Computers and Operations Research 104 (2019) 74–89

Contents lists available at ScienceDirect

Computers and Operations Research

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

A branch and price algorithm for EOS constellation imaging and downloading integrated scheduling problem

Xiaoxuan Hu[a,b,*], Waiming Zhu[a,b], Bo An[c], Peng Jin[a,b], Wei Xia[a,b]

[a] School of Management, Hefei University of Technology, Hefei, China
[b] Key Laboratory of Process Optimization and Intelligent Decision-making, Ministry of Education, Hefei, China
[c] School of Computer Science and Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 26 February 2018
Revised 26 October 2018
Accepted 5 December 2018
Available online 6 December 2018

Keywords:
EOS constellation
Integrated scheduling
Branch and price
Primal dual column generation

ABSTRACT

Earth observation satellite (EOS) constellation imaging and downloading integrated scheduling (EOSCIDIS) is a challenging optimization problem with several engineering applications. Given a large number of imaging requests and a constellation consisting of a group of EOSs, a workable schedule must be developed for each EOS to allow for imaging and downloading in the most efficient manner given a certain scheduling horizon. No studies have proposed exact algorithms for the EOSCIDIS problem, although it has considerable potential for applications. In this paper, we developed a branch and price (B&P) algorithm for the EOSCIDIS problem. First, we decomposed the problem by extracting the sub-structure of the problem, and the decomposition resulted in a master problem and multiple pricing problems. Solving the linear relaxed master problem with column generation (CG) method provides a very tight upper bounds for the primal problem. We then embedded the CG process into a branch and bound (B&B) framework to find optimal integer solutions. In order to accelerate the convergence of the algorithm, we developed a heuristic utilizing the generated columns to construct feasible integer solutions to prune B&P tree branches earlier. We tested the B&P algorithm on highly-simulated instances that are constructed according to the Chinese Gaofen-1 satellite and 3 types of random instances which correspond to short, medium and long term scheduling respectively. Both results show that our method could find high-quality solutions with optimality gap less than 10% for most instances using reasonable computational costs .

1. Introduction

Earth observation satellites (EOSs) are launched for the primary purpose of collecting surface images from orbit. EOS imagery has recently been used in agriculture, map making, land surveys, and disaster warning and recovery (Madry, 2013). An EOS constellation is an imaging system with a certain geometrical shape, composed of a group of EOSs providing coordinated ground coverage. These systems are beneficial as they reduce revisit time, the duration between repeated observations of the same site. Multiple EOS constellations have been commissioned in recent years, including the French Pléiades (Lemaître et al., 2002) and Italy‘s COSMO-SkyMed (Bianchessi and Righini, 2008). Presently, several countries have committed to enhancing EOS constellation infrastructure, particularly China, which will continue to increase investment in this field.

An EOS will first image a series of ground-based sites and then store the acquired image data to on-board memory (i.e., high performance hard disks). These data are then downloaded to ground stations during radio communications, freeing storage resources for subsequent observations.

This process requires manual intervention, in which commands are computed according to workable imaging and downloading schedules. EOS constellation imaging and downloading integrated scheduling (EOSCIDIS) is an optimization problem that has become of interest as the number of constellations in operation has increased. Given a large number of imaging requests and limited imaging and downloading resources (EOS constellation), an optimal imaging and downloading schedule would allow for imaging and downloading in the most efficient manner given a certain scheduling horizon. The EOSCIDIS problem is a generalization of the EOS scheduling (EOSS) problem. The primary difference between them is that EOSCIDIS takes integrated consideration of imaging and downloading operations while EOSS focuses on imaging schedules

* Corresponding author.
E-mail addresses: xiaoxuanhu@hfut.edu.cn (X. Hu), zhuwaiming@mail.hfut.edu.cn (W. Zhu), boan@ntu.edu.sg (B. An), jinpeng@hfut.edu.cn (P. Jin), xiawei@hfut.edu.cn (W. Xia).

https://doi.org/10.1016/j.cor.2018.12.007
0305-0548/© 2018 Elsevier Ltd. All rights reserved.

only (Malladi et al., 2016). As a result, the EOSCIDIS problem is more complex.

EOSCIDIS is a high-dimensional optimization problem that is closely related to the assignment (Jarrah et al., 2014), knapsack (Wang et al., 2013; Furini et al., 2015), inventory (Lewin, 2012; Song et al., 2016), and machine scheduling (Janiak et al., 2015) problems. It is characterized by complex constraints and a large-scale solution space. As the imaging and downloading operations influence each other, their coupled relationship must be considered in addition to action constraints and resource limitations. Real world scheduling problems also involve a large number of imaging requests and opportunities (Bensana et al., 1999). Modeling the problem with accurate mathematical formulation and finding a solution with acceptable computational costs is challenging. In fact, the traditional EOSS problem is known to be NP-complete (Gabrel et al., 1997).

While EOSS problems have been widely studied, there is far less research available on the EOSCIDIS problem. Prior studies have used dynamic programming (DP) (Gabrel and Vanderpooten, 2002; Lemaître et al., 2002), branch and bound (B&B) (Gabrel et al., 1997; Verfaillie et al., 1996), column generation (CG) (Gabrel and Murat, 2003), branch and cut (B&C) (Wang et al., 2016), and several other inexact algorithms for various EOSS problems. However, an exact algorithm has yet to be developed for the EOSCIDIS problem. In this paper, we conduct a study on the application of exact algorithms to EOS constellations and propose an original branch and price (B&P) algorithm to solve the presented problem. This study was inspired by engineering applications developed by Chinese Gaofen satellites.

The primary contributions of the paper are as follows. (1) We found the problem to be decomposable and developed a period splitting method to divide the overall scheduling horizon into several time periods. As a result, a primal mixed integer linear program (MILP) formulation was proposed. (2) We showed that the primal problem was suitable for reformulation using the Dantzig–Wolfe (DW) decomposition method (Dantzig and Wolfe, 1960). Thus, we employed the DW method to separate the primal problem into a master problem and multiple pricing problems. Our DW decomposition causes a tighter upper bound for the primal problem. (3) To compute the optimal integer solutions for the primal problem, we first used the primal dual column generation method (PDCGM) (Gondzio et al., 2013; 2016) to produce the best solution to the linear relaxed master problem. Within the CG process, we developed an efficient DP algorithm to solve the pricing problems. (4) We then embedded the CG process into a B&B framework and developed a B&P algorithm to compute optimal integer solutions. (5) To accelerate the convergence of the B&P algorithm, we also developed a heuristic embedded into the B&B framework. The heuristic was evolved by utilizing the generated columns to find good integer solutions in the B&B process.

The remainder of this paper is organized as follows. Section 2 gives a brief review of related works. Section 3 presents a description of the EOSCIDIS problem. Section 4 introduces the period splitting method and provides a primal MILP formulation of the problem. Section 5 provides a detailed introduction of the B&P algorithm. Section 6 details computational experiences for both simulated and randomly generated instances. Conclusions are presented in Section 7.

2. Literature review

Several previous studies have examined the EOSS problem from different perspectives, such as the French SPOT satellites (Bensana et al., 1999; Gabrel, 2006; Mansour and Dessouky, 2010; Verfaillie et al., 1996), the Pléiades constellation (Lemaître et al., 2002; Tangpattanakul et al., 2015a; 2015b), Italy‘s COSMO-SkyMed constellation (Bianchessi and Righini, 2008), and the Chinese Huanjing satellites (Liu et al., 2014; Wang et al., 2011). Both inexact and exact algorithms have been developed to solve the EOSS problem.

