Exact and inexact scheduling algorithms for multiple earth observation satellites under uncertainties of clouds

Wang J, Demeulemeester E, Qiu D, Liu J.

KBI_1514

Exact and inexact scheduling algorithms for multiple earth observation satellites under uncertainties of clouds

Jianjiang Wang[a,b], Erik Demeulemeester[b], Dishan Qiu[a], Jin Liu[a]

[a] *Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China*

[b] *Research Center for Operations Management, Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Belgium*

Email: jianjiangwang.nudt@gmail.com, erik.demeulemeester@kuleuven.be, ds_qiu@sina.com, liujin229234@163.com

Abstract

Most earth observation satellites (EOSs) are equipped with optical sensors, which cannot see through clouds. Hence, many observations will be useless due to the presence of clouds. In this work, in order to improve the possibility of completing the tasks under uncertainties of clouds, we take the scheduling of each task to multiple resources into account and establish a novel non-linear mathematical model. To solve the problem efficiently under different scenarios, we propose an exact algorithm and some heuristic algorithms. With respect to the exact algorithm, we divide the complicated problem into a master problem and multiple subproblems, with a subproblem for each resource. A labeling-based dynamic programming algorithm is proposed to solve each subproblem. Afterwards, based on the solutions of the subproblems, we develop an enumeration algorithm to solve the master problem. Furthermore, we design five heuristics to solve the large-scale problems that generally fail to be solved by the exact algorithm due to the large space complexity. Experimental results show that the solutions of our model perform better than those of previous studies, and we also reveal the strengths and weaknesses of the proposed algorithms while solving different size instances.

Keywords: earth observation satellites, uncertainties of clouds, model decomposition, enumeration, dynamic programming, heuristic

1. Introduction

Earth Observation Satellite (EOS) scheduling means to allocate the tasks submitted by users to satellites, making the schedule satisfy the operational constraints. Because of some special advantages, e.g. an expansive coverage area, long-term surveillance, a high frequency of repeated observations, accurate and effective information access and unlimited airspace borders, EOSs have been extensively employed in earth resources exploration, nature disaster surveillance, urban planning, crop monitoring, etc. Due to the explosively increased applications, the number of satellites, in spite of being increased quickly, is still too limited. Hence, it is nontrivial for EOS scheduling to get high observation effectiveness and efficiency.

Up to now, a great number of studies focusing on EOS scheduling have been proposed, in which EOS scheduling was formulated and solved in different ways:

Mathematical programming: Without considering memory and energy constraints, Benoist et al. [4], Habet et al. [19, 20, 21] and Lemaître et al. [25] formulated the problem with mathematical programming models. Liao et al. [28, 29], Lin et al. [30, 31, 32, 33] and Marinelli et al. [34] proposed the time-indexed formulations of EOS scheduling, and established mixed integer programming models. In addition, mixed integer programming models were also constructed on the basis of a "flow variable" formulation [8, 9, 15, 16]. Hall et al. [22] formulated the problem as a longest path problem with time windows, and suggested an integer linear programming model.

Constraint satisfaction problem: Lemaître et al. [24] and Verfaillie et al. [43] formulated EOS scheduling as constraint satisfaction problems. Agnèse et al. [1], Bensana et al. [5] and Verfaillie et al. [44] proposed valued constraint satisfaction problem (VCSP) formulations for SPOT-5 satellite scheduling, without considering energy constraints.

Knapsack problem: Vasquez et al. [41, 42] and Wolfe et al. [49] formulated EOS scheduling as 0-1 knapsack problems.

Graph-based formulation: Gabrel et al. [13, 14] adopted a directed acyclic graph (DAG) model to describe the satellite scheduling problem. Besides, Sarkheyli et al. [37] and Zufferey et al. [52] modeled EOS scheduling as graph coloring problems.

Besides, Frank et al. [12] and Pralet et al. [35] adopted the Constraint-Base Interval (CBI) language to describe the problem.

In addition, the solution approaches for EOS scheduling can be classified into the following categories.

Exact algorithms: Agnèse et al. [1] and Bensana et al. [5] proposed depth-first branch and bound algorithms for SPOT-5 satellite scheduling. Also, Benoist et al. [4], Bensana et al. [5] and Verfaillie et al. [44] suggested Russian Doll search algorithms, which are based on branch and bound but replace one search by n successive searches on nested subproblems, using the results of each search when solving larger subproblems, in order to improve the lower bound on the global valuation of any partial assignment. Besides, Gabrel et al. [14], Hall et al. [22] and Lemaître et al. [25] developed dynamic programming methods to get the optimal solutions of EOS scheduling problems.

Metaheuristics: A large number of metaheuristics were proposed for EOS scheduling, which primarily contain tabu search algorithms [5, 9, 11, 31, 33, 41, 51, 52], genetic algorithms [2, 23, 26, 38, 39, 49, 51], ant colony algorithms [27, 45, 50], local search algorithms [24, 25, 43] and simulated annealing algorithms [17, 18, 51].

Heuristics: Agnèse et al. [1], Bensana et al. [5] and Lemaître et al. [25] proposed greedy algorithms to get feasible solutions for EOS scheduling problems. On the basis of heuristic rules, Bianchessi et al. [7, 10], Hall et al. [22], Wang et al. [46, 47, 48] and Wolfe et al. [49] developed constructive algorithms that can solve the problem efficiently, without guaranteeing the optimality of the solutions. Bianchessi et al. [8], Lin et al. [30, 33] and Marinelli et al. [34] adopted lagrangian relaxation heuristics to solve the problems, obtaining close-to-optimal solutions.

Many observations performed by EOSs are lost due to the presence of clouds, because most EOSs are equipped with optical sensors that cannot see through clouds [17, 18]. For example, around 80% of the observations with the currently operational optical SPOT satellites are useless due to the presence of clouds [3]. Hence, clouds are a nontrivial issue for EOS scheduling, which cannot be ignored. Unfortunately, to the best of our knowledge, among all the previous studies, only a few have considered the impact of clouds. Lin et al. [30, 31, 32, 33] formulated the presence of clouds as a set of covered time windows, and forbade the tasks to be observed in the covered time windows of scheduling. In practice, the drawback and infeasibility of Lin's approach is that there exist a lot of uncertainties of clouds, which are always changing over time [2, 6, 24] and it is impossible to be forecasted exactly, so decision makers cannot get the deterministic information of clouds before scheduling. Liao et al. [28, 29] considered the uncertainties of clouds, formulated the presence of clouds for each observation window as a stochastic event, and established a model with the objective of maximizing the weighted sum of a function of the profits and the expected number of executed tasks.

In deterministic scheduling, a task will be completed successfully once it has been scheduled. Hence, it is sufficient to be scheduled once to a resource for each task, and there is no difference between scheduling a task to one resource or to multiple resources. In contrast, in the scheduling under uncertainties, a scheduled task can fail to be completed at a certain probability. Therefore, if we allocate a task to multiple resources, the task will have a higher probability to be completed and the scheduling system will be more robust. Unfortunately, in the previous studies of uncertain EOS scheduling, the characteristics of deterministic scheduling are simply and inadequately migrated, and each task will be restricted to be scheduled to only one resource.

In this paper, in order to make more tasks be completed under uncertainties of clouds, we take into account the scheduling of a task to multiple resources for one time and construct a corresponding expectation model. Due to the complexities of the problem, the proposed model is neither linear nor quadratic, which brings too many challenges for solving. Both exact and inexact algorithms are proposed to find optimal and good feasible solutions, respectively. For the problems of small size, we divide the complicated problem into a master problem and multiple subproblems. For each subproblem, we obtain all feasible solutions with a labeling-based dynamic programming algorithm. Moreover, for the master problem, based on the solutions of the subproblems, all feasible solutions,

Table 1: Notations

T	set of tasks, $T = \{1, ..., n\}$
i, j	task index, $i, j \in T \cup \{\text{s,t}\}$, in which s, t are dummy tasks
ϖ_i	profit of task i, $i \in T$
O	set of orbits, $O = \{1, ..., m\}$
k	orbit index, $k \in O$
b_{ik}	$b_{ik} = 1$ if orbit k is available for the observation of task i, otherwise $b_{ik} = 0$, $i \in T, k \in O$
M_k, E_k	memory capacity and energy capacity of orbit k, $k \in O$
m_k, e_k	memory and energy consumption for each unit time of observation of orbit k, $k \in O$
$[ws_{ik}, we_{ik}]$	time window of observation of task i on orbit k, $i \in T, k \in O$
θ_{ik}	slewing angle of observation of task i on orbit k, $i \in T, k \in O$
st_{ij}^k	setup time between task i and task j on orbit k, $i, j \in T, k \in O$
ρ_{ij}^k	energy consumption for slewing between task i and task j on orbit k, $i, j \in T, k \in O$
$\tilde{\lambda}_{ik}$	binary stochastic variable, $\tilde{\lambda}_{ik} = 1$ denotes that task i can be successfully observed on orbit k, otherwise $\tilde{\lambda}_{ik} = 0$, $i \in T, k \in O$
p_{ik}	probability that task i will be successfully observed on orbit k, $i \in T, k \in O$

which are the combinations of solutions for subproblems, are enumerated and the optimal solution is selected. In addition, considering the huge space complexity of the exact algorithm and the limit of computer memory, the exact algorithm based on enumeration is only successful in solving small to medium size problems. Hence, we design a number of efficient heuristic algorithms to solve the large-scale problems, to get the feasible solutions that are close-to-optimal efficiently. Afterwards, a large number of experiments by simulation are conducted to verify that the solutions of our model perform better than those of previous studies under uncertainties of clouds. Besides, the feasibility of both the exact algorithm and the heuristics is verified, coupled with an evaluation of the qualities and the performances of solving.

The paper is organized as follows. In the next section, we describe the problem and formulate the problem with a novel mathematical model. Subsequently, Section 3 divides the problem into a master and some subproblems, and proposes an enumeration algorithm and a dynamic programming algorithm for solving. Section 4 suggests a number of efficient heuristic algorithms for the large-scale problems. Numerical results of our approach are presented in Section 5. The last section offers conclusions and directions for future research.

2. The uncertain EOS Scheduling Problem

In this study we focus on the scheduling of multiple EOSs under uncertainties of clouds, in which the presence of clouds for observations is formulated as stochastic events. Essentially, the problem is a stochastic programming problem, and we construct a novel mathematical model for this problem.

2.1. Problem description

In EOS scheduling, users normally submit two types of requests: (1) a target, i.e., a circle with limited dimension, or (2) a polygon which may cover a wide geographical area. Due to its large size, a polygon usually is failed to be observed in a single orbit and therefore partitioned into multiple strips [9, 11, 46]. In order to facilitate the description, a target can be seen as a single strip. Hence, the tasks in this work are corresponding to the strips that require being observed. Furthermore, the profit associated with each strip is defined as a piecewise linear convex function of the surface of the polygon that is acquired, which is illustrated in Figure 1.

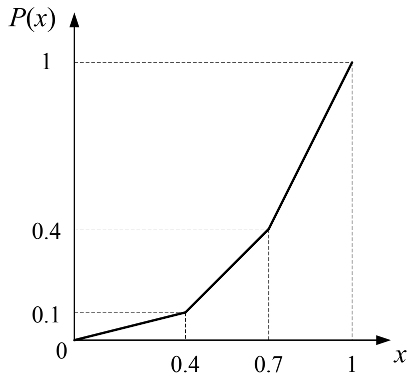

Figure 1: Non-linear profit function $P(x)$ associated with the acquired portion x of a polygon [9, 11, 40]

In the previous studies, researchers usually formulate the satellites as the resources, and a task will have at most one observation window on each resource. However, if the scheduling horizon is long enough, a satellite will orbit the earth for multiple orbits and pass over a strip for multiple times. Hence, the observation windows for a task on each satellite will not be unique [46, 47], which makes the problem difficult for modeling and solving. To handle the difficulties, we formulate the orbits of the satellites as the resources. Hence, there will be at most one observation window for each task on each resource, regardless of the length of the scheduling horizon.

