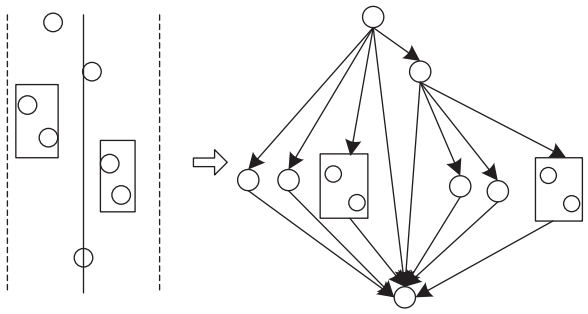

**Fig. 7.** Acyclic directed graph model construction process.

If a task is a component of a cluster-task, the tasks and the cluster-task are exclusive in the graph model, to be in consistent with constraint (7). Fig. 7 shows the model construction process with the consideration of overall tasks and obtained cluster-tasks.

### 4.3. Hybrid ant colony optimization mixed with local search

Ant colony optimization, proposed by Dorigo [30], is a heuristic algorithm in which a colony of artificial ants cooperates in finding good solutions to difficult discrete optimization problems [31]. It is efficient in solving various path-searching problems, such as the traveling salesman problem [32], the vehicle routing problem [33], and the flow shop scheduling problem [34]. In this study, we present a novel ACO–LS algorithm that integrates traditional ACO with a local search, which significantly speeds up the solving process and improves the solution quality. The framework of ACO–LS is described as below.

```
Initialize parameters;
while (termination conditions not met) do
  for i ← 1 to AntSize do
      Construct a feasible soulution for each ant;
      Apply the local search to each acquired solution;
  end for
  Update the pheromone;
end while
```

#### 4.3.1. Feasible solution construction

A state transition rule guides the path searching process of ants. Assume that Allow$_{ji}^{m}$ is the collection (ant's visibility set)

of possible successors of task $t_i$ when the $m$th ant is ready to transfer from task $t_i$ to a successor task in $G_j$. Any task $t_l$ in $\text{Allow}_{ji}^m$ should satisfy following three conditions:

- There exists a directed edge from $t_i$ to $t_l$, to meet constraint (10).
- Task $t_l$ has not been arranged into the other orbits, to meet constraint (7).
- If task $t_l$ is added to the current schedule, the schedule still meets constraints (8), (9) and (11).

If $\text{Allow}_{ji}^m \neq \emptyset$, the $m$th ant will select a task $t_h \in \text{Allow}_{ji}^m$ as its next observation task according to the following probability sate transition rule:

$$t_h = \begin{cases} \max\limits_{t_l \in \text{Allow}_{ij}^m} \{[\tau_{jil}]^\alpha \times [\eta_{jil}]^\beta\} & \text{if } q \le q_0 \\ \psi & \text{otherwise,} \end{cases} \tag{12}$$

where, $\tau_{jih}$ is the pheromone trail associated with the edge from $t_i$ to its successor task $t_h$ in orbit $o_j$. The parameter $q_0$ $(0 \le q_0 \le 1)$ determines the relative frequency between exploitation and exploration. The value of each pheromone is initialized as $\tau_0$. $\eta_{jih}$ is the greedy heuristic information. $\alpha$ and $\beta$ stand for the weight of pheromone and heuristic information, respectively.

Furthermore, $\psi$ is random variable which gives the probability of choosing task $t_h$ to execute after $t_i$ in orbit $o_j$. The value of $\psi$ is determined by the probability distribution given below.

$$\text{Pr}^m(j,t_i,t_h) = \begin{cases} [\tau_{jih}]^\alpha [\eta_{jih}]^\beta / \sum\limits_{t_l \in \text{Allow}_{ij}^m} [\tau_{jil}]^\alpha [\eta_{jil}]^\beta & \text{if } t_h \in \text{Allow}_{ji}^m, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

To make effective use of various heuristic information associated in the scheduling, The greedy heuristic information $\eta_{jih}$ is defined as

$$\eta_{jih} = \frac{\lambda \cdot p_h}{\max\{p_l | t_l \in \text{Allow}_{ji}^m\}} \cdot (\omega \cdot \frac{\max\{ts_l - te_i | t_l \in \text{Allow}_{jh}^m\} - (ts_h - te_i)}{\max\{ts_l - te_i | t_l \in \text{Allow}_{jh}^m\}} + \upsilon \cdot \frac{\max\{\Delta\theta_{il} | t_l \in \text{Allow}_{ji}^m\} - \Delta\theta_{ih}}{\max\{\Delta\theta_{il} | t_l \in \text{Allow}_{ji}^m\}}), \tag{14}$$

where $p_h$ is the priority of task $t_h$, $ts_h - te_i$ denotes the temporal gap between $t_i$ and $t_h$, and $\Delta\theta_{ih}$ $(\Delta\theta_{ih} = |\theta_h - \theta_i|)$ denotes the sensor slewing angle range from $t_i$ to $t_h$. Coefficients $\lambda$, $\omega$ and $\upsilon$ stand for weight.