Inexact algorithms used in this field have included heuristic and meta-heuristic methods. One of the most widely used heuristic methods is the greedy constructive algorithm (Wang et al., 2011). The primary benefit of this algorithm is its low computational cost, while its primary disadvantage is its tendency to produce suboptimal results. Neighbor search approaches (Globus et al., 2004; Liu et al., 2017) are one of commonly employed meta-heuristics. One critical issue with these types of algorithms is the use of iterations and the devise of neighborhood structures in the search space. The most prominent neighbor search algorithms employed in EOSS are the simulated annealing (SA) algorithm (Globus et al., 2004; Peng et al., 2011) and tabu searching (Sarkheyli et al., 2013; Wang et al., 2015). While these algorithms often provide a satisfactory solution, the computational cost is generally higher than heuristics. Artificial intelligence (AI) algorithms are another widely used type of meta-heuristic. They are typically derived from natural laws and used to solve complex combinatorial optimization problems. The most prominent AI algorithms employed in EOSS are the genetic algorithm (Tangpattanakul et al., 2015a; Waiming et al., 2017; Wolfe and Sorensen, 2000), ant colony optimization (ACO) (Liu et al., 2014; Wu et al., 2013; Xu et al., 2016; Zhang et al., 2014), and particle swarm optimization (PSO) (Zhang et al., 2013). However, due to the particularity of the EOSS solution space, AI algorithms require careful adjustment to achieve efficient optimization performance. In addition, several studies have combined AI algorithms and local searches (Liu et al., 2014; Wu et al., 2013). These algorithms are usually functional but typically suffer from premature convergence, owing to the optimization mechanism of AI algorithms. Although they are widely used, inexact algorithms have certain weaknesses. They are not guaranteed to produce high-quality solutions on a consistent basis and struggle to provide good optimality evaluation rules (i.e., tight upper bounds), to evaluate the quality of the resulting solution. Recently, a novel matheuristic has been suggested to solve the problem (Malladi et al., 2017). This matheuristic is actually a hybrid of meta-heuristic and mathematical programming solvers. The method is very appealing and the achieved test results are encouraging.

The most commonly used exact algorithms for solving EOSS problems include graph theory-based DP (Gabrel and Vanderpooten, 2002; Lemaître et al., 2002), B&B (Gabrel et al., 1997; Verfaillie et al., 1996), CG (Gabrel and Murat, 2003), and B&C (Wang et al., 2016).

Gabrel et al. (1997), Gabrel and Vanderpooten (2002), Gabrel and Murat (2003), and Gabrel (2006) employed a mathematical programming method to develop exact algorithms for solving EOSS problems. Both single- and multiple-orbit EOS scheduling problems have been studied (Gabrel et al., 1997). In these applications, the EOS could pitch its camera which requires the imaging operation to be flexible in an imaging time window. Gabrel proposed two methods to overcome this issue. The first was converting a continuous optimization problem into discrete problems by discretizing the imaging start intervals. In this case, a polynomial time algorithm existed for the single-orbit EOSS problem and efficient algorithms could be developed (based on graph theory) to solve the multiple EOSS problem. The second approach involved solving a continuous optimization problem directly. In this case, Gabrel developed an exact B&B method for small-scale and approximated a heuristic for large-scale single-orbit EOSS instances, respectively. More realistic problems have since been investigated (Gabrel and Vanderpooten, 2002), such as a single EOS daily management problem with multiple optimization objectives (i.e., demand satisfaction, priorities, and satellite use). A two-stage method was developed to solve this problem.

First, all Pareto optimal paths were acquired using an efficient label-setting shortest path algorithm. Interactive path selection was then carried out to obtain the ultimate solution based on a tradeoff among the specified criteria. Gabrel also introduced the use of a mathematical programming method to solve the EOSS problem in detail (Gabrel and Murat, 2003). Beginning with a simple case, Gabrel introduced a method for modeling the problem using graph theory. A series of formulations for various criteria were provided. Finally, extending the SPOT 5 scheduling problem, a CG method was used to calculate tight upper bounds. The validation of this paper demonstrates the linear relaxation problem of re-formulation (obtained by DW decomposition) is an extreme tight upper bound for the primal problem. This is consistent with our investigations in this paper. Gabrel has demonstrated the viability of mathematical programming methods for solving the EOSS problem, most of which were based on graph theory. However, these and previous studies have not considered downloading operations.

Verfaillie et al. (1996) developed an efficient B&B algorithm referred to as the Russian doll search (RDS) for solving the SPOT 5 EOSS problem. The main idea of RDS is replacing one search with multiple successive searches on nested sub-problems in a traditional B&B procedure. This strategy results in better bounds and thereby plenty of nodes that could be pruned much earlier. The method is extremely simple but very powerful. Validation studies have shown that RDS outperforms depth-first B&B in both optimality and efficiency when solving SPOT 5 EOSS problems. Lemaître et al. (2002) developed a DP algorithm for solving a modified agile EOSS problem. The DP algorithm is a polynomial algorithm originating from the standard shortest-path problem. However, this DP method is not optimal for the primal problem. And this also indicates that a decomposition maybe efficient. Vasquez and Hao (2003) studied the upper bounds of the SPOT 5 daily scheduling problem, based on a "logic-constrained" knapsack formulation. The author showed that neither linear relaxation nor logic constraint relaxation of the problem are tight upper bounds for the original problem. As a result, the author proposed a partition-based approach to calculate the tight upper bounds. Inspired by "divide and conquer", the overall problem was first divided into several smaller sub-problems, which could be optimally solved using an iterative enumerative algorithm. The sum of all sub-problem optimal values serves as a tight upper bound for the original problem. The authors developed optimization criteria for improved partitioning and a Tabu search algorithm to perform the partitioning. Habet et al. (2010) introduced a bounding method for a single agile EOSS problem. Some constraints were relaxed and the objective function was linearized. A simpler problem with an optimal DP algorithm was obtained. Wang et al. (2016) studied the EOSS problem in the presence of cloud-based uncertainties. As a result, a novel assignment formulation was developedand a sample approximation method was used to convert the problem into an integer linear program. The authors developed a B&C algorithm to solve the problem.

In the researches on developing exact algorithm for EOSS problems, some of these studies did not consider storage capacity constraints (Gabrel et al., 1997; Gabrel and Vanderpooten, 2002; Habet et al., 2010), though some did (Gabrel, 2006; Gabrel and Murat, 2003). Some of them even considered energy limitations (Wang et al., 2016). However, none of them considered downloading operations (Karapetyan et al., 2015) that significantly affect EOS storage resources. This is significant because downloading and imaging are highly-coupled operations. As such, it is imperative to study imaging and downloading integrated scheduling. This problem creates new challenges for employing mathematical programming methods as it leads to more complicated formulations. As a result, the "divide and conquer" model (Vasquez and Hao, 2003) could be beneficial in this setting. Gabrel and Murat (2003) also suggested using DW decomposition (Dantzig and Wolfe, 1960).