Moreover, we assume that there will be an opportunity for data downloading for each orbit, and the data will be all downloaded resulting in the memory becoming empty again. Hence, the memory capacity for each orbit is equal to the fixed capacity of the satellite's on-board memory. Besides, satellites collect solar energy by solar panels from the sun and store energy in batteries as they orbit the earth. Because both the period and the rate of collecting energy can be approximately considered as constants, and the maximum charge capacity is fixed, we assume the available energy for each orbit to be a constant. Hence, both the memory and the energy capacities for each orbit can be formulated as constants in this study.

Some notations of this study are summarized in Table 1. Let T be the set of tasks (strips) submitted by users and let O be the set of orbits within the scheduling horizon. With each task $i \in T$ is associated a profit ϖ_i. Each orbit $k \in O$ is associated with a memory capacity M_k, an energy capacity E_k, a memory consumption for each unit of observation time m_k and an energy consumption for each unit of observation time e_k. Let $b_{ik} = 1$ denote that task i can be observed on orbit k, otherwise $b_{ik} = 0$. $[ws_{ik}, we_{ik}]$ denotes the time window for task i on orbit k, and θ_{ik} denotes the slewing angle. Many of these notions are illustrated in Figure 2. In this work, we only consider non-agile satellites, which have the maneuverability of rolling (slewing) which indicates a movement that is perpendicular to the direction of the orbit, without the maneuverability of pitching which indicates a movement along the direction of the orbit. Hence, the time windows for observations are fixed without flexibility, such that the start and finish time of task i on orbit k will be fixed as $[ws_{ik}, we_{ik}]$, and the duration will be $we_{ik} - ws_{ik}$.

After observing a task, the satellite requires a sequence of transformation operations to observe the next one, which are sensor shutdown$\rightarrow$ slewing$\rightarrow$ attitude stability$\rightarrow$ startup. Hence, there should be sufficient setup time between two consecutive tasks, and the required setup time can be calculated by the following formula:

$$st_{ij}^k = sd_k + |\theta_{ik} - \theta_{jk}|/s_k + as_k + su_k,$$

where st_{ij}^k is the setup time between task i and task j on orbit k, and sd_k, as_k and su_k are the times of sensor shutdown, attitude stability and startup on orbit k, respectively. Besides, s_k is the slewing velocity of orbit k, and θ_{ik} and θ_{jk} are the slewing angles of tasks i and j on orbit k, respectively.

For observing task i on orbit k, the memory consumption can be computed by $(we_{ik} - ws_{ik})m_k$. Differently from memory, energy will not only be consumed by observation, but also by sensor

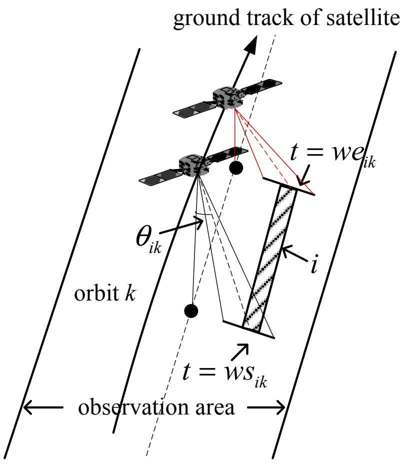

Figure 2: Time window for observation

slewing. The energy consumption for observing task i on orbit k is $(we_{ik} - ws_{ik})e_k$. Let ρ_{ij}^k denote the energy consumption of slewing between consecutive tasks i and j on orbit k, which can be calculated by the formula below:

$$\rho_{ij}^k = |\theta_{ik} - \theta_{jk}|\pi_k,$$

where π_k is the energy consumption for each unit slewing angle on orbit k. Besides, it should be noted that the parameters for each orbit, such as the times of sensor shutdown, attitude stability and startup, memory and energy consumption per unit observation time, and energy consumption per unit slewing angle, are inherited from the satellite to which the orbit belongs.

Considering the uncertainties of clouds, we formulate the presence of clouds for observations as stochastic events, denoted by 0-1 stochastic variables $\tilde{\lambda}_{ik}, i \in T, k \in O$. $\tilde{\lambda}_{ik} = 1$ if the observation of task i on orbit k can be successfully observed without the presence of clouds, otherwise $\tilde{\lambda}_{ik} = 0$. Let p_{ik} denote the probability for a successful observation of task i on orbit k, i.e., no presence of clouds, thus we can obtain $p\{\tilde{\lambda}_{ik} = 1\} = p_{ik}$ and $p\{\tilde{\lambda}_{ik} = 0\} = 1 - p_{ik}$. Using this way, the changes of clouds are implicitly formulated by the different uncertain impacts of clouds on different time windows, which will be forecasted prior to scheduling.

2.2. Mathematical model

In this study, we formulate the EOS scheduling problem with flow variables, and establish a non-linear mathematical model. In this model, we use binary decision variables $x_{ij}^k \in \{0,1\}$ $(i,j \in T \cup \{s,t\}, k \in O)$, in which $T = \{1,...,n\}$ is the set of real tasks and s,t are dummy tasks for starting and terminating, respectively. $x_{ij}^k = 1$ if both tasks i,j are scheduled on orbit k, and task i is the immediate predecessor of task j; otherwise $x_{ij}^k = 0$. The mathematical model is given below:

$$max \sum_{i \in T} \varpi_i \cdot \{1 - \prod_{k \in O} (1 - p_{ik} \cdot \sum_{\substack{j \in T \cup \{t\} \\ j \neq i}} x_{ij}^k)\} \tag{1}$$

subject to

$$\sum_{\substack{j \in T \cup \{t\} \\ j \neq i}} x_{ij}^k = \sum_{\substack{j \in T \cup \{s\} \\ j \neq i}} x_{ji}^k, \ \forall i \in T, k \in O \tag{2}$$

$$\sum_{\substack{j \in T \cup \{t\} \\ j \neq i}} x_{ij}^k \leq b_{ik}, \forall i \in T, k \in O \tag{3}$$

$$x_{ij}^k = 0, \forall i,j \in T, k \in O, \text{if } we_{ik} + st_{ij}^k > ws_{jk} \tag{4}$$

$$\sum_{i\in T}\sum_{\substack{j \in T\cup\{t\}\\ j\neq i}} x_{ij}^k (we_{ik} - ws_{ik}) m_k \leq M_k, \forall k \in O \tag{5}$$

$$\begin{aligned}\sum_{i\in T}\sum_{\substack{j \in T\cup\{t\}\\ j\neq i}} & x_{ij}^k (we_{ik} - ws_{ik}) e_k \\ & + \sum_{i\in T}\sum_{\substack{j \in T\\ j\neq i}} x_{ij}^k \rho_{ij}^k \leq E_k, \forall k \in O\end{aligned} \tag{6}$$

$$x_{ij}^k \in \{0,1\}, \forall i,j \in T\cup\{s,t\}, k \in O \tag{7}$$

The objective (1) is to maximize the expectation value of the profits of the executed tasks under uncertainties of clouds, in which $1-\prod_{k\in O}(1-p_{ik}\cdot\sum_{\substack{j \in T\cup\{t\}\\ j\neq i}} x_{ij}^k)$ denotes the completion probability for task i when it is scheduled to multiple orbits. The set of constraints (2) that are flow balance constraints ensure that the number of predecessors is equal to the number of successors for each task. Constraints (3) enforce that each task can only be scheduled to the orbits that are available for it. There must be sufficient setup times between consecutive tasks for transformations, which is enforced in constraints (4). Note that the setup time constraints are much stronger than the common non-overlapping constraints. The setup time constraints do not only forbid the overlapping of tasks, but also require sufficient setup times between consecutive tasks for some transformation operations. Constraints (5) check that the memory consumption of the scheduled tasks cannot exceed the memory capacity for each orbit. Constraints (6) compute the energy consumption of the task sequence for each orbit, and enforce that the energy consumption must be less than or equal to the capacity.

Note that constraints (3) and (4) just fix some variables in some cases. Hence, we will fix the relevant variables prior to solving and modify the constraints to make the model brief. With respect to constraints (3), if $\exists i \in T, k \in O, b_{ik} = 0$, we can fix the corresponding variables beforehand: $x_{ij}^k = 0, \forall j \in T\cup\{t\}$. Hence, constraints (3) only have an effect when $b_{ik} = 1, \forall i \in T, k \in O$, and we modify constraints (3) as follows:

$$\sum_{\substack{j \in T\cup\{t\}\\ j\neq i}} x_{ij}^k \leq 1, \forall i \in T, k \in O, b_{ik} = 1. \tag{8}$$

Besides, with respect to constraints (4), we can directly fix the variables and remove the constraints prior to solving. Therefore, the simplified model can be formulated as: maximize (1) subject to (2), (5)-(8).

3. An exact algorithm

It has been described in the previous section that the proposed model in this study is neither linear nor quadratic, which brings too many challenges for its solution. To the best of our knowledge, currently there is no existing algorithm that can solve this model. In this paper, we reformulate the problem as a master problem for selection and some subproblems for path planning. For the master problem, we propose an enumeration algorithm, and we solve the subproblems with a dynamic programming algorithm.

3.1. Master problem and enumeration

From the analysis of the model, we can conclude that only the objective is coupled and difficult to solve, and all constraints can be straightforwardly decomposed by orbits. Hence, the constraints are structured with diagonal blocks. With the model decomposition, we can reformulate the problem as a master problem with the objective (1) and some subproblems with the constraints (2), (5)-(8). Hence, the subproblem is to find all the feasible maximal solutions satisfying the constraints (2), (5)-(8). The solution of the master problem is the combination of the feasible solutions for all subproblems.

To facilitate the reformulation, we define the following notations:

R_k: the set of all feasible solutions for orbit k, $k \in O$.

r: the index of feasible solution, $r \in R_k$, $k \in O$.

α_{ijrk}: $\alpha_{ijrk} = 1$ if task i is the immediate predecessor of task j of solution r on orbit k, otherwise $\alpha_{ijrk} = 0$.

z_{kr}: decision variables, $z_{kr} = 1$ if feasible solution r is selected for orbit k, $r \in R_k$, $k \in O$, otherwise $z_{kr} = 0$.

The formulation of the master problem is described below:

$$max \sum_{i \in T} \varpi_i \cdot \{1 - \prod_{k \in O} (1 - p_{ik} \cdot \sum_{r \in R_k} \sum_{\substack{j \in T \cup \{t\} \\ j \neq i}} z_{kr} \cdot \alpha_{ijrk})\} \tag{9}$$

subject to

$$\sum_{r \in R_k} z_{kr} = 1, \forall k \in O \tag{10}$$

$$z_{kr} \in \{0, 1\}, \forall r \in R_k, k \in O \tag{11}$$

The objective (9) is to maximize the expectation value of the profits of the executed tasks. Constraints (10) represent that one and only one solution should be selected for each orbit.

Unfortunately, the master problem above is still neither linear nor quadratic, and thus cannot be solved by existing solvers or algorithms. However, the master problem has been simplified in comparison with the original model. In this study, using dynamic programming, we can solve the subproblem for each orbit and get all the feasible maximal solutions. Choosing a feasible solution for each orbit, we can make a combination that is corresponding to a feasible solution of the original problem, and then all feasible solutions of the original problem are obtained by enumerating all combinations. Finally, we will select the optimal solution that is the feasible solution with maximum objective function value.

3.2. Subproblem and dynamic programming

As described above, the subproblem for each orbit is to search for all feasible solutions, which can be solved with a dynamic programming algorithm. In order to facilitate the description, we firstly define a directed acyclic graph $G^k = (V^k, A^k)$ for each orbit k, $k \in O$. In the remainder we drop the superscript when this does not provoke ambiguity. With the task-on-node representation the nodes in V represent the tasks that are available for the orbit, plus two special nodes $\{s, t\}$ representing the dummy starting and dummy terminating tasks. Each node in V represents an available task and all nodes are numbered following the chronological order of the corresponding time windows. A is the set of arcs, which is defined as follows:

- $\forall j \in V \cup \{t\}$, $(s, j) \in A$;
- $\forall j \in V \cup \{s\}$, $(j, t) \in A$;
- $\forall i, j \in V$, $(i, j) \in A$ iff task j can be observed after task i, i.e. the setup time constraints between tasks i and j are satisfied.