The definition of $\eta_{jih}$ indicates that it is more likely to choose a successor task with higher priority, shorter temporal gap from the current task, and smaller slewing angle deviated from the current task.

The procedure that the $m$th ant constructs a feasible schedule is described as below.

**Step 1**: Judge whether orbit set $O$ is empty. If $O = \emptyset$, the algorithm terminates, otherwise, go Step 2.

**Step 2:** The ant randomly selects an orbit $o_j \in O$ and starts to search a feasible path in $G_j$. Let $O = O \backslash o_j$ and go Step 3.

**Step 3**: Search the path along with directed edges in $G_j$ according to the ant's state transition rule repeatedly until no feasible successor task exists (i.e. for current task $t_i$, $Allow_{ji}^m = \emptyset$). Go to Step 1.

The solution construction procedure of an ant is described as follows.

**while** $O! = \emptyset$ **do**
  Randomly select an obit $o_j \in O$ and start to search a path in graph $G_j$;
  $t_i \leftarrow b_j$;
  Construct $Allow_{ji}^m$;
  **while** $Allow_{ji}^m! = \emptyset$ **do**
  Select and add a task $t_h \in Allow_{ji}^m$ to the schedule according to **formula**(12);
  $t_i \leftarrow t_h$;
  Reconstruct $Allow_{ji}^m$;
  **end while**
  $O \leftarrow O \backslash o_j$;
**end while**

#### 4.3.2. Local search

An increasing number of evidences have shown that the plus of local search (LS) will strengthen the capability of traditional ant colony optimization (ACO) [31,35]. This is resulted from the different optimization mechanisms used in these two algorithms: ACO is based on the stepwise construct of solution ingredients, while LS is based on the partial adjustment of a previous solution. Thereby, ACO and LS can supplement with each other. Namely, ACO can provide promising initial schedules for LS, which in turn can further optimize the schedules obtained by ants efficiently.

The LS algorithm employed in our study is straightforward but effective. In each orbit, we orderly select an unscheduled task with the highest priority and try to insert it into the existing schedule. If the insertion increases the profit of the schedule, we will update the schedule in accordance with the insertion. Otherwise, the insertion is invalid and the schedule will not be updated. This LS is based on the classical add and move operators. The neighborhood structure of LS is illustrated in Fig. 8.

Let $TU_j$ denote the collection of unscheduled tasks associated in orbit $o_j$. The procedure of LS is described as below.

**for** $j \leftarrow 1$ to $M$ **do**
  **while** $TU_j$ ! = $\emptyset$ **do**
  Select a task $t_i$ with highest priority from $TU_j$;
  $TU_j \leftarrow TU_j \backslash t_i$;
  Add tasks coflicting with $t_i$ to collection $F_{ji}$;
  Let $p_F$ denote the sum priorities of tasks in $F_{ji}$;
  **if** $p_i > p_F$ **then**
  Insert $t_i$ into the schedule;
  Remove tasks coflicting with $t_i$ from the schedule;
  **end if**

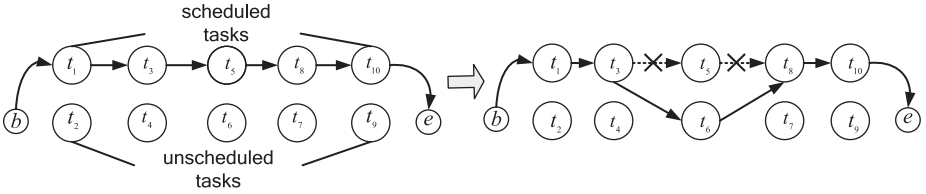


**Fig. 8.** Neighborhood structure of LS.

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

**end while**
**end for**

#### 4.3.3. Pheromone update rule

After all ants have found their paths, pheromone trails on paths will be updated according to the following formula:

$$\tau_{ih} = \begin{cases} (1-\rho)\tau_{ih} + \sum_{m=1}^{\text{AntSize}} Q_m + Q_G & \text{if } r \text{ and } (0,1) < q_1 \\ (1-\rho)\tau_{ih} + \sum_{m=1}^{\text{AntSize}} Q_m + Q_L & \text{otherwise} \end{cases}, \quad (15)$$

where $Q_m = f(s_m)/p_{\text{all}}$, $Q_G = f(s_G)/p_{\text{all}}$, and $Q_L = f(s_L)/p_{\text{all}}$, $s_m$ is the current schedule obtained by the $m$th ant, $f(s_m)$ is the fitness value of $s_m$, measured by the sum priorities of tasks in $s_m$, $s_G$ is the best-so-far schedule, $s_L$ is the iteration-best schedule, $p_{\text{all}}$ is the sum priorities of all tasks possessing at least one time-window, $\rho$ $(0 < \rho < 1)$ is the pheromone evaporation rate.

Formula (14) indicates that ACO–LS utilizes the schedule information of every ant to update pheromones. Moreover, the best-so-far schedule will be selected to reinforce pheromones with probability $q_1$ $(0 \le q_1 \le 1)$, and the iteration-best schedule will be selected to reinforce pheromones with probability $1-q_1$. This update strategy strives for a balance between the exploitation and exploration ability of ACO–LS.