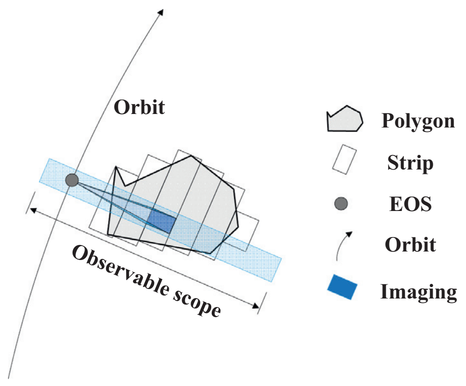

Fig. 1. The polygon, strip, and imaging configuration.

3. Problem description

In this section, we discuss the EOSCIDIS problem in detail. First, we introduce the imaging requests and the constellation resource respectively. Then, we provide a detailed definition of the problem.

3.1. Imaging request

Multiple imaging requests are submitted by customers from various departments. As shown in Fig. 1, each imaging request is an earth region whose shape is a polygon. Because the imaging width of the camera carried on an EOS is limited, each polygon request (abbreviated as 'polygon') must be split into strip regions (abbreviated as 'strips'). The width of each strip is equal to the imaging width of the camera onboard the EOS (Lemaître et al., 2002). T represents the strip set where i and $r \in T$ denote the strip index.

We only consider scheduling with visible light imaging requests. In visible light imaging, each EOS has only one imaging direction for each polygon. In this paper, the EOSs in a constellation share an orbit and hence have the same imaging direction. As a result, we need to split each polygon, according to the imaging direction, only once.

Each polygon and its corresponding strips are associated with a profit, which is defined in advance. The profit of a strip is calculated using the profit of its corresponding polygon. If strip i is split from polygon g, the profit of strip i is calculated as follows:

$$v_i = a_i / a_0 \times v_0 \tag{1}$$

where the profit and area of polygon g are denoted as v_0 and a_0, the profit and *valid area* of strip i are denoted as v_i and a_i, respectively. The *valid area* is the intersection area between polygon g and strip i.

Because each polygon is split only once, there is no overlap between any two strips. Therefore, the total profit of imaged strips is the algebraic sum of profits for all imaged strips. (Different from the work of Cordeau and Laporte (2005), this paper does not consider the piecewise linear relationship between the total profit and the observed area of a polygon in an schedule.) This characteristic is particularly important because the resulting calculation of the total profit is linear.

Recording images of strips consumes storage resources of an EOS. The storage consumption of each strip depends linearly on the length of the strip. The term c_i represents the storage consumption volume of strip i. A longer strip requires larger values of c_i. These data can be calculated in advance.

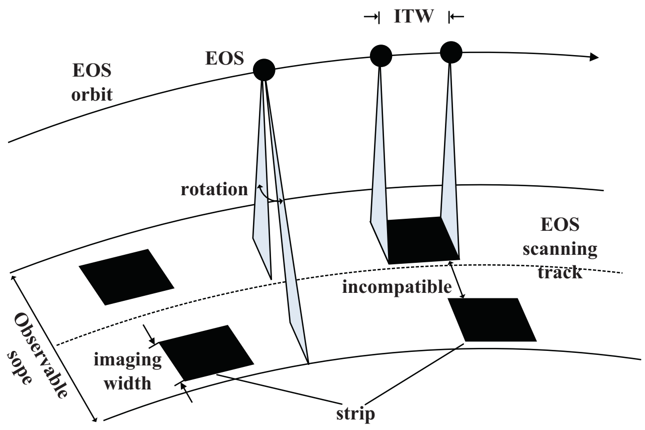

Fig. 2. The rotation, ITW and incompatibility relationship.

3.2. Constellation resource

There is a sun synchronous constellation which consists of several EOSs. In this paper, the orbits of the EOSs in the constellation are extremely similar except for the *local time of descending node* (LTDN). The EOSs within a constellation share an orbit. This type of constellation can revisit an Earth region more quickly. The Chinese Gaofen-1 and Gaofen-2 satellites are such a constellation although they do not share the same orbit. Throughout this study, S denotes the EOS set and $j \in S$ denotes an EOS.

Each EOS carries a visible light camera that can scan and image Earth surface as the EOS flies over the sunlit side of the Earth. When scanning, the camera can roll; namely, it can perform a vertical rotation along the scanning direction. Therefore, strips close to the EOS scanning tack have the chances of being imaged. The observable scope of the EOS is a long rectanglar area, as shown in Fig. 2.

There are two basic limitations of imaging actions for each EOS, also shown in Fig. 2. First, an EOS can only perform imaging operations over some relative short time intervals when the EOS flies above the strip. These intervals are called imaging opportunities or imaging time windows (ITWs). The availability of an ITW in an orbit depends on the relative spatial position between the EOS and the strip. These data can be computed by related software in advance. Multiple ITWs may exist between an EOS and a strip in a given scheduling horizon. W^j denotes the ITW set for an EOS j with a and $b \in W^j$ representing the ITW index. Each ITW corresponds to a specific strip where $L^{ij} \subseteq W^j$ denotes the index set corresponding to strip i on EOS j (i.e., if ITW $a \in L^{ij}$ then a corresponds to strip i). There is a schedule in which ITWs were adopted. If an ITW was adopted in a schedule, the EOS would image the corresponding strip. Second, an EOS can only acquire strip images one by one, because each EOS carries only one visible light camera. Hence, when two ITWs appear at the same time for an EOS, or there is not enough transition time to image the two corresponding strips, it can only select a single ITW. The two ITWs are called incompatible or mutually exclusive, as shown in Fig. 2. If $a \in W^j$ and $b \in W^j$ are incompatible, we label (a, b) as an incompatible ITW pair. The term O^j represents the incompatible ITW pair set for EOS j. Therefore, composing an imaging schedule is partly an ITW selection problem. A good schedule selects critical ITWs in order to image the most profitable strips.

When an EOS transits over a ground station, a communication opportunity is available (Álvarez and Erwin, 2015). Generally, there could be several communication opportunities for an EOS to a ground station in a given scheduling horizon. Each communication opportunity is called downloading time window (DTW),

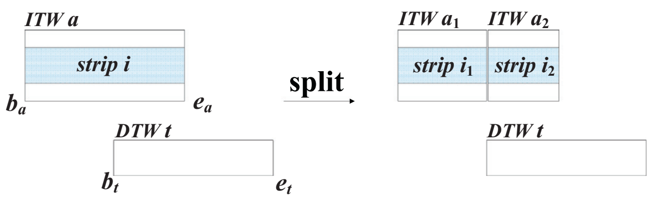

Fig. 3. Splitting strips according to the DTW.

where D_j is the DTW set for EOS j and $t \in D_j$ is the DTW index. Each communication opportunity has a finite downloading capacity denoted as o_{jt} , which depends on the length of DTW t. These data can also be calculated in advance. Once a communication opportunity arises, downloading operations begin immediately and continue until the end of the corresponding DTW. Hence, it is required that each downloading operation occupies the whole DTW regardless of how many images are downloaded. In other words, the beginning and ending time for each downloading operation are fixed and strictly equated to the time span of the corresponding DTW. EOSs can perform imaging and downloading operations simultaneously, as the camera and antenna work independently. Although the EOS can simultaneously perform imaging and downloading operations, it cannot download images that have not been taken. For example, if a download t starts at time b_t, it cannot download any image collected after b_t.