In this formulation, it is obvious that a path from the starting node s to the final node t represents a feasible solution for the subproblem. Hence, for each orbit, the subproblem is regarded as a path planning problem, which is to search for all feasible paths from s to t.

The subproblem for each orbit can be solved by a labeling-based dynamic programming algorithm. For each node j, we store a set of labels $P(j)$ that represent all paths from the starting node s to j. A label in $P(j)$, corresponding to a path $p = \{s, i_1, \ldots, i_k, j\}$, is denoted by $l_p = [i_k, \uparrow l_{p'}, M'_p, E'_p]$, where i_k is the immediate predecessor of j in path p, $\uparrow l_{p'}$ is a pointer on the label referring to the path $p' = \{s, i_1, \ldots, i_k\}$ in $P(i_k)$, and M'_p, E'_p are the memory consumption and energy consumption of path p, respectively. The labeling-based dynamic programming algorithm is described in **Algorithm 1**, in which $\Gamma^{-1}(j)$ is the set of all predecessors of j. At each iteration, we determine the label set for a node j, $j \leftarrow 1, \ldots, n, t$ (remember here that t represents the dummy terminating task). For each subpath in each predecessor i of j, we add the subpath with the node j and the relevant arc (i, j). If both the memory and energy constraints are satisfied, we will add the label of the path to $P(j)$, and then access the next subpath.

Algorithm 1 Labeling-based Dynamic Programming

```
1: Initialization:
2: $P(s) = \{[null, null, 0, 0]\}$
3: $P(j) = \emptyset \ (j = 1, \ldots, n, t)$
4: Solving:
5: for $j \leftarrow 1, \ldots, n, t$ do
6:     for all $i \in \Gamma^{-1}(j)$ do
7:         for all $l_p \in P(i)$ do
8:             if memory and energy constraints are satisfied then
9:                 $P(j) \leftarrow P(j) \cup \{[i, \uparrow l_p, M'_p + m(we_j - ws_j), E'_p + e(we_j - ws_j) + \rho_{ij}]\}$
10:            end if
11:        end for
12:    end for
13: end for
```

Table 2: A toy EOS scheduling instance

Task no.	Profit	Orbit no.					
		1		2		3	
		Window	Probability	Window	Probability	Window	Probability
1	7	[42,48]	0.86	-	-	[66,72]	0.75
2	9	-	-	[16,20]	0.92	-	-
3	3	[18,25]	0.72	[42,49]	0.87	-	-
4	6	[50,55]	0.96	[66,71]	0.89	-	-
5	6	-	-	-	-	[32,38]	0.86
6	7	[22,27]	0.91	[41,46]	0.87	[57,62]	0.82
7	9	-	-	[27,34]	0.78	[42,47]	0.91
8	8	[32,38]	0.92	[53,59]	0.64	[18,24]	0.90
Memory capacity		60		40		50	
Energy capacity		80		50		70	
Memory consumption for each unit time		2.5		2		2.5	
Energy consumption for each unit time		1.5		2		2	

By this algorithm, for each node j, all paths from the starting node s to j will be stored in the label set $P(j)$. In addition, all feasible paths from s to t, which are corresponding to the feasible solutions of the subproblem, are stored in the dummy terminating node t.

Definition 1. Path domination

For two feasible paths p and q, path p is dominated by path q if $V(p) \subset V(q)$, in which $V(p), V(q)$ are the set of nodes in paths p and q, respectively.

Definition 2. Dominated path and non-dominated path

Path p is a dominated path if $\exists q$ such that path p is dominated by path q, otherwise path p is a non-dominated path.

Subsequently, to obtain all the feasible maximal solutions for each subproblem, we need to remove all the dominated paths from s to t, and then all the non-dominated paths are corresponding to all the feasible maximal solutions.

3.3. Example

To describe the proposed model and algorithm more clearly, let us introduce a toy instance of EOS scheduling. The instance is composed of 8 non-dummy tasks and 3 orbits, and Table 2 outlines the following data: profits of tasks, availabilities for observations, time windows, probabilities of successful executions, memory and energy capacities, as well as memory and energy consumptions for each unit observation time. The symbol "-" denotes that the orbit is not available for observing the task.

Then the setup times and slewing energy are illustrated in Figures 3-5, in which the normal numbers above the lines denote the setup times and the italic numbers below the lines denote the slewing energy. In addition, the dashed lines represent the infeasible paths that do not satisfy the setup time constraints.

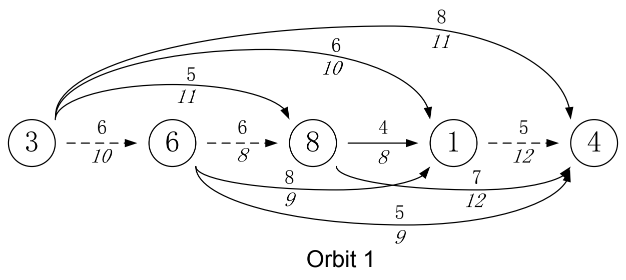

Figure 3: Setup times and slewing energy for orbit 1

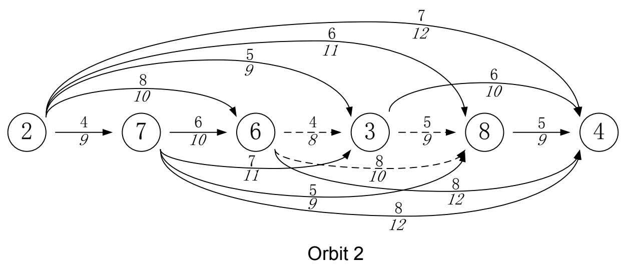

Figure 4: Setup times and slewing energy for orbit 2

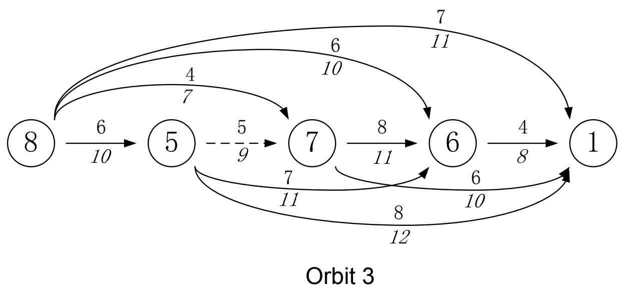

Figure 5: Setup times and slewing energy for orbit 3

With the labeling-based dynamic programming algorithm, we can get all non-dominated feasible paths for each orbit, which are listed as below:

Orbit 1:

$$s \to 3 \to 8 \to 1 \to t$$
$$s \to 3 \to 8 \to 4 \to t$$
$$s \to 6 \to 1 \to t$$
$$s \to 6 \to 4 \to t$$

Orbit 2:

$$s \to 2 \to 7 \to 6 \to t$$
$$s \to 2 \to 7 \to 4 \to t$$
$$s \to 2 \to 7 \to 8 \to t$$
$$s \to 2 \to 6 \to 4 \to t$$

$$s \to 2 \to 8 \to 4 \to t$$
$$s \to 7 \to 6 \to 4 \to t$$
$$s \to 2 \to 7 \to 3 \to t$$
$$s \to 7 \to 8 \to 4 \to t$$
$$s \to 2 \to 3 \to 4 \to t$$
$$s \to 7 \to 3 \to 4 \to t$$

Orbit 3:

$$s \to 8 \to 7 \to 6 \to t$$
$$s \to 8 \to 7 \to 1 \to t$$
$$s \to 7 \to 6 \to 1 \to t$$
$$s \to 8 \to 6 \to 1 \to t$$
$$s \to 8 \to 5 \to 6 \to t$$
$$s \to 8 \to 5 \to 1 \to t$$
$$s \to 5 \to 6 \to 1 \to t$$

Apparently, there are 280 ($4 \times 10 \times 7$) combinations of paths for the 3 orbits in this problem, which means 280 feasible solutions in all. After enumeration and calculating the corresponding objective function values, we can get that the following solution will yield the maximum objective value, which is the optimal solution.

$$\text{Orbit 1: } s \to 3 \to 8 \to 4 \to t$$
$$\text{Orbit 2: } s \to 2 \to 7 \to 6 \to t$$
$$\text{Orbit 3: } s \to 5 \to 6 \to 1 \to t$$

It is obvious that in the optimal solution task 6 has been scheduled to two orbits (Orbit 2 & Orbit 3), which increases the probability of completing task 6.

4. Inexact algorithms

As described in the previous text, the proposed exact algorithm will fail to handle large-scale problems that are more practical, because of its large space complexity. Therefore, in this section, we design five heuristic procedures for solving the large-scale problems efficiently. In contrast with the exact algorithm, the heuristics normally can only get feasible solutions without guaranteeing the optimality. It must be emphasized that all the subsequent heuristics are multi-pass approaches that apply multiple passes to obtain multiple unique feasible solutions among which the best one is chosen. As such, sampling is essential. Hence, above all, we briefly introduce some sampling methods.

4.1. Sampling

In the case of adopting sampling, the selection of the priority rules is made using a bias scheme. The basic principle is to assign a probability of being selected to each activity (task or column for this study). Normally, there are three basic sampling methods:

- *Random sampling* assigns to each activity the same selection probability. Hence, the probability of activity i to be selected is
$$p_i = \frac{1}{|D|},$$
in which D is the set of activities, and $|D|$ is the number of activities.
- *Biased random sampling* assigns the probability dependent on the priority rules applied, to make sure each activity has a sensible chance of being selected. If the priority rule is to choose the activity with the highest priority value v_i, the probability of activity i being selected from the set D is $p_i = v_i / \sum_{j \in D} v_j$. Besides, if the activity with the smallest priority value should be chosen, the probabilities can be calculated as $p_i = 1/(v_i \times \sum_{j \in D} 1/v_j)$.

- *Regret based biased random sampling* calculates the probabilities indirectly using regret values. If again the objective of the priority rule is to select the activity with the highest priority value v_i, then the regret value r_i of activity i equals the difference between the priority value v_i and the smallest priority value in the set D: $r_i = v_i - \min_{j \in D} v_j$. If, however, the activity with the smallest priority value has to be chosen, then the regret value r_i of activity i equals the difference between the largest priority value in the set D and the priority value v_i: $r_i = \max_{j \in D} v_j - v_i$. In this study, we modify the regret values as follows: $r'_i = r_i + \varepsilon$. The inclusion of a constant $\varepsilon > 0$, which is set to 1 in this paper, ensures that at each decision the probability of being selected for each activity is strictly positive. The probability p_i of selecting activity i from the set D can then be calculated as $p_i = r'_i / \sum_{j \in D} r'_j$.