To prevent pheromones from being accumulated or evaporated too much, we limit them within a value range $[\tau_{\min}, \tau_{\max}]$.

$$\tau_{ih} = \begin{cases} \tau_{\max}, & \tau_{ih} > \tau_{\max} \\ \tau_{ih}, & \tau_{\min} \le \tau_{ih} \le \tau_{\max} \\ \tau_{\min}, & \tau_{ih} < \tau_{\min} \end{cases} \quad (16)$$

#### 4.3.4. Termination criterions of ACO–LS

ACO–LS adopts two termination criterions. The first one: if overall iterations reach to *Max_Iter*, the algorithm terminates. The second one: if the solution has no improvement after every *Mid_Iter* iterations, the algorithm terminates. Values of *Max_Iter* and *Mid_Iter* should be initialized before the algorithm works.

## 5. Experimental Simulation

### 5.1. Simulation setup

To evaluate performance of the proposed algorithm, we needed to conduct extensive simulations. As introduced in Section 2, previous researches usually focused on specific satellites or constellations, causing that constraints and models considered by distinct academic groups were different to some extent. Thus, there is no benchmark instance consistent with the model presented in this study.

We defined two groups of reference scenarios to test the efficiency of our scheduling method. The first group contains two on-orbit satellites of China, i.e., JB-3A and JB-3C; and the second group contains four on-orbit satellites, i.e., JB-3A, JB-3C, CBERS-1 and CBERS-2 (CBERS series satellites were cooperatively developed by China and Brazil). Each satellite circles the Earth about 100 min each time and runs about 14 orbits a day. Sensors on satellites can slew laterally among angle range $[-33^\circ, 33^\circ]$. Each group includes ten scenarios with 100 to 1000 point targets with a step of 100. The point targets are randomly distributed on the Earth surface with latitude among $[-33^\circ, 60^\circ]$ and longitude among $[0^\circ, 153^\circ]$. Each target is associated with a priority uniformly distributed among [2,10]. The scheduling horizon is 24 h. Time-windows and slewing angles associated with targets and satellites were calculated in prior using Analytical Graphics Inc.'s professional software named Satellite Tool Kit, also known as the STK (see www.stk.com). The orbit data of JB-3A, JB-3C, CBERS-1 and CBERS-2 can be downloaded from the satellite database of STK.

**Table 1**
Experimental parameters.

| $\sigma$ | $\Delta T$ | $W_j$ | $w_j$ | $E_j$ | $eo_j$ | $es_j$ | $v_j$ | $a_j$ | $c_j$ |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 120 | 1000 | 1 | 1500 | 1 | 1 | 1 | 10 | 10 |

Besides, other presumed attributes of satellites and sensors and related parameters used in our experimental tests are listed in Table 1.

The algorithm was coded in C++ and run on a PC with Intel Pentium Dual E2108 (2.00 GHz) CPU and 1.0 GB RAM under Windows XP.

### 5.2. Computational results

According to our extensive experimental results analysis, the parameters of the ACO–LS algorithm are configured and listed in Table 2.

To assess ACO–LS's effectiveness, we compared it with the Russian Doll Search (RDS) algorithm [3,36] and the Partition-based Approach (PBA) [7].

RDS is an exact algorithm based on Branch-and-Bound techniques. It replaces one search by several searches on nested subproblems and utilizes the results of each search to improve the lower bound on the global valuation. This algorithm generated good results for small scale scheduling instance of Spot 5.

PBA follows the divide and conquer principle and integrates dynamic programming with Tabu Search. It could produce tight upper bound for scheduling problems of Spot 5.

It should be noted that the scheduling problem considered in this paper is somewhat different from that of Spot 5. Some modifications were conducted to employ RDS and PBA. (1) Since one optical sensor was mounted on each satellite, stereo imaging requests were not considered and the so-called ternary constraints in Ref. [7] were not taken into account. (2) Besides the memory storage constraint, the analogous knapsack constraint concerning energy capacity was considered, which was relaxed like the memory storage constraint when applied PBA.

The computational results of the two groups of scenarios are presented in Table 3 and Table 4, respectively, where

*SN* denotes the serial number of cases,
*TN* denotes the number of time-windows between point targets and satellites,
*EN* denotes the number of targets owning at least one time-window,
*CN* denotes the number of obtained cluster-tasks,
$R$ and $F$ denote the obtained profit and scheduled tasks by the RDS algorithm,
$R^*$ and $F^*$ denote the upper bounds obtained by PBA,
$R_{\max}$, $R_{\min}$ and $\overline{R}$, respectively denote the best, worst and mean obtained profit by ACO–LS,
$\overline{F}$ denotes the average number of the finished tasks,
and $\overline{T}$ denotes the mean running time.

We compute the gaps between RDS and ACO–LS, and that between PBA and ACO–LS, respectively. Assume $Gap_R$ denote the gap between RDS and ACO–LS, and $Gap_p$ denote the gap between PBA and ACO–LS. Then we let $Gap_R = (R-\overline{R})/R$ and $Gap_p = (R^*-\overline{R})/R^*$.