Acquired images are temporarily stored in the on-board storage medium. This is a recorder or hard disk with finite storage space (i.e., storage capacity q_j for EOS j). Cameras carried by new-generation EOSs provide high-resolution images, increasing the required storage capacity. Although storage technology is developing quickly, wide-field Earth images require significantly more storage resources. An EOS cannot continue imaging operations if storage limit is reached. As such, storage capacity becomes a resource limitation in the scheduling problem. Note that the storage resource is supplementary because the EOS could download images to ground stations if communication is available. Imaging operations increase while downloading operations decrease the volume of stored images. Downloading data through a downlink is much faster than accumulating data through imaging. What's more, in order to better utilize the EOSs resources, we must carry out futher splits on some strips, which is shown in Fig. 3. The reason is disccussed next. Strip images are compressed into a single file to be stored on the EOS. The image data cannot be downloaded before being completely imaged. However, if a strip is located close to a ground station, a tractable situation may arise as shown in Fig. 3. When download t begins at b_t, the image of strip i is fragmented if ITW a is adopted and b_t satisfies: $b_a < b_t < e_a$. To facilitate the issue, we split strip i into two sub-strips i_1 and i_2, if such a condition occurs.

Composing a daily or long-term schedule for an EOS constellation requires the imaging and downloading operations to be integrated. Such schedules are complex because imaging and downloading operations are highly coupled. On one hand, the download volume depends on the size of collected images. On the other hand, a download operation may influence the selection of ITWs because it can release corresponding storage resources for collecting new images.

3.3. Integrated scheduling problem

The integrated scheduling problem can be summarized as satisfying imaging requests using limited constellation resources in the most profitable manner. Specifically, we need to compose a schedule indicating the detailed arrangements of all imaging and downloading operations for each EOS. This includes the selection

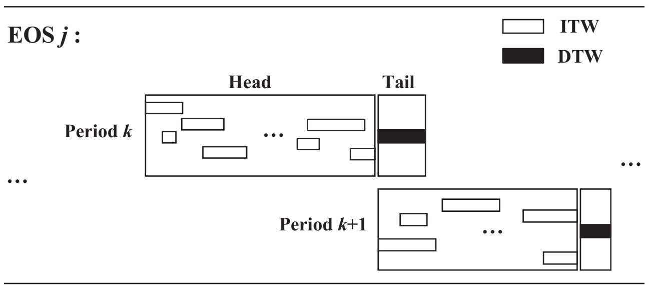

Fig. 4. A period definition for EOS *j*.

of ITWs and the determination of download size for each DTW. The objective of the problem is maximizing the total profit of all imaged strips.

To compose such schedule(s), some realistic constraints should be considered including: (1) Each strip must be imaged during its corresponding ITWs. (2) Each EOS can only image a single strip at a given time. (3) The amount of stored image data cannot exceed the storage capacity of an EOS at any time. (4) The downloaded data size cannot exceed the volume of data stored on-board. (5) The downloaded data size cannot exceed the download capacity of the downloading opportunity.

Besides, some reasonable assumptions are made for the problem in this paper. These assumptions are widely used in related literatures: (1) Each EOS carries only one visible light camera; (2) Energy is sufficient; (3) There is no stereo imaging requirement that needs to be imaged more than once.

4. Problem formulation

In this section, we model the problem with an MILP formulation. Before that, we introduce a periods splitting pre-process.

4.1. Periods definition and splitting method

The number of DTWs is significantly less than the number of ITWs (e.g., the number of download opportunities in simulated scheduling instances varying from 48–96 h does not exceed 200). Therefore, splitting the scheduling time horizon for each EOS into several relatively-short periods can reduce the number of variables and constraints used when modeling the problem. Based on this, we propose a pre-processing method, in which the entire scheduling time horizon is divided into several time periods.

Each period satisfies the following two conditions (see Fig. 4):

(1) Each period consists of a head and tail component. There are several ITWs but no DTWs in the head section and a DTW in the tail. In each period, the EOS performs a number of imaging operations but only one download operation.

(2) There exists (at most) one ITW for each strip in the head of a period and a one-to-one match between a strip and a period can be established. Otherwise, if there are more than two ITWs for a strip in a period, each ITW must be indexed. This is not beneficial for improving the formulation.

We developed a period splitting method based on two types of splitting: natural and dummy splitting.

Definition 1. The beginning of each DTW is used as the natural splitting time.

The critical time is the beginning of each DTW, when the EOS begins to download image data. At this time, there are no ITWs to span over and we can split the scheduling horizon. Hence, we use this time as the natural splitting time.

Split method: We create two periods for a given set of ITWs and a natural splitting time, as well as its corresponding DTW. We then place such ITWs, the end time of which is earlier than the natural splitting time, into the head of the first period. The others are placed into the head part of the second period. Finally, we place the DTW corresponding to the natural splitting time into the tail of the first period.

The periods obtained by natural splitting may not satisfy **Condition (2)**. Therefore, we establish a dummy download time window to split the periods further. The dummy splitting time corresponds to dummy DTWs.

Definition 2. Let B_i (B_r) and E_i (E_r) denote the beginning and end time of ITW i (r) respectively. For a given ITW r, if there is no ITW i satisfying $B_i < E_r < E_i$, then E_r is a dummy splitting time.

Split method: The splitting method in dummy splitting is similar to natural splitting. The only difference involves creating a virtual DTW with zero capacity for the first period. There is at least one dummy splitting time in an orbit for each EOS. This was observed from the actual situation. Therefore, we could split any period into sub-periods, the time span of which is strictly less than the time span of an orbit. As such, they must satisfy **Condition (2)**.

For each EOS, we first carry out the natural splitting, followed by a dummy splitting. This is the period splitting algorithm detailed in Algorithm 1.

Algorithm 1 period splitting algorithm.

```
Require: time windows list, $TL$;
Ensure: periods set $PR$;
1: find natural splitting set $ST$ from $TL$;
2: for $s = 1$; $s < |ST|$; $s++$ do
3:     perform natural splitting to get a new period $p$;
4:     put $p$ into $PR$;
5: end for
6: for $p = 1$; $p < |PR|$; $p++$ do
7:     while $p$ does not satisfy condition (2) do
8:         make a “suitable“ dummy splitting;
9:         // “suitable“ means the dummy splitting
10:        //can provide a new period $p'$ such that:
11:        //$p$ satisfy condition (2) and
12:        //$p$ has the most ITWs
13:        drop $p'$ from $p$, and put $p'$ into $PR$;
14:    end while
15: end for
```

4.2. An MILP formulation

Period splitting divides the whole scheduling horizon into multiple periods. Each has an independent sub-structure and all periods form the overall scheduling structure. Based on the period splitting, a primal program is formulated.

Let P^j be the period set of EOS j and $k \in P^j$ be the period index. $O^{jk} \subseteq O^j$ is the incompatible ITW pair set in period k on EOS j, where $O^j = \cup_{\forall k \in P^j} O^{jk}$.