4.2. Algorithm description

The first heuristic algorithm **Heuristic 1** is based on all the feasible maximal solutions (columns) for each subproblem (orbit), which can be obtained by labeling-based dynamic programming as described in the exact algorithm. Differently from the exact algorithm that enumerates all combinations of the columns to get all feasible solutions, **Heuristic 1** selects columns for each orbit and combines them heuristically to get a subset of combinations, which are corresponding to a subset of feasible solutions. Afterwards, the feasible solution in the subset with the maximum objective will be marked as the best solution.

```
Heuristic 1: Column-based heuristic selection
Step 1. Obtain the set of columns $COL_k$ for each orbit $k$, $k \in O$, with labeling-based
        dynamic programming algorithm (see Section 3.2).
Step 2. Compute the profits of the columns
        $prof_{kr} = \sum_{i \in T} \varpi_i \cdot p_{ik} \cdot \pi_{irk}$, for each column $col_{kr} \in COL_k$, each orbit $k \in O$, in
        which $\pi_{irk} = 1$ if task $i$ is scheduled to orbit $k$ in column $col_{kr}$, otherwise $\pi_{irk} = 0$.
Step 3. Generate a certain number $Z$ of feasible solutions based on the heuristic combination
        of the columns
        For $l = 1, \ldots, Z$
            For each orbit $k \in O$
                Select a column $col_{kr}$ from $COL_k$ with biased random sampling, with the
                profits of the columns being priority values, and the priority rule is to select
                the column with the highest priority value.
            End for
            Combine the obtained columns for each orbit to form a feasible solution $sol_l$,
            and put $sol_l$ into the solution set $SOL$.
        End for
Step 4. Select the best solution from $SOL$, i.e., the solution with the maximum objective
        value.
```

Heuristic 2 produces all the feasible maximal solutions for each orbit, but only the L best solutions are selected on each orbit for combination. Hence, **Heuristic 2** is identical to **Heuristic 1** except that there is one more step between **step 2** and **step 3**, which is described as below:

```
Step 2'. Cut down and update the column sets
         For each orbit $k \in O$
             Choose a predefined number $L$ of columns from $COL_k$ with maximum profits
             to form a new and small column set $COL_k$, namely delete the other columns.
         End for
```

Both **Heuristic 1** and **Heuristic 2** get all columns for each orbit with dynamic programming which will be much too time consuming. Besides, if we cannot use enough time to get all columns for each orbit, we will not obtain any solutions, neither optimal nor feasible ones. To overcome this drawback, we consider obtaining a subset of columns for each orbit using a heuristic procedure, thus we get **Heuristic 3**. **Heuristic 3** adopts a backtracking approach to obtain L solutions for each orbit. In detail, the procedure starts from the dummy terminating node t, and it selects a direct predecessor i using biased random sampling. Subsequently, the above procedure will be repeated until reaching the starting node s, which implies that a feasible solution has been produced.

Finally, the above procedures will be repeated L times to get L feasible solutions. As a consequence, **Heuristic 3** is identical to **Heuristic 1** except that **Step 1** is replaced by:

```
Step 1'. Generate a predefined number $L$ of columns for each orbit
         For each orbit $k \in O$
            $COL_k \leftarrow \emptyset$
            For $l = 1, \ldots, L$
               $j \leftarrow t$ // $t$ is the terminate node of orbit $k$
               While $j \neq s$ // $s$ is the starting node
                  Select node $i$, $i \in \Gamma^{-1}(j)$ with biased random sampling, in which $\Gamma^{-1}(j)$ is
                  the set of all predecessors of $j$. The priority value of node $i$ is the sum of
                  the profit of node $i$ and the profits of all (direct and indirect) predecessors
                  of node $i$, and the priority rule is to select the predecessor with the highest
                  priority value.
                  $j \leftarrow i$
               End while
               Obtain the column $col_{kl}$, $COL_k \leftarrow COL_k \cup \{col_{kl}\}$
            End for
         End for
```

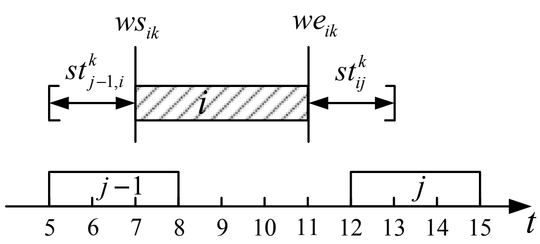

Figure 6: Conflicts of task i

All the above algorithms can be attributed to column-based heuristics, which combine the columns for each orbit to form feasible solutions. In contrast, **Heuristic 4** handles the problem as a knapsack problem and solves it based on task allocation and retraction. Before describing **Heuristic 4** in detail, some definitions are proposed below:

Definition 3. Task Requirement

We define task requirement TR_i that represents the priority to schedule task i:

$$TR_i = \frac{\varpi_i \cdot [1 - \prod_{k \in O}(1 - p_{ik} b_{ik})]}{\sum\limits_{k \in O} b_{ik}}$$

Definition 4. Conflict

A conflict, say $Conf_{ik}$, is defined as the set of conflicting tasks that are scheduled to orbit k and violating the setup time constraints with task i.

Definition 5. Conflict Set

We define the conflict set $ConfSet_i$ of task i to be the set of all distinct conflicts on all orbits.

Figure 6 is a simple example of a task i, where $Conf_{ik} = \{j-1, j\}$. Assume task i is only available on orbit k, the conflict set of task i is $ConfSet_i = \{Conf_{ik}\}$.

Definition 6. Task Retraction Expense

The task retraction expense TRE_{ik} is defined as the decrease in expected profits if task i is retracted from orbit k.

$$TRE_{ik} = \varpi_i \cdot [P_i(O_i) - P_i(O_i \setminus k)]$$

in which O_i is the set of orbits that task i has been scheduled, and $P_i(O_i)$ represents the relevant completing probability of task i. It must be noted that $P_i(O_i) = 0$ if $O_i = \emptyset$.

```
Heuristic 4 Knapsack-based heuristic
Step 1. Initialize the current optimal solution $CurOptSol = \emptyset$, current optimal objective
        value $CurOptObj = 0$, solution index $l = 0$, predefined number of solutions $Z$.
Step 2. If $l < Z$, Go to Step 3, otherwise the algorithm ends.
Step 3. Initialize the task set $T$, the orbit set $O$, the set of scheduled tasks $Sche_k = \emptyset$ for
        each orbit $k$, $k \in O$, and the conflict set $ConfSet_i = \emptyset$ for each task $i$, $i \in T$.
Step 4. Update the optimal solution
        If $T = \emptyset$
          Compute the objective value $Obj_l$ of the current solution $Sol_l$
          If $Obj_l > CurOptObj$
            $CurOptObj \leftarrow Obj_l$, $CurOptSol \leftarrow Sol_l$
          End if
          $l \leftarrow l + 1$, go to Step 2
        Else
          Go to Step 5.
        End if
Step 5. Select task $i$ from $T$ with maximum task requirement $TR_i$, $T \leftarrow T \setminus i$.
Step 6. Schedule task $i$ on each orbit
        For each orbit $k \in O$
          If there is no conflict of task $i$ on orbit $k$
            Schedule task $i$ to orbit $k$, $Sche_k \leftarrow Sche_k \cup \{i\}$
            While memory and energy constraints are not all satisfied
              Select task $j$ from $Sche_k$ with regret based biased random sampling, with
              the task retraction expense $TRE_{jk}$ being the priority values, and the pri-
              ority rule is to select the task with the smallest priority value.
              Retract task $j$ from orbit $k$, $Sche_k \leftarrow Sche_k \setminus j$, and update the relevant
              memory and energy.
            End while
          Else
            Obtain the conflict $Conf_{ik}$, $ConfSet_i \leftarrow ConfSet_i \cup \{Conf_{ik}\}$
          End if
        End for
Step 7. If task $i$ has been successfully scheduled to at least one orbit, go to Step 4,
        otherwise go to Step 8.
Step 8. Calculate the conflict expense $ConfE_{ik}$ for each conflict $Conf_{ik}$, $Conf_{ik} \in$
        $ConfSet_i$
Step 9. Select the conflict $Conf_{ik}$ from $ConfSet_i$ with regret based biased random sam-
        pling, with the priority values being $\varpi_i p_{ik} - ConfE_{ik}$ for each conflict, and the
        priority rule is to select the conflict with the highest priority value.
Step 10. Schedule task $i$ to orbit $k$ with conflict $Conf_{ik}$
         $Sche'_k \leftarrow Sche_k$
         For each task $j$, $j \in Conf_{ik}$
           $Sche_k \leftarrow Sche_k \setminus j$
         End for
         $Sche_k \leftarrow Sche_k \cup \{i\}$
         If memory and energy constraints are not both satisfied
           $Sche_k \leftarrow Sche'_k$
         End if
         Go to Step 4
```

Definition 7. Conflict Expense

We define the conflict expense $ConfE_{ik}$ of Conflict $Conf_{ik}$ as the decrease in expected profits if we retract the conflicting tasks in $Conf_{ik}$,

$$ConfE_{ik} = \sum_{j \in Conf_{ik}} TRE_{jk}$$

Firstly, **Heuristic 4** selects the task i from the set T with maximum requirement, and schedules i to each orbit. For an orbit k, if there are no conflicting tasks, task i will be scheduled on k, and we retract some tasks based on regret based biased random sampling to satisfy both the memory and energy constraints if necessary. If task i has been successfully scheduled on at least one orbit, we will access the next task; otherwise, we will select a conflict $Conf_{ik}$ from the set $ConfSet_i$ with regret based biased random sampling. Subsequently, the tasks in $Conf_{ik}$ will be retracted to accommodate task i if both the memory and energy constraints are satisfied. Then we will select the next task until $T = \emptyset$, which implies that we have obtained a feasible solution. Finally, the above procedure will be repeated Z times to get Z feasible solutions, and the best solution will be selected.

```
Heuristic 5 Heuristic based on conflict resolution
Step 1. Initialize the current optimal solution $CurOptSol = \emptyset$, current optimal objective
        value $CurOptObj = 0$, solution index $l = 0$, predefined number of solutions $Z$.
Step 2. Initialize the task set $T$, the orbit set $O$, schedule each task $i$, $i \in T$ to all the
        available orbits to get an initial solution $IniSol$ that is normally infeasible.
Step 3. If $l < Z$, Go to Step 4, otherwise the algorithm ends.
Step 4. Obtain the conflicting assignment set $ConfA$ of the schedule $IniSol$, as well as
        the conflicting assignments $ConfAs_{ik}$ for each assignment $Assign_{ik}$, $Assign_{ik} \in$
        $ConfA$.
Step 5. For each conflicting assignment $Assign_{ik}$, $Assign_{ik} \in ConfA$, compute the retrac-
        tion preference $RP_{ik}$.
Step 6. Select the conflicting assignment $Assign_{ik}$ from $ConfA$ with regret based biased
        sampling, with the retraction preference being the priority values, and the priority
        rule is to select the assignment with the highest priority value. $ConfA \leftarrow ConfA \setminus$
        $Assign_{ik}$, retract the assignment $Assign_{ik}$ from the initial solution, $IniSol \leftarrow$
        $IniSol \setminus Assign_{ik}$
Step 7. Update the conflicting assignment set $ConfA$ and the retraction preference of the
        relevant assignments.
Step 8. If $ConfA = \emptyset$, go to Step 9, otherwise go to Step 6.
Step 9. Resolve the memory and energy conflicts
        For each orbit $k$, $k \in O$
            While memory and energy constraints are not both satisfied
                Select task $j$ from the set $Sche_k$ of tasks that are scheduled to orbit $k$ with
                regret based biased random sampling, with the task retraction expense $TRE_{jk}$
                being priority values, and the priority rule is to select the tasks with the
                smallest priority value.
                Retract task $j$ from orbit $k$, $Sche_k \leftarrow Sche_k \setminus j$, and update the relevant
                memory and energy;
            End while
        End for
Step 10. Compute the objective value $Obj_l$ of the current solution $Sol_l$. If $Obj_l >$
         $CurOptObj$, update the optimal solution $CurOptObj \leftarrow Obj_l$, $CurOptSol \leftarrow$
         $Sol_l$. $l \leftarrow l + 1$, go to Step 3.
```

Also, on the basis of the knapsack formulation, **Heuristic 5** firstly allocates each task to all available orbits to get an initial solution, which is normally infeasible. Then, **Heuristic 5** removes all conflicts to get a solution that respects the setup time constraints. Afterwards, some tasks are also removed for each orbit in order to satisfy the memory and energy constraints, yielding a feasible solution. Finally, the above procedures will be repeated for Z times to obtain Z feasible solutions, and the best solution will be selected. Hence, **Heuristic 5** is inspired by the notion of conflict resolution.