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

**Table 2**
Parameters of the ACO–LS algorithm.

| $\alpha$ | $\beta$ | $\lambda$ | $\omega$ | $v$ | $\rho$ | $q_0$ | $q_1$ | $\tau_{max}$ | $\tau_{min}$ | $\tau_0$ | AntSize | *Max_Iter* | *Mid_Iter* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 0.5 | 0.3 | 0.2 | 0.1 | 0.9 | 0.85 | 4 | 0.01 | 0.1 | 10 | 300 | 30 |

**Table 3**
Simulation results of scenarios with two satellites.

| *SN* | *EN* | *TN* | *CN* | RDS | | PBA | | ACO–LS | | | | | $Gap_R$ | $Gap_P$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $R$ | $F$ | $R^*$ | $F^*$ | $R_{max}$ | $R_{min}$ | $\overline{R}$ | $\overline{F}$ | $\overline{T}$ (s) | | |
| B1 | 84 | 126 | 5 | **482** | **72** | 487 | 74 | **482** | **482** | **482** | **72** | 68.85 | 0 | 0.01 |
| B2 | 134 | 266 | 9 | **753** | **115** | 762 | 118 | **753** | **753** | **753** | **115** | 137.18 | 0 | 0.012 |
| B3 | 202 | 378 | 16 | **1097** | **173** | 1106 | 175 | 1093 | 1085 | 1089.5 | 172 | 193.56 | 0.07 | 0.015 |
| B4 | 244 | 503 | 22 | **1143** | **186** | 1156 | 189 | 1131 | 1118 | 1127.6 | 184 | 299.37 | 0.014 | 0.025 |
| B5 | 302 | 635 | 32 | – | – | 1331 | 203 | 1315 | 1299 | 1307.2 | 198 | 380.41 | – | 0.018 |
| B6 | 356 | 714 | 39 | – | – | 1398 | 215 | 1364 | 1352 | 1359 | 208 | 435.65 | – | 0.028 |
| B7 | 398 | 832 | 47 | – | – | 1462 | 231 | 1431 | 1417 | 1428.5 | 225 | 588.55 | – | 0.023 |
| B8 | 442 | 998 | 59 | – | – | 1548 | 246 | 1507 | 1482 | 1498.4 | 234 | 725.62 | – | 0.032 |
| B9 | 517 | 1064 | 81 | – | – | 1698 | 263 | 1641 | 1605 | 1632.5 | 251 | 882.25 | – | 0.039 |
| B10 | 554 | 1145 | 106 | – | – | 1801 | 278 | 1732 | 1698 | 1718.65 | 264 | 1024.54 | – | 0.046 |

**Table 4**
Simulation results of scenarios with four satellites.

| *SN* | *EN* | *TN* | *CN* | RDS | | PBA | | ACO–LS | | | | | $Gap_R$ | $Gap_P$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $R$ | $F$ | $R^*$ | $F^*$ | $R_{max}$ | $R_{min}$ | $\overline{R}$ | $\overline{F}$ | $\overline{T}$ (s) | | |
| C1 | 92 | 227 | 10 | **556** | **92** | 565 | 94 | **556** | **556** | **556** | **92** | 105.51 | 0 | 0.016 |
| C2 | 179 | 541 | 24 | **1084** | **166** | 1097 | 169 | 1083 | 1079 | 1082.1 | 165 | 193.48 | 0.002 | 0.014 |
| C3 | 285 | 799 | 40 | – | – | 1679 | 265 | 1655 | 1639 | 1648.6 | 258 | 295.62 | – | 0.018 |
| C4 | 381 | 1006 | 53 | – | – | 2199 | 358 | 2129 | 2117 | 2124.5 | 344 | 419.38 | – | 0.034 |
| C5 | 441 | 1267 | 68 | – | – | 2567 | 391 | 2498 | 2475 | 2487.8 | 377 | 557.88 | – | 0.031 |
| C6 | 512 | 1421 | 76 | – | – | 2924 | 443 | 2783 | 2754 | 2772.3 | 421 | 754.40 | – | 0.052 |
| C7 | 603 | 1723 | 95 | – | – | 3404 | 516 | 3194 | 3167 | 3182.5 | 479 | 987.61 | – | 0.065 |
| C8 | 687 | 2064 | 126 | – | – | 3763 | 568 | 3612 | 3574 | 3597.4 | 544 | 1327.28 | – | 0.044 |
| C9 | 752 | 2238 | 157 | – | – | 4171 | 621 | 3984 | 3937 | 3956.2 | 586 | 1735.78 | – | 0.052 |
| C10 | 834 | 2516 | 196 | – | – | 4626 | 659 | 4353 | 4312 | 4338.4 | 612 | 2427.85 | – | 0.062 |

From the obtained results in Tables 3 and 4, we can easily get some interesting observations.