Let Y: $|S| \times |P^j| \times |T| \to \{0, 1\}$ be the selection decision variables where $y_{ijk} \in Y$ indicates whether strip i was selected in period k on EOS j. The term $\vartheta_i(Y) = \sum_{\forall j \in S} \sum_{\forall k \in P^j} y_{ijk}$ indicates the number of times strip i was imaged. Therefore, the optimized objective is:

$$z^P = \max \sum_{\forall i \in T} \left[v_i \cdot \min\left(1, \vartheta_i(Y)\right) \right] \tag{2}$$

In order to transform (2) into a linear form, we provide an alternative objective function and complementary constraints:

$$z^P = \max \sum_{\forall i \in T} \left[v_i \cdot \vartheta_i(Y) \right] \tag{3}$$

$$\vartheta_i(Y) \le 1, \forall i \in T \tag{4}$$

Proposition 1. *If $Y^* \in \mathbb{Q}$ optimizes objective (3) where $\mathbb{Q} := \{Y|\vartheta_i(Y) \le 1, \forall i \in T\}$, then Y^* also optimizes (2).*

Proof. $\sum_{\forall i \in T}\left[v_i \cdot \min(1, \vartheta_i(Y))\right] \le \sum_{\forall i \in T}(v_i \cdot \vartheta_i(Y)) \le \sum_{\forall i \in T}(v_i \cdot \vartheta_i(Y^*)) = \sum_{\forall i \in T}\left[v_i \cdot \min(1, \vartheta_i(Y^*))\right]. Therefore, Y^* also optimizes (2).$ □

When the EOS encounters an incompatible ITW pair, it can only select a single ITW. This results in the following constraints (5).

$$y_{ijk} + y_{rjk} \le 1, \forall (i, r) \in O^{jk}, \forall j \in S, \forall k \in P^j \tag{5}$$

The size of stored images should not exceed the storage capacity of EOS j *whenever*. We used discrete stages instead of continuous scheduling horizon. And each period can be viewed as a stage. Let $D : |S| \times |P^j| \to [0, +\infty]$ be the downloading variables where $d_{jk} \in D$ indicates the volume of data downloading in the period k on EOS j. Therefore, in each stage the storage capacity constraint is:

$$\sum_{t=1}^{k}\sum_{\forall i \in T}(y_{ijt} \cdot c_i) - \sum_{t=1}^{k-1} d_{jt} \le q_j, \forall j \in S, \forall k \in P^j. \tag{6}$$

If the storage size in each stage is less than or equal to the storage capacity, the data size would not exceed the storage capacity.

The downloading capacity constraint says that the data downloading volume in each downloading operation should not exceed corresponding downloading capacity:

$$d_{jk} \le o_{jk}, \forall j \in S, \forall k \in P^j. \tag{7}$$

Constraint (8) regarding volume downloading indicates that data download volume should not exceed the amount of data which can be stored on the EOS. An extreme state occurs when the DTW has a large download capacity while the EOS has not stored any images; thus, nothing can be downloaded to the DTW.

$$-\sum_{t=1}^{k}\sum_{\forall i \in T}(y_{ijt} \cdot c_i) + \sum_{t=1}^{k} d_{jt} \le 0, \forall j \in S, \forall k \in P^j \tag{8}$$

Hence we get an MILP formulation called primal program: (3)–(8). Constraints (6)–(8) correspond to the number of periods, which does not exceed 200 in our experiments. However there were still too many constraints and variables to process (i.e., constraints (5) and variables y_{ijk}).

5. A branch and price algorithm

To solve the problem, We developed a B&P algorithm (Barnhart et al., 1998), in which a CG procedure based on DW decomposition (Dantzig and Wolfe, 1960) was embedded into a B&B framework (Morrison et al., 2016). In addition, a well substructure of the problem was obtained where the dynamically generated columns (i.e., solutions of the sub-problems) form the subschedules of the whole problem. We therefore develop an efficient heuristic also embedded into the B&B framework, utilizing the subschedules to construct efficient schedules (i.e., high quality solutions) to significantly prune B&P tree nodes.

5.1. Decomposition and re-formulation

The problem depicted by constraints (3) and (5) is actually an interval scheduling problem (Kolen et al., 2007), also called a weighted activity selection problem, for which an efficient dynamic algorithm is available (Lew and Mauch, 2006). It can be observed that a subproblem can be solved efficiently; however, the whole problem is difficult to solve. This strongly suggests the use of decomposition to break up the overall problem into pricing problem(s) and a master problem.

DW decomposition (Dantzig and Wolfe, 1960) is one of the most popular decomposition methods in (mixed) integer programming. One master problem (MP) and multiple pricing problems, namely price problems, are obtained if the primal program has a block diagonal structure. Due to the MP obtained using DW decomposition usually has a large number of columns, a restricted MP is iteratively solved and the columns are implicitly enumerated. This process is called column generation (Desrosiers and Lübbecke, 2005).

5.1.1. Master problem

Let $\Re^{jk}$ be the feasible region defined by (5) and $\sigma_{jku} \in \Re^{jk}$ be the uth solution for the period k on EOS j where $u \in U^{jk} = \{1, 2, \cdots, |\Re^{jk}|\}$ is the solution index. We use ε_{ijku} to indicate whether strip i was selected in σ_{jku}. We establish new variables λ_{jku} indicating whether σ_{jku} was adopted in the schedule. We then obtain a substitution: $y_{ijk} = \sum_{\forall u \in U^{jk}}(\varepsilon_{ijku} \cdot \lambda_{jku})$. Defining $c_{jku} = \sum_{\forall i \in T}(c_i \cdot \varepsilon_{ijku})$ and $w_{jku} = \sum_{\forall i \in T}(p_i \cdot \varepsilon_{ijku})$, we carry these three terms into primal program to replace corresponding parts. Constraint (5) is removed while convexity constraints are added. Then we obtained the master problem as follows.

$$z^{MP} = \max \sum_{\forall j \in S}\sum_{\forall k \in P^j}\sum_{\forall u \in U^{jk}}(w_{jku} \cdot \lambda_{jku}) \tag{9}$$

$$\sum_{\forall j \in S}\sum_{\forall k \in P^j}\sum_{\forall u \in U^{jk}}(\varepsilon_{ijku} \cdot \lambda_{jku}) \le 1, \forall i \in T \tag{10}$$

$$\sum_{t=1}^{k}\sum_{\forall u \in U^{jk}}(c_{jtu} \cdot \lambda_{jku}) - \sum_{t=1}^{k-1} d_{jt} \le q_j, \forall j \in S, \forall k \in P^j \tag{11}$$

$$-\sum_{t=1}^{k}\sum_{\forall u \in U^{jk}}(c_{jtu} \cdot \lambda_{jku}) + \sum_{t=1}^{k} d_{jt} \le 0, \forall j \in S, \forall k \in P^j \tag{12}$$

$$d_{jk} \le o_{jk}, \forall j \in S, \forall k \in P^j \tag{13}$$

$$\sum_{\forall u \in U^{jk}} \lambda_{jku} = 1, \forall j \in S, \forall k \in P^j \tag{14}$$

$$\lambda_{jku} = \{0, 1\}, d_{jk} \ge 0, \forall j \in S, \forall k \in P^j, \forall u \in U^{jk} \tag{15}$$

MP has fewer constraints but a large number of variables and columns. Therefore, CG can be employed to solve the linear relaxed MP. Unlike the classical set covering program, this MP has complicated constraints (11) and (12). Macroscopically, they have blocked diagonal structures and each block has a cumulative structure internallly. The complication of MP leads to a highly degenerate (i.e., extremely slow convergence) in the CG process.

We found that using an exact CG process (using Simplex to solve the restricted MP) would generate a large number of infeasible solutions (Desrosiers and Lübbecke, 2005; Lübbecke and Desrosiers, 2005) for MPs that would seldom enter the basis (or enter with extremely small coefficients). This would bring little change to the optimal dual solution (also known as a shadow price). The sub-problems would generate extremely similar columns in the next iteration of the CG process. As a result, this would be endlessly repeated with little improvement.