Definition 8. Assignment

An assignment, say $Assign_{ik}$, refers to the assignment of orbit k to task i, namely allocate task i to orbit k. A is the set of all assignments, $Assign_{ik} \in A$.

Definition 9. Conflicting Assignment Set

Conflicting assignment set, say $ConfA$, is defined as the set of all conflicting assignments. $Assign_{ik} \in ConfA$ if assignment $Assign_{ik}$ is conflicting with some other assignments in the current schedule.

Table 3: Parameters of satellites

Satellite	Slewing velocity	Startup time	Shutdown time	Stability time	Memory /time	Energy /time	Energy /deg
CBERS-2	2	5	8	3	2	1.5	1.5
IKONOS-2	2.5	8	5	6	4	2.5	4
SPOT-5	3	10	10	9	3	3.5	1

Definition 10. Conflicting Assignments
Conflicting assignments $ConfAs_{ik}$, corresponding to assignment $Assign_{ik}$, is the set of assignments that are conflicting with $Assign_{ik}$.
Definition 11. Assignment Retraction Expense
The assignment retraction expense ARE_{ik} is defined as the decrease of the objective if assignment $Assign_{ik}$ is retracted.
Definition 12. Implicit Assignment Value
We define the implicit assignment value $ImpAssignV_{ik}$ of retracting assignment $Assign_{ik}$ as below: if we don't retract assignment $Assign_{ik}$, we have to retract the assignments in $ConfAs_{ik}$, which will also result in the objective decreasing. Hence, retraction of assignment $Assign_{ik}$ has the value to avoid retracting assignments in $ConfAs_{ik}$, which is defined as:

$$ImpConfV_{ik} = \sum_{Assign_{jk} \in ConfAs_{ik}} ARE_{jk}$$

Definition 13. Retraction Preference
The retraction preference RP_{ik} of the conflicting assignment $Assign_{ik}$, $Assign_{ik} \in ConfA$ is defined as

$$RP_{ik} = ImpConfV_{ik} - ARE_{ik}$$

5. Computational results

For this section, we created a great number of problem instances in order to evaluate the effectiveness and efficiency of our proposed approaches. The computational tests have two components: on some small instances, we verify the superiority of our proposed non-linear robust model, we verify the feasibility and we evaluate the performance of both the exact algorithm and the heuristics; on the other hand, the performances of the heuristics are evaluated and compared on some large problem instances.

In order to verify the effectiveness and efficiency of our algorithm, the tasks are randomly generated in the area: latitude 0°-60° and longitude 0°-150°. Without loss of generality, the profits of tasks are integers, uniformly distributed in the interval [1,10]. In correspondence with the literature [5, 11, 23, 36], three different satellites are considered in this paper. The parameters of the satellites are outlined in Table 3, and the orbit models of the satellites are obtained from the Satellite Tool Kit (STK). Hence, the time windows and slewing angles of observations can be calculated by STK in advance. In addition, the memory capacity and energy capacity for each orbit are randomly generated in the intervals [120,160] and [180,240], respectively. Considering the uncertainties of clouds, for each time window of observation, the probability that there is no presence of clouds, i.e. the observation is successful, will be uniformly distributed in [0.2,1].

The algorithms were implemented in C++ and ran on a personal laptop equipped with an Intel(R) Core(TM) i5-2430M 2.40 GHz (2 processors) and 4 Gb RAM, with operating system Windows 7.

5.1. Performance evaluation on the small problem instances

Our first computational experiment was ran for verifying the superiority of our proposed model and the feasibility of both the exact algorithm and the heuristics on some small problem instances. In this section, we firstly set the scheduling horizon to be 6 hours, which is corresponding to 9 orbits.

The numbers of tasks are 10, 20, 30, 40, 50 and 60, respectively. Then we increase the number of orbits to 21, and the numbers of tasks are set to be 10, 20 and 30, respectively. It must be noted that for 21 orbits, we can only test less tasks because the exact algorithm will fail to solve for more tasks due to its large space complexity. For each parameter setting, we will randomly generate 10 problem instances. Hence, we will have 90 instances in total. For all problem instances, the exact algorithm and the heuristics were applied. With respect to the heuristics, the predefined number of feasible solutions are set to 1000, and the number of columns for each orbit is set to 20.

In order to verify the superiority of our model, we compare the solutions of our model with those of Liao's model [29]. Liao also formulated the presence of clouds as stochastic events with some probabilities, and considered scheduling a task to only one resource and performing only once, which resembles the traditional deterministic scheduling. In addition, the objective of scheduling is [29]:

$$\max_{\gamma} E[\sum_{k} \varpi_k x_k + \sum_{i=0}^{I} \sum_{n=1}^{N} y_{in} \sum_{j=1,j\neq i}^{I} \gamma_{ijn}].$$

The objective is two-fold: 1) maximize the profits of the complete tasks without uncertainties; 2) maximize the expected number of the complete tasks under uncertainties of clouds. In this study, Liao's model is directly solved by CPLEX because it is a standard mixed integer programming model.

For the sake of comparison, we test the obtained solutions on a large sample with the sample size being 100000 for each instance. Then, we will compare the following statistics: the minimal, average and maximum profits of the complete tasks on the sample.

Table 4 shows the comparison results for each problem instance, in which column "m" shows the number of orbits, "n" indicates the number of tasks, and "No." represents the serial number of the instance. Besides, columns "a", "b" and "c" are corresponding to the minimum, average and maximum obtained scheduling profits of all scenarios in the sample, respectively. Because CPLEX is used to solve Liao's model that schedules each task to at most one orbit and therefore lacks robustness, the solutions of CPLEX are on average worse than those of our algorithms, which has been shown in Table 4. Hence, we can conclude that the proposed model is superior to Liao's model under uncertainties of clouds.

In addition, Table 5 describes the performances of both the exact algorithm and the heuristics for solving the small instances, in which column "m" denotes the number of orbits, and "n" indicates the number of tasks. In addition, column "Obj" contains the average of the scheduling objective values for the 10 instances, and "Time" contains the average values of the solution times. In columns "Time", 0.000 implies that the average running time is less than 0.001 second. In columns "Obj" of the heuristics, a bolded number denotes that the heuristic gets the optimal solutions for all problem instances in that set. From Table 5, it is observed that **Heuristic 5** can get optimal solutions for more instances than the other heuristics. Besides, for most problem instances, **Heuristic 2** and **Heuristic 5** can get better solutions, as well as **Heuristic 3** and **Heuristic 4** perform worse normally. Note that the objective function of Liao's model that is solved by CPLEX is different from that of ours. Hence, the comparison between the objective values of CPLEX with those of our algorithms is meaningless, and we only compare the solution times. With respect to the solution times, we can conclude that both the exact algorithm and the heuristics can solve the small problems very fast and efficiently (less than 1 second), which are also much faster than CPLEX.

Table 4: Performance evaluation on the samples

m	n	No.	Exact Algorithm			Heuristic1			Heuristic2			Heuristic3			Heuristic4			Heuristic5			CPLEX		
			a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
		0	0	6.90	13	0	6.90	13	0	6.90	13	0	6.90	13	0	6.90	13	0	6.90	13	0	6.71	13
		1	0	11.27	18	0	11.27	18	0	11.27	18	0	11.27	18	0	11.27	18	0	11.27	18	0	11.26	18
		2	6	20.34	23	6	20.34	23	6	20.34	23	6	20.34	23	6	20.34	23	6	20.34	23	1	18.46	23
		3	0	21.07	22	0	21.07	22	0	21.07	22	0	21.07	22	0	21.07	22	0	21.07	22	0	19.14	22
		4	4	22.40	23	4	22.40	23	4	22.40	23	4	22.40	23	4	22.27	23	4	22.27	23	0	20.75	23
	10	5	0	1.91	2	0	1.91	2	0	1.91	2	0	1.91	2	0	1.91	2	0	1.91	2	0	1.62	2
		6	0	5.50	14	0	5.50	14	0	5.50	14	0	5.50	14	0	5.50	14	0	5.50	14	0	5.50	14
		7	0	4.68	15	0	4.68	15	0	4.68	15	0	4.68	15	0	4.68	15	0	4.68	15	0	4.68	15
		8	0	16.38	23	0	16.38	23	0	16.38	23	0	16.38	23	0	14.99	23	0	14.99	23	0	14.54	23
		9	0	16.57	29	0	16.57	29	0	16.57	29	0	16.57	29	0	16.57	29	0	16.57	29	0	16.55	29
		AVG	1.0	12.70	18.2	1.0	12.70	18.2	1.0	12.70	18.2	1.0	12.70	18.2	1.0	12.55	18.2	1.0	12.55	18.2	0.1	11.92	18.2
		0	0	15.47	22	0	15.47	22	0	15.47	22	0	15.47	22	0	14.96	22	0	14.96	22	0	13.65	22
		1	0	29.08	40	0	29.08	40	0	29.08	40	0	29.08	40	0	25.25	40	0	29.08	40	0	26.34	40
		2	4	35.23	42	4	35.23	42	4	35.23	42	4	35.23	42	0	33.46	42	0	33.46	42	0	30.45	42
		3	0	20.11	32	0	20.11	32	0	20.11	32	0	20.11	32	0	19.37	32	0	19.37	32	0	14.90	32
		4	0	22.61	32	0	22.61	32	0	22.61	32	0	22.61	32	0	22.61	32	0	22.61	32	0	20.93	32
9	20	5	6	30.69	41	6	30.69	41	6	30.69	41	6	30.69	41	6	30.69	41	6	30.69	41	3	30.38	41
		6	4	31.12	37	4	31.12	37	4	31.12	37	4	31.12	37	4	31.07	37	4	31.07	37	2	29.08	37
		7	7	30.79	39	7	30.79	39	7	30.79	39	7	30.79	39	4	30.59	39	4	30.59	39	3	29.92	39
		8	0	15.46	25	0	15.46	25	0	15.46	25	0	15.46	25	0	15.46	25	0	15.46	25	0	14.76	25
		9	7	36.30	58	7	36.30	58	7	36.30	58	7	36.30	58	5	36.04	58	5	36.04	58	1	33.85	58
		AVG	2.8	26.69	36.8	2.8	26.69	36.8	2.8	26.69	36.8	2.8	26.69	36.8	1.9	25.95	36.8	1.9	26.33	36.8	0.9	24.43	36.8
		0	5	27.30	38	5	27.30	38	5	27.30	38	5	27.30	38	5	24.98	32	5	24.98	32	2	22.80	32
		1	10	37.83	51	10	37.83	51	10	37.83	51	10	37.83	51	10	37.83	51	10	37.83	51	6	36.45	51
		2	1	16.22	19	1	16.22	19	1	16.22	19	1	16.22	19	1	16.22	19	1	16.22	19	0	15.18	19
		3	0	16.03	34	0	16.03	34	0	16.03	34	0	16.03	34	0	16.03	34	0	16.03	34	0	16.03	34
		4	0	28.01	42	0	28.01	42	0	28.01	42	0	28.01	42	0	27.60	42	0	27.87	42	0	26.19	42
	30	5	10	37.38	41	10	37.38	41	10	37.38	41	10	37.28	41	10	37.12	41	10	37.12	41	10	36.83	41
		6	5	39.53	49	5	39.53	49	5	39.53	49	5	39.53	49	5	38.94	49	5	38.94	49	5	35.93	49
		7	0	17.73	26	0	17.73	26	0	17.73	26	0	17.73	26	0	16.93	26	0	16.93	26	0	15.80	26
		8	4	31.19	40	4	31.19	40	4	31.19	40	4	31.19	40	4	31.19	40	4	31.19	40	0	30.27	40
		9	25	48.49	55	25	47.95	55	25	47.95	55	20	47.04	55	16	41.27	48	12	41.47	48	2	39.02	55
		AVG	6.0	29.97	39.5	6.0	29.92	39.5	6.0	29.92	39.5	5.5	29.82	39.5	5.1	28.81	38.2	4.7	28.86	38.2	2.5	27.45	38.9