(1) The RDS algorithm is capable of solving relatively small-scale problem (i.e., scenario B1, B2, B3, B4, C1 and C2) within predefined maximum running time, which is restricted within two hours. However, it failed in solving other larger scale problems.
(2) The ACO–LS algorithm is able to find optimal solutions for scenarios B1, B2 and C1 within reasonable computing time. What is more important, ACO–LS can generate high quality results for all other scenarios that are very close to the upper bounds obtained by PBA, indicating ACO–LS's great efficiency.
(3) The running time for ACO–LS solving each SOSP increases approximately linearly with the size of targets and satellites, allowing it to address real-world large size SOSP.
(4) The values of $R_{max}$, $R_{min}$ and $\overline{R}$ are close, validating the robustness of the ACO–LS algorithm.

### 5.3. Comparison with other scheduling algorithms

We also compared the ACO–LS algorithm with the traditional ACO, Tabu Search (TS) [6], genetic algorithm (GA) [10], simulated annealing (SA) [18] and a highest priority first schedule (HPFS) algorithm. For fairness, all the algorithms solve the scheduling scenarios with the consideration of task clustering. Concise descriptions of the comparative algorithms are given below.

**ACO:** Traditional ant colony optimization does not integrate with the local search technique.

**TS:** The Tabu Search algorithm developed by Vasquez and Hao [6] was used to address the daily photograph scheduling of Spot 5. Some modification was implemented to cater to the difference between our scheduling problem and M. Vasquez's: (1) Removed the ternary constraints and related operations; (2) Added two knapsack constraints concerning energy capacity and maximum sensor slewing times.

**GA:** In literature [10], five criterions (i.e. priority, deadline, profit, area and emergency) were considered in the fitness function of GA. We considered only the priority criterion into the fitness function when applied the GA algorithm.

**SA:** We adopted the simulated annealing algorithm incorporating the mutation operator called temperature-dependent swap, which performed best in Globus's tests.

**HPFS**: This algorithm means highest priority first schedule. We rank observation tasks in descendent order with respect to their priorities and each time try to insert the task with the highest priority into the schedule.

The comparison between our algorithm and other algorithms is shown in Fig. 9. It is observed from Fig. 9 that the ACO–LS algorithm outperforms all other algorithms in our tests. In addition, the TS algorithm is superior to ACO, GA, SA and HPFS. The straightforward HPFS algorithm exhibits the lowest efficiency.

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

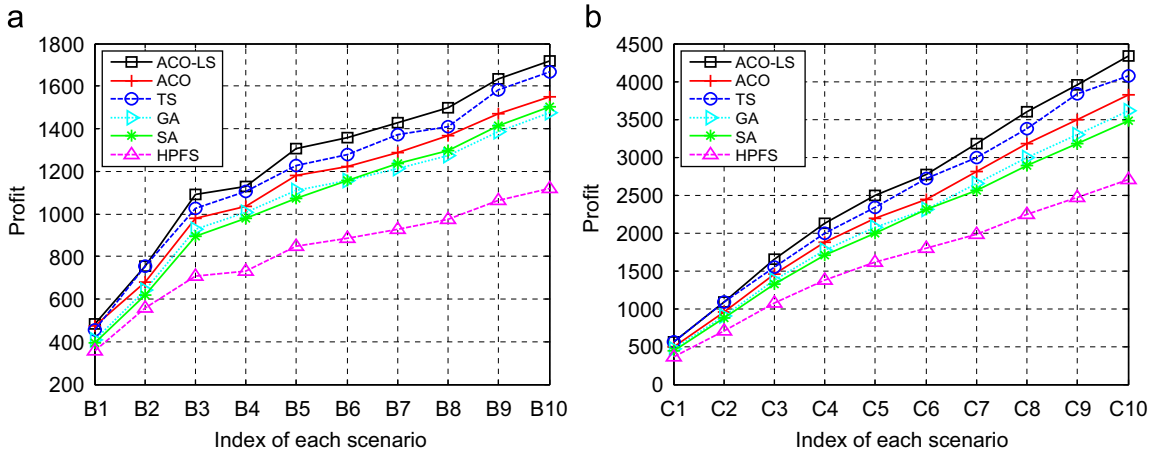


**Fig. 9.** Comparison among different scheduling algorithms.

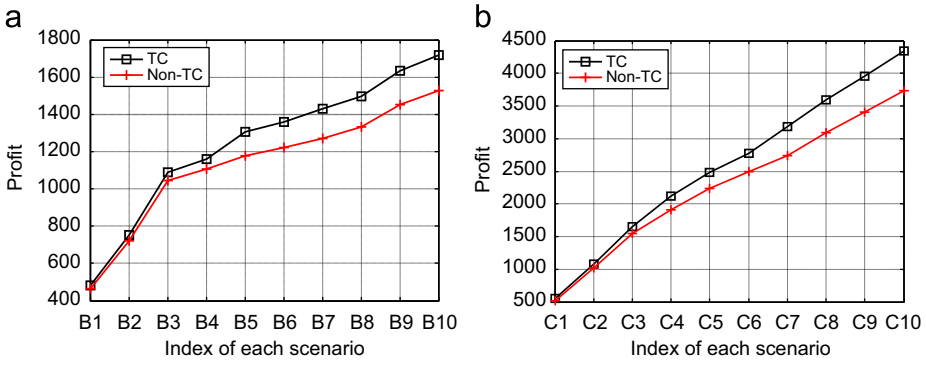


**Fig. 10.** Performance impact of task clustering.

Thus, we can safely come to two conclusions: (1) ACO–LS is a competitive algorithm to solve SOSP; (2) The local search technique noticeably improves the performance of traditional ACO.