It can be concluded that MP in this paper is much different from the classical set covering program or set partition program. The generated infeasible solutions are difficult to handle. On this consideration, we employ the new inexact CG process, PDCGM which was proposed by Gondzio et al. (2013, 2016). According to our investigation, PDCGM could efficiently solve the linear relaxed MP.

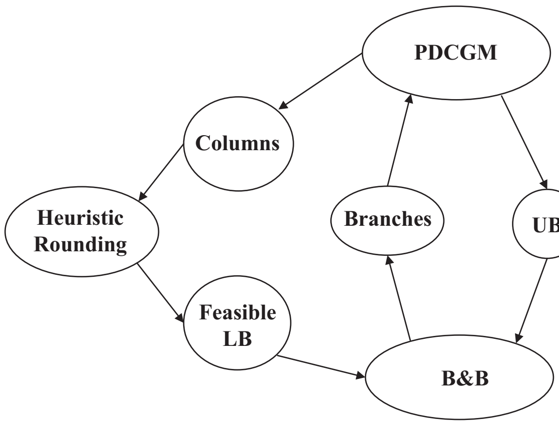

Fig. 5. The structure of the B&P algorithm.

5.1.2. Price problem

In CG process, we need to solve multiple pricing problems during each iteration. The solutions of these pricing problems correspond to the columns in MP. Let θ_i^τ, ζ_{jk}^τ, ξ_{jk}^τ, ϕ_{jk}^τ respectively represent the optimal dual solutions corresponding to constraints (10–12, 14) at iteration τ. The term ε_{ijku}, which is a parameter indicating whether strip i is selected in σ_{jku} in MP, now is a variable. We obtained the pricing problem for period k on EOS j at iteration τ based on optimal condition of Simplex, shown in (16–19).

$$z_{jk}^\tau = \max \sum_{\forall i \in T} (w_i \cdot \varepsilon_{ijku}) \tag{16}$$

$$w_i = v_i - \theta_i^\tau + \left[\sum_{t=k}^{|P^j|} (\xi_{jt}^\tau - \zeta_{jt}^\tau) \right] \tag{17}$$

$$\varepsilon_{ijku} + \varepsilon_{rjku} \le 1, \forall (i, r) \in O^{jk} \tag{18}$$

$$\varepsilon_{ijku} = \{0, 1\}, \tag{19}$$

Eq. (16)is the objective of the pricing problem and constraints (18) correspond to (5) which mean that each EOS can only image strip one by one. The structure of the pricing problem, which is an interval scheduling problem (weighted activity selection problem), is relative simple. A DP algorithm can be easily developed to solve it.

An MP and multiple pricing problems (the number of pricing problems equals the number of total periods) were obtained by DW decomposition. The pricing problems, which only have relationships with optimal dual solutions, were mutually independent.

A B&P algorithm was further developed to solve the problem to optimal. Fig. 5 illustrates the structure of our B&P method. The PDCGM is embedded into a B&B framework. In order to accelerate the convegence of the algorithm, we developed a heuristic to find good integer solutions (lower bounds, LBs) utilizing the generated historical columns, to prune B&P branches.

5.2. The algorithms

In this section, we provide a detailed introduction of the proposed B&P algorithm. A DP algorithm for solving pricing problems, a heuristic for obtaining good integer solutions utilizing historical columns, and branching-searching-pruning strategies in B&B, are respectively discussed.

5.2.1. DP algorithm for pricing problem

The pricing problem is actually an interval scheduling problem or called weighted activity selection problem. A DP algorithm is developed to solve it (Lew and Mauch, 2006). For given ITWs in a period k on EOS j, we sort them according to the end time in ascending order. We denote the sorted ITWs list as imaging actions list $\{1, 2, 3 \cdots N\}$. To ensure integrity, we add a virtual start and end imaging action 0 and $N+1$. Let $AL = \{0, 1, 2, 3 \cdots N, N+1\}$ be the actions list and $a \in AL$ be the action index. The terms s_a and e_a respectively represent the start and end time of action a, while $w(a)$ is the weight (computed by Eq. (17)) associated with action a. The pricing problem involves selecting conflict-free actions with the highest total weights, as shown in Fig. 6.

The conflict-free imaging action set for action a is: $E_a = \left\{a \neq a' | e_{a'} + \ell_{a'}^a \le s_a\right\}$, where $\ell_{a'}^a$ is the transition time from imaging action a' to a. In order to obtain optimal sub-structure, we define: $\pi(a) = \max\{E_a\}$. Each action can be viewed as a stage where we define $g = \{1, 2 \cdots N+1\}$ as stages, $d(g)$ as the decision at stage g, and $f(g)$ as the objective at stage g, respectively. If action $g-1$ is selected at stage g, then $f(g) = w_{g-1} + f(\pi(g-1))$. Otherwise $f(g) = f(g-1)$. Therefore, we obtain the optimal substructure and the dynamic decision equation, as

$$f(g) = \max_{d_g \in \{0,1\}} \Big(d_g \cdot \big(w_{g-1} + f(\pi(g-1)) \big) + (1 - d_g) \cdot f(g-1) \Big). \tag{20}$$

According to this decision equation, we can enumerate the decisions for all stages. Hence a DP algorithm is developed as shown in Algorithm 2.

Algorithm 2 DP algorithm for pricing problems.

```
Require: imaging actions list $AL$;
Ensure: selected imaging actions list $SL$;
1: calculate $\pi(a)$;
2: for $a = 1$; $a < N$; $a{+}{+}$ do
3:     for $a' = 1$; $a' < N$; $a{+}{+}$ do
4:         if $a' = a$ then
5:             Continue;
6:         end if
7:         if $e_{a'} + \ell_{a'}^a < s_a$ then
8:             $\pi(a){+}{+}$;
9:         end if
10:    end for
11: end for
12: calculate $f(g)$ and $d(g)$ according to formular (20);
13: for $g = 1$; $g < N+1$; $g{+}{+}$ do
14:     $f_a = w(g+1) + f[\pi(g-1)]$;
15:     $f_b = f(g-1)$;
16:     if $f_a > f_b$ then
17:         $f(g) = f_a$;
18:         $d(g) = 1$;
19:         put the action associated with stage $g$ into $SL$;
20:     else
21:         $f(g) = f_b$;
22:         $d(g) = 1$;
23:     end if
24: end for
```

Ensured by the optimal condition of DP, Algorithm 2, the time complexity of which is $O(n^2)$, where n is the length of action list, can provide at least one optimal solution for each pricing problem. The very efficient Algorithm 2 can return a column for each block of the restricted MP in each iteration of CG rapidly, which fairly speed the convergencing of CG process. The CG process starts from some initial columns that form the initial restricted MP. In this

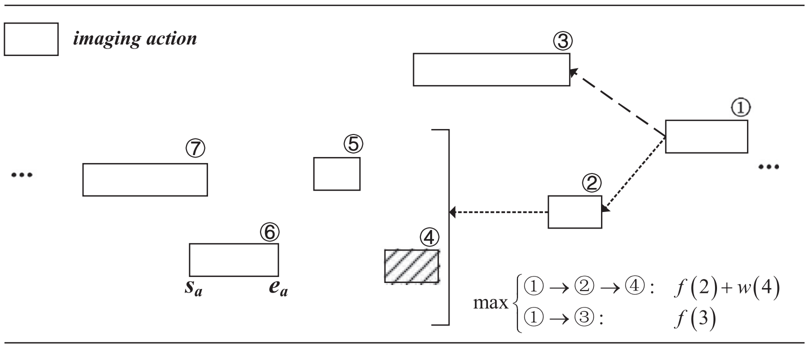
imaging action
①
②
③
④
⑤
⑥
⑦
s_a
e_a
$\max\begin{cases}①\to②\to④: & f(2)+w(4)\\ ①\to③: & f(3)\end{cases}$

Fig. 6. Selection of actions in the DP algorithm.

paper, we provide two types of inital columns. The first columns corrresponds to special solutions that no actions are selected for each pricing problem. The primal feasibility are guaranteed when these columns are added. The second columns are generated by Algorithm 2 where the weights of all actions are set to the profits of corresponding strips. We make such strategy that generating the second type of columns for the purpose of speeding the CG process.