m	n	No.	Exact Algorithm			Heuristic1			Heuristic2			Heuristic3			Heuristic4			Heuristic5			CPLEX		
			a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
9	40	0	8	46.35	62	8	46.35	62	8	46.35	62	8	42.37	59	5	41.93	54	5	41.93	54	1	38.94	57
		1	3	41.32	68	3	41.32	68	3	41.32	68	3	41.32	68	3	37.03	65	3	37.03	65	1	34.61	68
		2	25	64.79	84	25	64.79	84	25	64.79	84	25	64.79	84	22	60.39	81	20	62.20	81	18	61.93	84
		3	4	32.91	52	4	32.51	52	4	32.51	52	4	32.51	52	0	30.58	49	0	30.58	49	0	28.91	50
		4	35	90.01	116	43	88.35	116	35	89.98	116	35	88.13	116	26	79.80	114	32	81.87	110	24	77.93	115
		5	9	47.79	60	9	47.75	60	9	47.79	60	9	47.75	60	9	47.10	60	9	47.10	60	9	47.05	60
		6	13	53.58	72	13	53.58	72	13	53.58	72	13	53.58	72	12	52.46	70	12	52.46	70	8	49.93	72
		7	6	24.62	34	6	24.62	34	6	24.62	34	6	24.62	34	1	22.85	33	2	23.82	33	3	21.45	33
		8	24	65.35	98	24	65.35	98	24	65.35	98	21	58.65	90	17	58.72	91	17	58.72	91	15	56.34	91
		9	30	76.76	106	30	76.76	106	30	76.76	106	30	76.65	106	19	68.38	104	19	70.35	98	18	66.42	105
		AVG	15.7	54.35	75.2	16.5	54.14	75.2	15.7	54.31	75.2	15.4	53.04	74.1	11.4	49.92	72.1	11.9	50.60	71.1	9.7	48.35	73.5
	50	0	40	88.62	107	40	88.07	107	38	85.81	107	43	89.49	107	27	77.62	105	31	80.65	102	30	78.27	107
		1	23	68.70	88	23	68.70	88	23	68.70	88	20	67.29	88	20	64.84	88	17	63.86	84	14	60.31	88
		2	18	52.58	64	18	52.58	64	18	52.58	64	16	51.13	61	24	53.84	63	18	52.58	64	17	48.90	66
		3	3	41.75	61	3	41.75	61	3	41.75	61	3	41.75	61	0	40.51	61	0	40.51	61	0	40.09	61
		4	25	73.67	114	24	70.96	108	24	70.53	108	24	71.00	108	14	64.33	101	19	65.92	101	14	64.32	101
		5	18	56.28	87	18	56.28	87	18	56.28	87	18	56.28	87	10	54.09	82	10	54.09	82	8	51.54	87
		6	16	45.50	68	16	45.50	68	16	45.50	68	16	45.50	68	6	41.22	63	6	41.22	63	6	40.52	63
		7	12	44.44	61	12	44.44	61	12	44.44	61	7	44.27	64	8	44.08	65	12	44.44	61	5	43.52	70
		8	21	67.76	96	21	67.76	96	24	67.50	98	26	67.24	96	19	61.30	90	18	58.67	87	10	48.63	89
		9	22	66.96	95	22	66.96	95	22	66.96	95	22	66.96	95	14	50.44	72	14	50.44	72	6	44.84	76
		AVG	19.8	60.63	84.1	19.7	60.30	83.5	19.8	60.01	83.7	19.5	60.09	83.5	14.2	55.23	79.0	14.5	55.24	77.7	11.0	52.10	80.8
	60	0	14	54.58	78	14	54.58	78	14	54.58	78	14	52.27	75	9	49.12	76	12	50.14	76	9	48.52	78
		1	13	55.72	82	13	55.72	82	13	55.72	82	13	51.14	80	4	47.39	69	4	47.39	69	4	46.05	78
		2	26	79.23	102	26	78.34	102	26	78.34	102	26	79.17	102	25	72.21	98	24	73.49	97	11	57.18	101
		3	33	71.60	88	33	71.60	88	33	71.60	88	30	68.54	82	24	65.85	88	24	65.28	87	13	57.99	88
		4	17	63.04	100	17	63.04	100	17	63.04	100	16	61.50	94	10	56.53	90	10	56.53	90	8	53.37	93
		5	19	61.22	92	19	61.21	92	19	61.22	92	19	60.90	92	16	58.73	90	16	58.69	87	12	52.83	94
		6	10	55.82	81	19	53.76	81	19	55.59	81	19	53.76	81	9	47.80	72	10	50.50	81	10	48.76	81
		7	31	72.80	109	31	72.67	109	31	72.67	109	31	71.66	105	28	70.24	108	27	68.76	107	26	69.53	108
		8	13	55.26	87	13	55.26	87	13	55.26	87	13	55.26	87	10	51.54	81	11	51.46	81	9	48.51	82
		9	17	75.92	115	17	75.92	115	17	75.92	115	17	75.92	115	12	66.48	106	10	67.91	106	5	64.73	110
		AVG	19.3	64.52	93.4	20.2	64.21	93.4	20.2	64.40	93.4	19.8	63.01	91.3	14.7	58.59	87.8	14.8	59.01	88.1	10.7	54.75	91.3

m	n	No.	Exact Algorithm			Heuristic1			Heuristic2			Heuristic3			Heuristic4			Heuristic5			CPLEX		
			a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
21	10	0	8	28.42	38	8	28.42	38	8	28.42	38	8	28.42	38	8	28.42	38	8	28.42	38	0	25.73	38
		1	0	26.61	40	0	26.61	40	0	26.61	40	0	26.61	40	0	26.61	40	0	26.61	40	0	23.07	40
		2	7	27.93	43	7	27.93	43	7	27.93	43	7	27.93	43	7	27.93	43	7	27.93	43	1	23.91	43
		3	0	26.67	40	0	26.67	40	0	26.67	40	0	26.67	40	0	26.67	40	0	26.67	40	0	23.16	40
		4	8	39.34	48	8	39.34	48	8	39.34	48	8	39.34	48	8	39.34	48	8	39.34	48	6	36.19	48
		5	2	11.02	14	2	11.02	14	2	11.02	14	2	11.02	14	2	11.02	14	2	11.02	14	0	10.39	14
		6	1	26.77	31	1	26.77	31	1	26.77	31	1	26.77	31	1	26.52	31	1	26.52	31	0	21.59	31
		7	0	14.08	20	0	14.08	20	0	14.08	20	0	14.08	20	0	14.08	20	0	14.08	20	0	13.78	20
		8	20	56.10	64	20	56.10	64	20	56.10	64	20	56.10	64	16	55.23	64	16	55.23	64	8	50.45	64
		9	3	22.74	26	3	22.74	26	3	22.74	26	3	22.74	26	3	22.74	26	3	22.74	26	0	21.46	26
		AVG	4.9	27.97	36.4	4.9	27.97	36.4	4.9	27.97	36.4	4.9	27.97	36.4	4.5	27.86	36.4	4.5	27.86	36.4	1.5	24.97	36.4
	20	0	11	56.00	91	11	56.00	91	11	56.00	91	11	56.00	91	11	55.70	91	11	55.70	91	9	53.05	91
		1	24	69.94	100	24	69.94	100	24	69.94	100	24	69.94	100	23	68.04	100	23	68.04	100	19	62.83	100
		2	28	77.96	108	28	77.96	108	28	77.96	108	28	77.96	108	24	71.47	104	28	75.85	104	24	73.00	104
		3	45	85.21	107	45	85.21	107	45	85.21	107	45	85.21	107	41	82.60	107	40	82.02	106	33	76.95	107
		4	9	38.58	49	9	38.58	49	9	38.58	49	9	38.58	49	9	38.58	49	9	38.58	49	7	36.11	49
		5	23	68.22	95	23	68.22	95	23	68.22	95	23	68.22	95	23	64.85	95	23	64.85	95	13	62.00	95
		6	20	61.70	81	20	61.70	81	20	61.70	81	20	61.70	81	20	60.64	81	20	60.64	81	15	56.18	81
		7	23	64.15	78	23	64.15	78	23	64.15	78	23	64.15	78	23	64.15	78	23	64.15	78	19	58.45	78
		8	43	89.77	104	43	89.77	104	43	89.77	104	42	88.37	103	36	86.49	104	36	87.05	104	26	77.42	104
		9	11	48.91	71	11	48.91	71	11	48.91	71	11	48.91	71	11	47.50	71	11	47.50	71	8	45.26	71
		AVG	23.7	66.04	88.4	23.7	66.04	88.4	23.7	66.04	88.4	23.6	65.90	88.3	22.1	64.00	88.0	22.4	64.44	87.9	17.3	60.12	88.0
	30	0	27	78.25	107	27	78.25	107	27	78.25	107	27	77.78	107	20	73.00	104	20	73.00	104	17	67.47	106
		1	46	91.52	121	46	91.52	121	46	91.52	121	46	90.41	121	44	90.98	121	44	90.55	121	30	82.69	121
		2	61	123.03	153	62	120.54	146	62	120.54	146	61	120.96	153	55	118.60	153	55	118.99	153	43	108.64	153
		3	49	91.93	117	49	91.93	117	49	91.93	117	49	91.33	117	47	92.15	115	49	91.33	117	37	85.13	117
		4	34	82.98	119	34	82.98	119	34	82.98	119	34	82.66	119	34	78.92	105	32	78.74	110	23	73.07	112
		5	62	104.22	130	57	103.19	130	57	103.19	130	57	98.84	124	57	96.08	124	57	99.65	129	46	92.43	130
		6	47	96.87	131	35	94.90	125	35	94.90	125	36	90.04	120	41	97.49	131	41	93.96	131	40	90.13	131
		7	55	106.54	124	54	106.03	124	54	105.68	124	56	101.73	120	55	104.50	124	51	105.25	124	43	96.83	130
		8	32	79.48	112	32	79.48	112	32	79.48	112	32	79.41	112	23	71.61	107	23	71.61	107	20	67.69	108
		9	51	89.94	104	51	89.02	104	53	89.85	104	51	89.02	104	40	80.57	102	39	85.15	103	31	77.21	104
		AVG	46.4	94.47	121.8	44.7	93.78	120.5	44.9	93.83	120.5	44.9	92.22	119.7	41.6	90.39	118.6	41.1	90.82	119.9	33.0	84.13	121.2

Table 5: Performance evaluation on the small problems

m	n	Exact Algorithm		Heuristic1		Heuristic2		Heuristic3		Heuristic4		Heuristic5		CPLEX
		Obj	Time	Obj	Time	Obj	Time	Obj	Time	Obj	Time	Obj	Time	Time
9	10	12.5441	0.000	**12.5441**	0.000	**12.5441**	0.000	**12.5441**	0.000	**12.5441**	0.002	**12.5441**	0.000	0.097
	20	26.3332	0.000	**26.3332**	0.000	**26.3332**	0.000	26.2148	0.000	25.9514	0.002	**26.3332**	0.000	0.187
	30	28.8549	0.000	28.8469	0.000	28.8469	0.000	28.8280	0.000	28.8089	0.004	**28.8549**	0.002	0.212
	40	50.5927	0.034	50.4335	0.002	50.5901	0.002	49.3968	0.003	49.9066	0.008	50.5314	0.003	0.306
	50	55.5849	0.047	55.1814	0.000	55.2485	0.002	53.8159	0.000	54.9366	0.014	55.2023	0.002	0.332
	60	59.1567	0.056	58.5871	0.002	58.8266	0.000	56.5123	0.002	58.6562	0.016	58.9921	0.002	0.429
21	10	27.8513	0.000	**27.8513**	0.002	**27.8513**	0.003	**27.8513**	0.000	**27.8513**	0.008	**27.8513**	0.000	0.112
	20	64.4809	0.000	64.4221	0.003	64.4221	0.004	64.1291	0.002	63.9871	0.010	64.4221	0.002	0.256
	30	91.2543	0.549	90.4597	0.002	90.5318	0.004	88.8026	0.002	90.0757	0.030	90.7597	0.011	0.466

Table 6: Parameter setting for Heuristic 2

Number of orbits	Number of tasks	ColNum					
		20	40	60	80	100	120
21	120	**266.243**	260.458	259.053	257.167	255.686	255.918
	160	332.227	326.802	365.518	366.826	**368.686**	313.108
	200	394.446	385.398	450.676	458.468	**462.991**	375.288
42	120	**416.727**	409.864	407.020	404.470	401.282	401.327
	160	518.572	513.077	531.058	531.771	**532.193**	499.465
	200	612.545	642.876	675.658	681.091	**683.462**	628.059
AVG		423.460	423.079	448.164	449.965	**450.717**	412.194

5.2. Performance evaluation on the large problem instances

Due to its large space complexity, it is truly infeasible for the exact algorithm to solve the large-scale problems that are more practical. Hence, heuristics will be applied more extensively in practice. In this section, to evaluate the performance of the proposed heuristics, we randomly create a number of large-scale problem instances. In detail, the numbers of tasks are set to 120, 160 and 200, respectively, and the numbers of orbits are 21 and 42, respectively. Similarly with the former experiments, for each parameter setting, we randomly create 10 problem instances. Therefore, we have 60 instances in total. Besides, differently from the former experiments, we do not limit the number of solutions beforehand, but we set a time limit on applying each heuristic, and thus the numbers of solutions considered depend on the solution times. The time limits are set to 1, 10 and 60 seconds, respectively. We will compare the performances of the heuristics for each time limit.