### 5.4. The performance impact of task clustering strategy

To evaluate the performance improvement brought by task clustering, we performed experiments to compare the scheduling results obtained by ACO–LS with or without the consideration of task clustering (shortly denoted by TC and Non-TC, respectively).

The comparison is displayed in Fig. 10, from which we discover that task clustering significantly improves the quality of the solution of every scenario. Especially, when the problem size becomes larger, task clustering's advantage appears more obvious. There are two reasons to explain this phenomenon. First, task clustering enables satellites to finish more tasks at the cost of less scarce sensor opening times. Secondly, with the scheduling problem size increasing, conflicts among observing tasks become more severe, causing the reduction of scheduling efficiency. However, more targets also mean more clustering opportunities, which in a certain degree will alleviate the transition time conflict to strengthen the scheduling capability. Therefore, when observing targets densely distribute in a certain area and mutual conflicts become very heavy, considering task clustering into the satellite observation scheduling process is necessary.

### 5.5. The cost-benefit impact of the task clustering strategy

It should be very important to investigate the impact of the task clustering mechanism on cost-benefits. In this study, two kinds of resources (related to the cost) are considered, i.e., memory storage and energy. So we need to analyze the consumption of these two kinds of resources with or without the consideration of the task clustering mechanism, respectively.

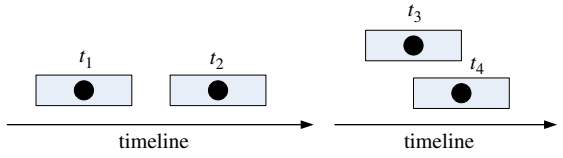


**Fig. 11.** Two cases that influence the resource consumption of satellites.

Intuitively, the impact on resource consumption caused by the task clustering mechanism is case-dependent. That is to say, in some cases, it may cause satellites to consume more resources, while in some other cases it can save resources. For instance, as shown in Fig. 11, we assume that task $t_1$ and $t_2$ are clustered into cluster-task $u$, and task $t_3$ and $t_4$ are clustered into cluster-task $v$.

As for cluster-task $u$, its time-window is $[ts_1,te_2]$. Apparently, in this case, we have

$$[ts_1,te_2] > [ts_1,te_1]+[ts_2,te_2].$$

So, after the task clustering operation, the execution of cluster-task $u$ will consume more memory storage and energy in time interval $[ts_2,te_1]$.

As for cluster-task $v$, its time window is $[ts_3,te_4]$. However, in this case, we have

$$[ts_3,te_4] < [ts_3,te_3]+[ts_4,te_4].$$

Therefore, task clustering will make the execution of cluster-task $v$ save memory storage and energy for $te_3-ts_4$ time unit.