5.2.2. A Heuristic

The generated historical columns, actually the sub-schedules, are helpful in constructing good feasible integer solutions. Combining these sub-schedules into a feasible complete schedule for the primal problem requires little computational costs. Therefore, it is beneficial to develop a rounding heuristic utilizing the generated historical columns by combining sub-schedules to find good feasible integer solutions. This heuristic can be embedded into the B&B framework to prune branches and nodes of B&P tree.

Our rounding heuristic is based on the greedy algorithm. The main idea of the algorithm is orderly selecting the most profitable columns (sub-schedules) to construct a complete schedule. The selected columns would be revised for feasibility (e.g., removing a strip from the sub-schedule). The heuristic is embedded into the B&B framework and would be called at the end of each CG process. Our computational experiences show that the heuristic could greatly reduce the optimality gap. The heuristic is introduced in details in Algorithm 3.

Algorithm 3 heuristic for good integer solutions.

```
Require: sub-schedules set (historical columns) for period $k$ on EOS $j$ - $SC^{jk}$;
Ensure: whole schedule $C$ and imaged strips list $IS$;
1:  for $j = 1$; $j < |S|$; $j++$ do
2:      for $k = 1$; $k < |P^j|$; $k++$ do
3:          remove strip $i$ from $SC^{jk}$, where strip $i$ in $IS$;
4:          update the profit and memory of $SC^{jk}$;
5:          select the “feasible“ sub-schedule $SC^{jk*} \in SC^{jk}$ with the maximal profit;
6:          //the “feasible“ means the storage capacity
7:          //constraints are satisfied if adding
8:          //the sub-schedule to schedule C
9:          add the sub-schedule $SC^{jk*}$ to schedule $C$;
10:         determine the downloading volume for period $k$ on EOS $j$ in schedule $C$;
11:         update the storage volume for period $k$ on EOS $j$ in schedule $C$;
12:         put strip $i$ into $IS$, where strip $i$ is in $SC^{jk*}$;
13:     end for
14: end for
```

5.2.3. B&b strategies

In this section, the branching, searching and pruning strategies are introduced, as shown in Fig. 7.

Here we employ the binary branching strategy. An imaging action a_{ijk}, associated with a weight (see Eq. (17)), denotes imaging strip i in period k on EOS j, and $x_{ijk} = \{0, 1\}$ denotes that whether a_{ijk} is adopted. A binary branching would create two branches, which are respectively denoted by $x_{ijk} = 0$ and $x_{ijk} = 1$.

If these branches were added into the MP, the structures of both MP and pricing problems would be destroyed. Therefore, in order to maintain the structures, we intentionally add the branches into pricing problems. A branch $x_{ijk} = 0$ means that the strip i would not be imaged in period k on EOS j. If the imaging action a_{ijk} is removed, the pricing problem corresponding to period k on EOS j would never return a column in which strip i was imaged. Furthermore, if the weight of action a_{ijk} is reset to $-M$ where M is a big-enough number then action a_{ijk} would never be selected because we use an optimal DP method to solve the pricing problem. That is, $x_{ijk} = 0$ would always hold in this case. On the contrary, for branch $x_{ijk} = 1$, if the weight of action a_{ijk} is reset to M, the action would always be selected provided no other incompatible actions exist with the same weight M. In other words, $x_{ijk} = 1$ would always hold in this case. Note that two issues should be considered for a $x_{ijk} = 1$ branch:

1) If there exists an action $a_{ij'k'}$, $j' \neq j$, $k' \neq k$, $\forall j' \in S$, $\forall k' \in P^{j'}$, then a branch $a_{ij'k'} = 0$ should be added at the same time;

2) if there exists an action $a_{i'jk}$, $i' \neq i$, $\forall i' \in T$, such that $(i, i') \in O^{jk}$, then a branch $a_{i'jk} = 0$ should be added at the same time.

After each branching, some variables should be removed from branch-able variables list accordingly. This would also lead to a reduction of B&P tree.

A Best First (BF) searching strategy was employed. At the end of each branching, the leaf node with the best feasible lower bound was selected as the branching node. Two new branches were added as the new leaf nodes for this node. This operation is iterated until no new branch is found.

We developed three pruning strategies. Rule (1) If the UB obtained by solving a branching problem is less than or equal to the feasible lower bound, this node should be pruned. Rule (2) When adding an $x_{ijk} = 1$ branch, if the required minimal storage resource given by $\sum_{\forall i \in T^1} c_i$, $T^1 = \{i | x_{ijk} = 1 \text{ by branching}\}$ for period k is greater than the storage capacity, this node should not be solved and should be pruned directly. Rule (3) When adding an $x_{ijk} = 0$ branch, if the total profit of removed strips by branching: $\sum_{\forall i \in T^0} p_i$, $T^0 = \{i | x_{ijk} = 0 \text{ by branching}\}$, is greater than or equal to UB − LB, then this node should not be solved and should be pruned directly.

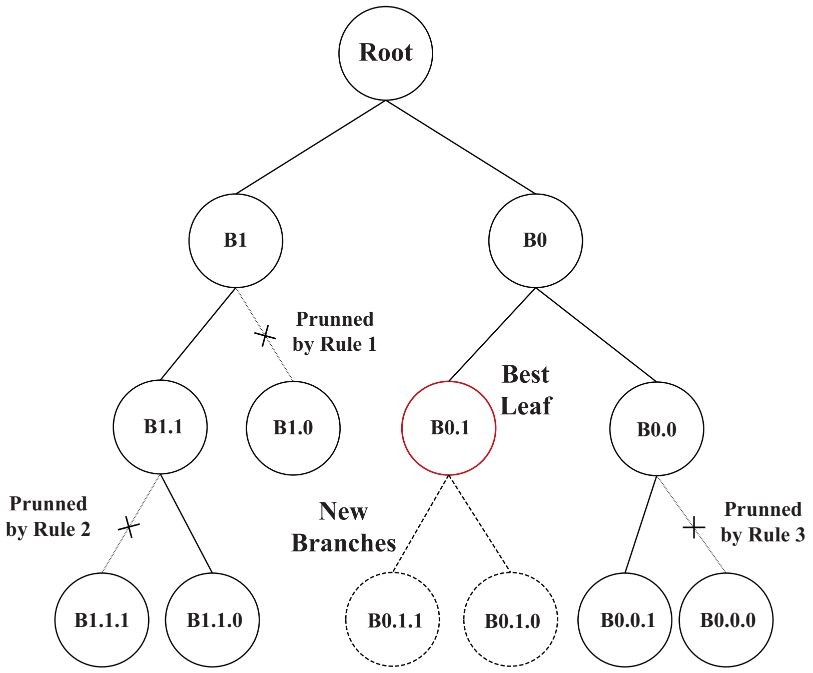

Fig. 7. Branching, searching and pruning of the B&P tree.