Then, before we compare the different heuristics, we firstly need to set the parameter that determines the number of columns for each orbit for **Heuristic 2** and **Heuristic 3**, in order to make the performances of the algorithms as best as possible. For parameter setting, the time limits are set to 10 seconds. In addition, to make the performance comparison fair, we will set two problem instance sets A and B as previously described, respectively. Firstly, we will test the parameter of **Heuristic 2** and **Heuristic 3** on the instance set A, and then the performance comparison for all the heuristics will be performed on the instance set B.

Table 6 and Table 7 outline the testing results for parameter setting, which are the averages of the objective values. It is shown in Table 6 that setting the number of columns for each orbit to $ColNum = 100$ for **Heuristic 2** leads to the best solutions for most cases. Thus the parameter for **Heuristic 2** is set to 100 in the following experiments. Similarly with **Heuristic 2**, **Heuristic 3** also yields the best performance with parameter setting $ColNum = 100$, which will also be set in this manner in the following experiments.

After setting the parameters, we compare the scheduling results of the different heuristics with different time limits, which are shown in Tables 8 and 9. In Tables 8 and 9, columns "Obj" are the average of the objective values of the solvable instances, and "#Fea" are the numbers of problem instances that can be solved to get feasible solutions. For columns "Obj", the bold numbers imply the largest average value for this parameter setting, without considering the number of instances that are solved successfully. In addition, to make it more intuitive, we also present the comparison results in Figures 7-12.

It is illustrated in Table 8 and Figures 7-9 that for the problems of 21 orbits, when the number of tasks is less, namely 120 tasks, **Heuristic 4** performs somewhat better than the other algorithms. However, if the number of tasks increases to 160 or 200, **Heuristic 2** will be the best option for most cases. In addition, the performances of **Heuristic 4** and **Heuristic 5** are clearly worse than those of the other heuristics for the larger problems with the number of tasks being 160 or 200. Besides, if the time limit is small and there are more tasks, **Heuristic 1** and **Heuristic 2** fail to get feasible solutions for some instances. That is because these two heuristics are based on all columns for each orbit that will be obtained by dynamic programming, of which the solution time will increase exponentially with the number of tasks. Thus, if the dynamic programming algorithm cannot get all feasible columns for each orbit within the time limit, we cannot get any feasible solutions from **Heuristic 1** and **Heuristic 2**. Hence, **Heuristic 1** and **Heuristic 2** are not available for large-scale problems when the time limits are very small.

Table 7: Parameter setting for Heuristic 3

Number of orbits	Number of tasks	ColNum 20	40	60	80	100	120
21	120	236.524	235.047	236.217	238.496	**239.737**	237.262
	160	307.406	309.543	333.588	335.421	**338.577**	307.975
	200	373.416	373.243	416.720	**425.423**	423.242	370.361
42	120	369.714	372.766	369.143	371.694	**374.839**	372.265
	160	475.523	477.457	487.087	**488.171**	488.046	471.544
	200	577.483	605.232	622.474	629.974	**634.521**	598.366
AVG		390.011	395.548	410.871	414.863	**416.493**	392.962

Table 8: Performance comparisons of the heuristics for 21 orbits

Time Limits	Number of tasks	Heuristic 1 Obj	#F	Heuristic 2 Obj	#F	Heuristic 3 Obj	#F	Heuristic 4 Obj	#F	Heuristic 5 Obj	#F
1	120	246.761	10	251.586	10	232.257	10	**264.269**	10	247.014	10
	160	289.689	10	**309.153**	10	300.236	10	291.534	10	266.616	10
	200	340.449	8	**375.375**	8	364.202	10	301.453	10	284.286	10
10	120	250.776	10	256.141	10	239.850	10	**264.568**	10	256.571	10
	160	348.016	10	**362.915**	10	336.405	10	297.419	10	272.776	10
	200	414.974	10	**449.676**	10	416.366	10	305.165	10	293.553	10
60	120	254.532	10	259.896	10	241.476	10	**266.497**	10	257.301	10
	160	355.290	10	**370.867**	10	343.158	10	300.905	10	278.157	10
	200	433.487	10	**463.934**	10	435.598	10	308.597	10	296.174	10

Table 9: Performance comparisons of the heuristics for 42 orbits

Time Limits	Number of tasks	Heuristic 1 Obj	#F	Heuristic 2 Obj	#F	Heuristic 3 Obj	#F	Heuristic 4 Obj	#F	Heuristic 5 Obj	#F
1	120	396.169	10	403.608	10	387.431	10	**424.069**	10	360.123	10
	160	450.790	8	476.767	8	464.958	10	**480.738**	10	364.216	10
	200	-	0	-	0	**611.007**	10	590.293	10	443.651	10
10	120	424.383	10	431.473	10	409.055	10	**432.770**	10	367.762	10
	160	524.721	10	**552.304**	10	521.883	10	500.973	10	401.163	10
	200	637.603	10	**692.670**	10	660.999	10	606.546	10	451.985	10
60	120	433.018	10	**440.953**	10	416.056	10	434.317	10	372.956	10
	160	537.448	10	**564.523**	10	530.659	10	502.300	10	409.139	10
	200	663.211	10	**714.795**	10	676.523	10	610.697	10	457.794	10

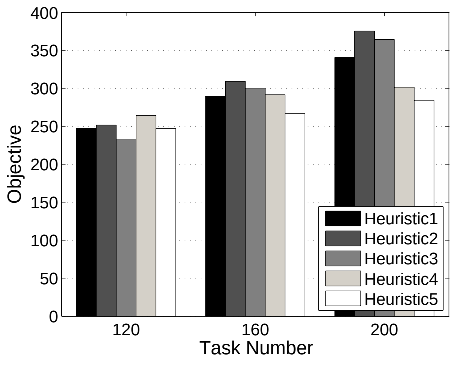

Figure 7: Comparison results for 21 orbits with 1 second

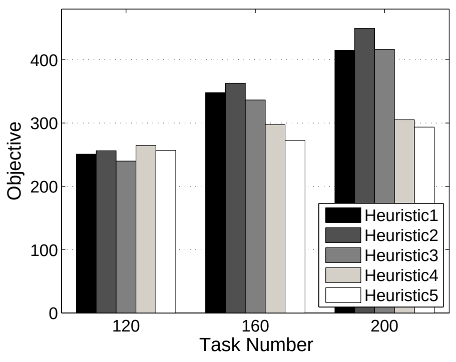

Figure 8: Comparison results for 21 orbits with 10 seconds

Table 9 and Figures 10-12 reveal the comparison results for the problems of 42 orbits, from which we can also conclude that for fewer tasks or a small time limit, **Heuristic 4** is the best algorithm. However, for more tasks (160 or 200) and larger time limits (10 or 60 seconds), **Heuristic 2** will be the best option. In addition, similarly to the previous experiments, for small time limits and large-scale problems (1 second for 160 or 200 tasks), **Heuristic 1** and **Heuristic 2** will fail to get feasible solutions for some instances. Comparing **Heuristic 3** and **Heuristic 4**, for less tasks, **Heuristic 4** performs better, and for more tasks, **Heuristic 3** performs better. Unfortunately, **Heuristic 5** always performs worst compared with others.

6. Conclusions and future work

In this paper, considering the uncertainties of clouds, we formulated the presence of clouds as stochastic events, and then investigated the scheduling of multiple EOSs. Due to the uncertainties of clouds, a task that is scheduled to multiple resources will be completed at a higher probability than if scheduled to one resource only. Hence, we took into account scheduling each task to multiple resources and formulated the problem with a novel mathematical model. First of all, we suggested an exact algorithm to solve the problem optimally, in which we reformulated the problem as a master problem and multiple subproblems. For each subproblem, a labeling-based dynamic programming algorithm was suggested to get all feasible solutions. On the basis of the solutions of the subproblems, we enumerated all feasible solutions of the master problem and selected the optimal solution. Due to its large space complexity, the proposed exact algorithm mostly fails to solve large-scale problems that are more practical. To overcome this drawback, we also designed five heuristic algorithms to solve the large-scale problems.

From the simulation experiments, we verified the superiority of our robust model compared with the traditional model that allocates each task to only one orbit. In addition, we validated that

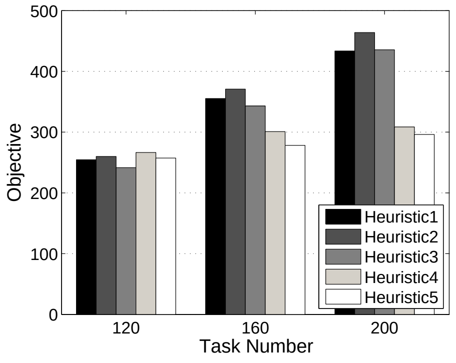

Figure 9: Comparison results for 21 orbits with 60 second

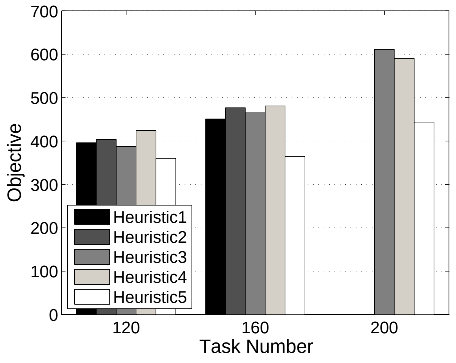

Figure 10: Comparison results for 42 orbits with 1 second

for small size problems, both the exact and heuristics can be adopted for solving the instances efficiently, and the heuristics can get feasible solutions that are very close to the optimal ones. Finally, for large-scale problems, we compare the performances of the different heuristics. From the comparison results, we conclude that for not too many tasks or small time limits, **Heuristic 4** performs best and **Heuristic 2** will be the best for larger problems and larger time limits.

In the future, we will consider the scheduling of agile EOSs under uncertainties. Different from the non-agile satellites in this study, the agile satellites do not only have the maneuverability of slewing, but also the maneuverability of pitching, along with the orbit. Hence, the satellite will have a longer time window for observation. Consequently, the scheduling will have more freedom, and we need not only to allocate the tasks to the orbits, but also to decide the starting and finishing times. Furthermore, the sequence of task executions will also not be fixed, as for any two tasks i and j, it is now possible to start the execution of task i before that of task j, or the execution of task j before that of task i, which obviously will make the problem more complicated. In addition, we will also take into account the probability of each possible outcome, formulating the problem as a sequential decision problem, and adopt a multistage stochastic programming approach to solve the problem.