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

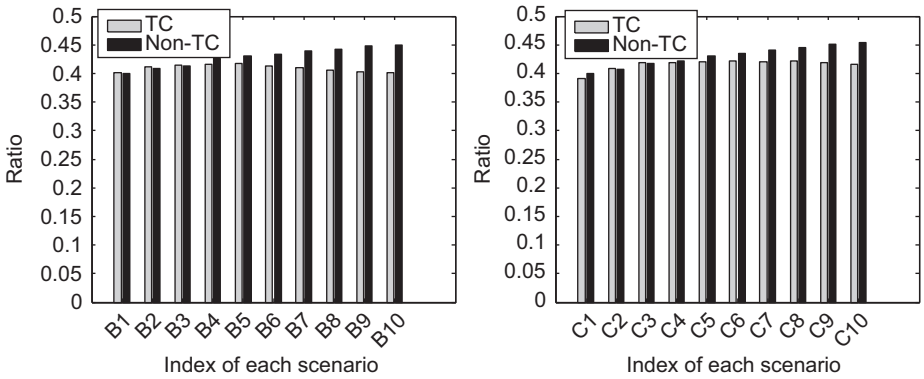


**Fig. 12.** Comparison of the ratio between the obtained profit and the consumed memory storage.

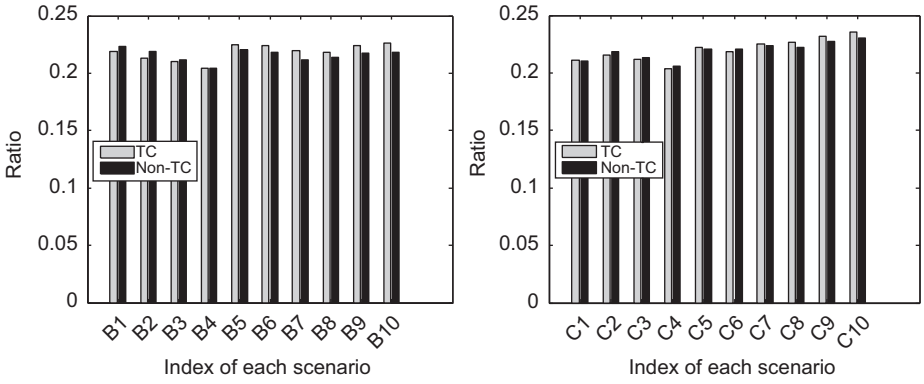


**Fig. 13.** Comparison of the ratio between the obtained profit and the consumed energy.

It should be noted that as interpreted in the introduction, task clustering always means less sensor opening times and smaller sensor slewing angles, which will effectively save energy.

Above analysis tells that the task clustering mechanism's impact on resource consumption varies with different conditions. Therefore, we numerically analyzed the cost-benefit impact of the task clustering mechanism in the experimental tests. Fig. 12 shows the ration between the obtained profit and consumed memory storage with or without the consideration of task clustering (shortly denoted by TC and non-TC as well). From Fig. 12 we find that the ratio of TC is usually higher than that of non-TC (except in scenario B1, B2, B3, C2); and with the number of targets getting larger, the gap between the ratio of TC and non-TC becomes lager as well. This phenomenon demonstrates that to achieve the same profit, the task clustering mechanism will consume more memory storage.

Similarly, Fig. 13 compares the ratio between the obtained profit and the consumed energy with or without the consideration of task clustering. Though there is no apparent regular in Fig. 13, we still can find that the ratio of Non-TC is higher when the number of targets is relatively small (such as in scenario B1, B2, B3, B5, C1, C2, C3). In contrast, the ratio of TC will become higher with the number of targets getting larger. This demonstrates that the task-clustering mechanism has potential in saving energy when solving large scale SOSP.

## 6. Conclusions and future work

Considering task clustering into the scheduling can improve the efficiency of satellite observation systems. Since both task clustering and scheduling process are complicated, we presented a two-phase strategy to address SOSP, i.e., partition the problem solution process into a task clustering phase and a task scheduling phase. In the task clustering phase, a graph model and an improved minimum clique partition algorithm are employed to generate cluster-tasks. In the task scheduling phase, we develop an acyclic directed model and the ACO–LS algorithm to obtain high quality schedules.

Extensive experimental simulations demonstrate that the ACO–LS algorithm is capable of finding optimal or near optimal solutions in reasonable computation cost. Our comparison tests show that ACO–LS outperforms some other algorithms like ACO, TS, GA, SA and HPFS. Moreover, we find that the task clustering technique can significantly improve the scheduling efficiency, especially when the problem scale becomes large. Task clustering is expected to be especially useful when a large number of observing targets densely distribute in a certain area.

The future work in our study is four-fold: (1) Develop a novel strategy that incorporates the task clustering into the scheduling process dynamically. (2) Extend our method to the scheduling of agile satellites. (3) Consider satellite observation and data transmission simultaneously when scheduling satellite activities. (4) Design dynamic scheduling mechanisms to enable a quick response to unexpected situations, such as emergency tasks and cloud disturbances.

## Acknowledgements

This work was supported by the National Basic Research Program of China under Grant no. 97361361 and the National Nature Science Foundation of China under Grant no. 61104180. We are also thankful to the anonymous referees.

Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009

## References


[1] Bianchessi N, Cordeau JF, Desrosiers J, Laporte G, Raymond. V. A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites. European Journal of Operational Research 2007;177(2): 750–62.

[2] 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. Computers & Industrial Engineering 2011;61(2):322–35.

[3] Bensana, E, Verfaillie, G, Agnese, JC, Bataille, N, Blumstein., D Exact and inexact methods for the daily management of an earth observation satellite. In: Proceedings of the international symposium on space mission operations and ground data systems. Munich, Germany; 1996. p. 507–14.

[4] Potter, W, Gasch, J. A photo album of earth: scheduling Landsat 7 mission daily activities. In: Proceedings of the international symposium on space mission operations and ground data systems. Tokyo, Japan; 1998.

[5] Wolfe J, Sorensen. SE. Three scheduling algorithms applied to the earth observing systems domain. Management Science 2000;46(1):148–66.

[6] Vasquez M, Hao JK. A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. Computational Optimization and Applications 2001;20(7):137–57.

[7] Vasquez M, Hao JK. Upper bounds for the SPOT 5 daily photograph scheduling problem. Journal of Combinatorial Optimization 2003;7:87–103.

[8] Gabrel V, Vanderpooten D. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. European Journal of Operational Research 2002;139(3):533–42.

[9] Lin W, Liao D, Liu C, Lee. Y. Daily imaging scheduling of an earth observation satellite. IEEE Transactions on Systems Man and Cybernetics 2005;35(2): 213–23.

[10] Baek S, Han S, Cho K, Lee D, Yang J, Bainum PM, Kim H. Development of a scheduling algorithm and GUI for autonomous satellite missions. Acta Astronautica 2011;68:1396–402.

[11] Miguel GMR, Luiz Antonio FC, Lorena N. Strong formulation for the Spot 5 daily photograph scheduling problem. Journal of Combinatorial Optimization 2010;20:385–98.

[12] Wang, J, Jing, N, Li, J, Chen., Z. A multi-objective imaging scheduling approach for earth observing satellites. In: Proceedings of the 9th annual conference on genetic and evolutionary computation. London, England; 2007. p. 2211–8.

[13] Frank, J, Jónsson, A, Morris, R, Smith, DE.. Planning and scheduling for fleets of earth observing satellites. In: Proceedings of the sixth international symposium on artificial intelligence, robotics, automation and space. Montreal, Canada; 2000.

[14] Lemaître, M, Verfaillie, G, Jouhaud, F, Lachiver, JM, Bataille, N. How to manage the new generation of agile earth observation satellites. In: Proceedings of the 6th international conference on space operations. Toulouse, France; 2000.

[15] Lemaître M, Verfaillie G, Jouhaud F, Lachiver JM, Bataille N. Selecting and scheduling observations of agile satellites. Aerospace Science and Technology 2002;6:367–81.

[16] Beaumet G, Verfaillie G, Charmeau MC. Feasibility of autonomous decision making on board an agile earth-observing satellite. IEEE Computational Intelligence Magazine 2011;27(1):123–39.

[17] Habet D, Vasquez M, Vimont Y. Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite. Computational Optimization and Applications 2010;47(2):307–33.

[18] Globus, A, Crawford, J, Lohn, J, Pryor, A. A comparison of techniques for scheduling fleets of earth-observing. In: Proceedings of the national conference on artificial intelligence. San Jose, CA, United states; 2004. p. 836–43.

[19] Mansour MAA, Dessouky. MM. A genetic algorithm approach for solving the daily photograph selection problem of the SPOT 5 satellite. Computers & Industrial Engineering 2010;58(3):509–20.

[20] Bianchessi N, Righini G. Planning and scheduling algorithms for the COSMO-SkyMed constellation. Aerospace Science and Technology 2008;12(7): 535–44.

[21] Liao D, Yang Y. Imaging order scheduling of an earth observation satellite. IEEE Transactions on Systems Man and Cybernetics Part B 2007;37(5): 794–802.

[22] Harrison, SA, Price, ME, Philpott, MS.. Task scheduling for satellite based imagery. In: Proceedings of the 18th Workshop of the UK planning and scheduling special interest group; University of Sanford, UK; 1999.

[23] Gabrel V, Murat C. Mathematical programming for earth observation satellite mission planning. In: Ciriani T, Fasano G, Gliozzi S, Tadei R, editors. Operations research in space and air. Boston: Kluwer Academic Publishers; 2003. p. 103–22.

[24] Muraoka, H, Cohen, R, Ohno, T, Doi, N.. ASTER observation scheduling algorithm. In: Proceedings of the international symposium on space mission operations and ground data systems. Tokyo, Japan; 1998.

[25] Wu G, Ma M, Zhu J, Qiu D. Multi-satellite observation integrated scheduling method oriented to emergency tasks and common tasks. Journal of Systems Engineering and Electronics 2012;23(5):723–33.

[26] Garey M, Johnson D. Computers and Intractability: a guide to the theory of NP-completeness Freeman. San Francisco, CA 1978.

[27] Tseng CJ, Siewiorek DP. Automated synthesis of data paths in digital systems. IEEE Transactions on Computer-Aided Design 1986;5(3):379–95.

[28] Kim JT, Shin DR. New efficient clique partitioning algorithms for register-transfer synthesis of data paths. Journal of the Korean Physical Society 2002;40(4):754–8.

[29] Verfaillie, G, Lemaître, M. Tutorial on planning activities for earth watching and observation satellites and constellations: from off-line ground planning to on-line on-board planning. In: Proceedings of the international conference on automated planning and scheduling. English Lake District, UK; 2006.

[30] Dorigo, M. Optimization, learning and natural algorithms. Politecnico diMilano, Italy; 1992.

[31] Dorigo M, Stützle T. Ant colony optimization. London, Cambridge: The MIT Press; 2004.

[32] Stützle T, Hoos. HH. MAX–MIN ant system. Future Generation Computer Systems 2000;16(8):889–914.

[33] Fuellerer G, Doerner KF, Hartl RF, et al. Ant colony optimization for the two-dimensional loading vehicle routing problem. Computers & Operations Research 2009;36(3):655–73.

[34] Rajendran C, Ziegler H. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research 2004;155(2):426–38.

[35] Huang K, Liao C. Ant colony optimization combined with taboo search for the job shop scheduling problem. Computers & Operations Research 2008;35:1030–46.

[36] Verfaillie, G, Lemaître, M, Schiex, T. Russian doll search for solving constraint optimization problems. In: Proceedings of the national conference on artificial intelligence; 1996. p. 181–7.


Please cite this article as: Wu G, et al. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.02.009