Acknowledgments

This research has been supported by the China Scholarship Council. We would like to acknowledge the China Scholarship Council for the financial support and the Research Center for Operations Management of the KU Leuven for providing a visiting research period to Jianjiang Wang. This research was also supported by the National Natural Science Foundation of China under Grants No. 61104180, and No. 71271213.

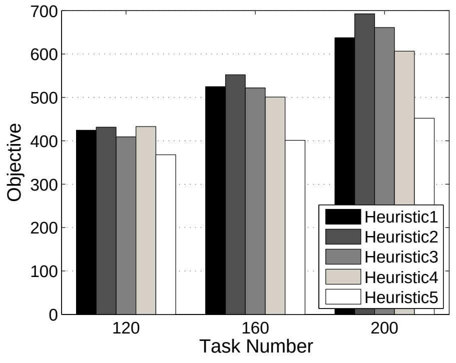

Figure 11: Comparison results for 42 orbits with 10 seconds

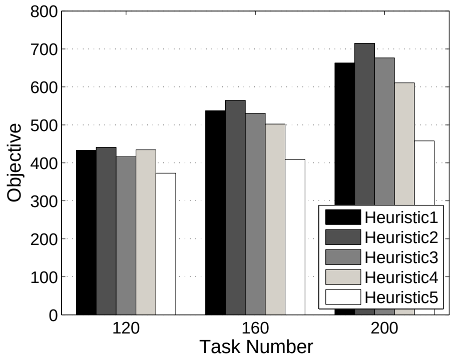

Figure 12: Comparison results for 42 orbits with 60 seconds

References

[1] Agnése JC, Bensana E. Exact and approximate methods for the daily management of an earth observation satellite. In: Proceeding of the fifth ESA workshop on aritificial intelligence and knowledge based systems for space; 1995.

[2] Barbulescu, L, Watson JP, Whitley LD, Howe AE. Scheduling space-ground communications for the air force satellite control network. J Scheduling, 2004; 7(1): 7-34.

[3] Beaumet G, Verfaillie G, Charmeau MC. Feasibility of autonomous decision making on board an agile earth-observing satellite.Comput Intell 2011; 27(1): 123-139.

[4] Benoist T, Rottembourg B. Upper bounds for revenue maximization in a satellite scheduling problem. 4OR-Q J Oper Res 2004; 2(3): 235-249.

[5] Bensana E, Verfaillie G, Agnese JC, Bataille N, Blumestein D. Exact and inexact methods for the daily management of an earth observation satellite. In: Proceeding of the international symposium on space mission operations and ground data systems; 1996. 4: 507-514.

[6] Bensana E, Verfaillie G, Michelon-Edery C, Bataille N. Dealing with uncertainty when managing an earth observation satellite. In: Proceeding of the fifth international symposium on artificial intelligence, robotics and automation in space (ISAIRAS'99); 1999, p.205-210.

[7] Bianchessi N, Piuri V, Righini G, Roveri M, Laneve G, Zigrino A. An optimization approach to the planning of earth observing satellites. In: Proceeding of the fourth international workshop on planning and scheduling for space (IWPSS'04); 2004.

[8] Bianchessi N, Righini G. A mathematical programming algorithm for planning and scheduling an earth observing SAR constellation. In: Proceeding of the fifth international workshop on planning and scheduling for space (IWPSS'06); 2006.

[9] 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. Eur J Oper Res 2007; 177(2): 750-762.

[10] Bianchessi N, Righini G. Planning and scheduling algorithms for the COSMO-SkyMed constellation. Aerosp Sci Technol 2008; 12(7): 535-544.

[11] Cordeau J, Laporte G. Maximizing the value of an earth observation satellite orbit. J Oper Res Soc 2005; 56(8): 962-968.

[12] Frank J, Jónsson A, Morris R, Smith DE. Planning and scheduling for fleets of earth observing satellites. In: Proceeding of the sixth international symposium on artificial intelligence, robotics, automation and space (ISAIRAS'01); 2001.

[13] Gabrel V, Moulet A, Murat C, Paschos VT. A new single model and derived algorithms for the satellite shot planning problem using graph theory concepts. Ann Oper Res 1997; 69: 115-134.

[14] Gabrel V, Vanderpooten D. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. Eur J Oper Res 2002; 139(3): 533-542.

[15] Gabrel V, Murat C. Mathematical programming for earth observation satellite mission planning. In: Ciriani TA, Fasano G, Gliozzi S, Tadei R, editors. Operations research in space and air, Springer; 2003, p.103-122.

[16] Gabrel V. Strengthened 0-1 linear formulation for the daily satellite mission planning. J Comb Optim 2006; 11(3): 341-346.

[17] Globus A, Crawford J, Lohn J, Morris R. Scheduling earth observing fleets using evolutionary algorithms: Problem description and approach. In: Proceedding of the third international NASA workshop on planning and scheduling for space; 2002.

[18] Globus A, Crawford J, Lohn J, Pryor A. A comparison of techniques for scheduling fleets of earth-observing. J Oper Res Soc 2003; 56(8): 962-968.

[19] Habet D, Vasquez M. Solving the selecting and scheduling satellite photographs problem with a consistent neighborhood heuristic. In: Proceeding of the sixteenth IEEE international conference on tools with artificial intelligence (ICTAI'04); 2004.

[20] Habet D. Tabu search to solve real-life combinatorial optimization problems: A case of study. In: Abraham A, Hassanien AE, Carvalho AP, editors. Foundations of Computational Intelligence, Springer 2009; 3: 129-151.

[21] Habet D, Vasquez M, Vimont Y. Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite. Comput Optim Appl 2010; 47(2): 307-333.

[22] Hall NG, Magazine MJ. Maximizing the value of a space mission. Eur J Oper Res 1994; 78(2): 224-241.

[23] Han S, Beak S, Cho K, Lee D, Kim H. Satellite mission scheduling using genetic algorithm. In: Proceeding of the international conference on instrumentation, control and information technology (SICE'08); 2008.

[24] Lemaître, M, Verfaillie G, Jouhaud F, Lachiver JM, Bataille N. How to manage the new generation of agile earth observation satellites. In: Proceeding of the international symposium on artificial intelligence, robotics and automation in space (ISAIRAS'00); 2000.

[25] Lemaître, M, Verfaillie G, Jouhaud F, Lachiver JM, Bataille N. Selecting and scheduling observations of agile satellites. Aerosp Sci Technol 2002; 6(5): 367-381.

[26] Li J, Li J, Chen H, Jing N. A data transmission scheduling algorithm for rapid-response earth-observing operations. Chinese J Aeronaut 2014; doi: http://dx.doi.org/10.1016/j.cja.2014.02.014.

[27] Li Y, Wang R, Xu M. Rescheduling of observing spacecraft using fuzzy neural network and ant colony algorithm. Chinese J Aeronaut 2014; doi: http://dx.doi.org/10.1016/j.cja.2014.04.027.

[28] Liao D, Tang Y. Satellite imaging order scheduling with stochastic weather condition forecast. In: Proceeding of the IEEE international conference on systems, man and cybernetics; 2005. 3: 2524-2529.

[29] Liao D, Tang Y. Imaging order scheduling of an earth observation satellite. IEEE Trans Syst Man Cy C 2007; 37(5): 794-802.

[30] Lin W, Liu C, Liao D, Lee Y. Daily imaging scheduling of an earth observation satellite. In: Prodeeding of the IEEE international conference on systems, man and cybernetics; 2003. 2: 1886 - 1891.

[31] Lin W, Liao D. A tabu search algorithm for satellite imaging scheduling. In: Prodeeding of the IEEE international conference on systems, man and cybernetics; 2004. 2: 1601-1606.

[32] Lin W, Chang S. Hybrid algorithms for satellite imaging scheduling. In: Proceeding of the IEEE international conference on systems, man and cybernetics; 2005. 3: 2518-2523.

[33] Lin W, Liao D, Liu C, Lee Y. Daily Imaging Scheduling of an Earth Observation Satellite. IEEE Trans Syst Man Cy A 2005; 35(2): 213-223.

[34] Marinelli F, Salvatore N, Rossi F, Smriglio S. A lagrangian heuristic for satellite range scheduling with resource constraints. Comput Oper Res 2011; 38(11): 1572-1583.

[35] Pralet C, Verfaillie G, Maillard A, et al. Satellite data download management with uncertainty about the generated volumes. In: Proceeding of the twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS'14); 2014.

[36] Pemberton JC, Greenwald LG. On the need for dynamic scheduling of imaging satellites. In: Proceeding of the American society for photographing and remote sensing (ASPRS'02); 2002.

[37] Sarkheyli A, Vaghei BG, Bagheri A. New tabu search heuristic in scheduling earth observation satellites. In: Proceeding of the second international conference on software technology and engineering (ICSTE'10); 2010.

[38] Soma P, Venkateswarlu S, Santhalakshmi S, Bagchi T, Kumar S. Multi-satellite scheduling using genetic algorithms. In: Proceeding of the eighth international conference on space operations (SpaceOps'04); 2004.

[39] Tangpattanakul P, Jozefowiez N, Lopez P. Multi-objective optimization for selecting and scheduling observations. In: Coello CA, Cutello V, Deb K, Forrest S, Nicosia G, Pavone M, editors. Lecture Notes in Computer Science, Springer; 2012.

[40] Tangpattanakul P, Jozefowiez N, Lopez P. A multi- objective local search heuristic for scheduling Earth observations taken by an agile satellite. Eur J Oper Res 2015; doi: 10.1016/j.ejor.2015.03.011

[41] Vasquez M, Hao J. A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. Comput Optim Appl 2001; 20(2): 137-157.

[42] Vasquez M, Hao J. Upper bounds for the SPOT 5 daily photograph scheduling problem. J Comb Optim 2003; 7(1): 87-103.

[43] Verfaillie G, Schiex T. Solution reuse in dynamic constraint satisfaction problems. In: Proceeding of the twelfth national conference on artificial intelligence (AAAI'94); 1994. p.307-312.

[44] Verfaillie G, Lemaitre M, Schiex T. Russian doll search for solving constraint optimization problems. In: Proceeding of the thirteenth national conference on Artificial intelligence (AAAI'96); 1996. p.181-187.

[45] Wang H, Xu M, Wang R, Li Y. Scheduling earth observing satellites with hybrid ant colony optimization algorithm. In: Proceeding of the international conference on artificial intelligence and computational intelligence (AICI'09); 2009.

[46] Wang J, Zhu X, Qiu D, Yang LT. Dynamic scheduling for emergency tasks on distributed imaging satellites with task merging. IEEE Trans Parall Distr 2014; 25(9): 2275-2285.

[47] Wang J, Zhu X, Yang LT, Zhu J, Ma M. Towards dynamic real-time scheduling for multiple earth observation satellites. J Comput Syst Sci 2014; doi: 10.1016/j.jcss.2014.06.016

[48] Wang P, Reinelt G, Gao P, Tan Y. A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation. Comput Ind Eng 2011; 61(2): 322-335.

[49] Wolfe J, Stephen SE. Three scheduling algorithms applied to the earth observing systems domain. Manag Sci 2000; 46(1): 148-168.

[50] Wu G, Liu J, Ma M, Qiu D. A two-phase scheduling method with the consideration of task clustering for earth observing satellites. Comput Oper Res 2013; 40(7): 1884-1894.

[51] Xhafa F, Herrero X, Barolli A, Takizawa, M. A Comparison Study on Meta-Heuristics for Ground Station Scheduling Problem. In: Proceeding of the seventeenth International Conference on Network-Based Information Systems (NBIS'14); 2014: 172-179.

[52] Zufferey N, Amstutz P, Giaccari P. Graph colouring approaches for a satellite range scheduling problem. J Scheduling 2008; 11(4): 263-277.

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be