6. Computational experiments

Our validation consists of two parts, the first is a numerical test of simulated instances and the other is a numerical test of random instances. In the first test, we established a constellation consisting of three EOSs. The orbit of each EOS is similar to the orbit of the China Gaofen-1 satellite. We collected multiple imaging requests from users, with these data serving as imaging polygons. We used related software to simulate the environment and compute the ITWs and DTWs. In the second test, scheduling instances were produced randomly. We carried out a numerical test of both data types.

All tests were run on a Debian 7 64-bit Linux system with an i7 and 2G RAM. The PDCGM software developed and released by Gondzio et al. (2013, 2016) was used. The software, the code of which is available for research use, efficiently implements the PDCGM technique. Except for the CPU time, all data can be re-acquired by running the program. When employing the PDCGM software, the linear relaxation of MP should be solved to optimal conditions in each node of the BP tree. This requires each call of PDCGM return ret = 0, indicating that a solution was achieved. PDCGM system parameters included: $Delta = 1E-3$, $D = 10$, $Epsilon = 0.2$. We employed the warm start technique (Gondzio and González-Brevis, 2015) which was integrated into the PDCGM. The box-step method was also integrated into the PDCGM. We used a loose box because we found that a tight box led to solution failures.

6.1. Simulated instances

The constellation consisted of three EOSs in a sun synchronous orbit, the altitude of which was 645 km and the inclination was 98.0506°. The *local time of descending nodes* for the three EOS were 0:0:0, 8:0:0, and 16:0:0 respectively. Each EOS carries a homogenous visible light camera, with an imaging width of ~30 km, and a homogenous antenna which can communicate with ground stations.The storage capacity of each EOS in the constellation is 2TB. This parameter is set according to the Chinese Gaofen-1 satellite.

All imaging requests in our simulated environment formed large rectangular areas, the length and width of which varied from dozens to hundreds of kilometers. Most of these requests were located in Asia, especially in China. Each imaging request is associated with a profit, which is determined according to factors such as location, type, and degree of urgency, etc. The users provide these data. Each imaging request is split into several strips according to the imaging width and scanning direction. The length of each strip varies from several to dozens of kilometers. The profit of each strip is calculated using Eq. (1). The storage consumption for each strip is a linear function of its length. The storage consumption of all strips varies from dozens to hundreds of GB. On average, recording a strip may require ~200 GB of storage.

Three ground stations were selected to communicate with the constellation. They were *Sanya, Kashi*, and *Jiamusi*, which are located in the south, northwest, and northeast regions of China. We used appropriate software to simulate the environment. The access data included ITWs and DTWs are computed in advance. We constructed 10 instances in which the number of polygons were [10, 20,···, 100]. The Polygons in each of these instances were randomly selected from all imaging requests. We tested each instance on three scheduling horizons: 48, 72, and 96 h. Longer scheduling horizons resulted in more ITWs, DTWs, and periods. Accordingly, more variables and constraints exist.

In these tests, we did not carry out a full search of the B&P tree, but a rounding heuristic based on CG. In other words, the B&B framework was not used. We ran a CG process to find tight UBs and then used a rounding heuristic utilizing the generated columns to find suitable integer solutions. We carried out these tests because we found that for simulated instances, this method provided tighter feasible LBs and UBs. The computational results are shown in Table 1. The abbreviations used are as follows:

ITC: Instance name
NPL: Number of polygons
NST: Total number of strips
NTW: Total number of ITWs
NPR: Total number of periods
SCH: Scheduling horizon

Table 1
Statistical computational results for simulated instances D01–D10.

ITC	NPL	NST	TP	SCH/h	NTW	NPR	NCF	NSR	UB	LB	GAP	CPU Time (s)
D01	10	170	33,406	48	520	65	458	121	15,403	15,404	0.00%	1.37
				72	809	101	723	134	23,273	23,274	0.00%	2.89
				96	1091	137	948	151	29,495	29,209	0.97%	11.58
D02	20	239	51,976	48	668	67	853	154	28,288	28,157	0.46%	1.03
				72	1012	102	1386	173	37,262	36,145	3.00%	4.94
				96	1365	137	1904	198	45,468	44,694	1.70%	10.00
D03	30	307	83,159	48	786	68	1263	172	39,952	39,417	1.34%	1.18
				72	1211	103	2015	209	56,806	55,207	2.81%	4.51
				96	1656	140	2877	241	71,726	68,526	4.46%	21.56
D04	40	378	120,476	48	928	72	1862	191	55,509	55,289	0.40%	1.57
				72	1430	111	2930	237	83,393	82,321	1.29%	14.48
				96	1945	150	4048	282	101,778	99,404	2.33%	21.59
D05	50	449	163,480	48	1099	74	2518	224	78,100	77,123	1.25%	1.91
				72	1678	114	3888	284	113,754	111,678	1.82%	16.05
				96	2254	153	5270	334	139,373	134,335	3.61%	20.78
D06	60	516	192,802	48	1206	71	2697	240	87,385	86,212	1.34%	2.86
				72	1849	110	4392	311	130,555	124,665	4.51%	16.71
				96	2493	149	6095	382	163,286	158,489	2.94%	36.05
D07	70	587	198,657	48	1377	74	3813	267	90,142	89,175	1.07%	2.85
				72	2089	114	5873	348	132,915	128,346	3.44%	10.04
				96	2803	154	7957	426	166,845	160,734	3.66%	27.36
D08	80	657	248,639	48	1531	75	4756	277	110,065	106,103	3.60%	3.60
				72	2321	115	7369	370	162,341	153,696	5.33%	13.98
				96	3102	153	9940	451	203,142	191,891	5.54%	47.29
D09	90	726	266,240	48	1639	75	5307	282	112,974	108,405	4.04%	3.43
				72	2500	115	8269	379	168,761	159,538	5.47%	28.13
				96	3371	155	11,410	469	212,431	195,512	7.96%	72.99
D10	100	796	296,950	48	1795	75	6,250	288	124,910	114,575	8.27%	4.86
				72	2730	115	9,726	375	184,414	152,545	17.28%	22.24
				96	3670	156	13,235	488	234,225	208,139	11.14%	74.61

NCF: Number of incompatible ITW pairs
NSR: Number of imaged strips in schedule
TP: Total profit of all strips
OBJ: Objective value (LB)
UB: Upper bound
GAP: Optimality gap, computed by (UB – LB) / UB
NNS: The number of searched tree nodes.

The number of strips varied from 170 to 796 and the number of polygons varied from 10 to 100. Accordingly, the number of ITWs varied from 520 to 3670 with the number of incompatible ITWs varying from 458 to 13,235. In total, on the same scheduling horizon, the optimality gaps showed an increasing trend with an increasing number of polygons. For the same instance, the optimality gaps showed an increasing trend with an increasing scheduling horizon. These results indicate the optimality gap would grow with increasing problem magnitude, which is mostly determined by the number of ITWs or number of periods. Similarly, the computational costs showed the same trend as the increasing problem magnitude. This can be concluded from the increasing CPU times shown in Table 1. Please note that the LBs of D01(48 h) and D01(72 h) are a little higher than the UBs in Table 1. The probable causes are computing deviations which appeared when using the PDCGM software to solve the master problems.

In most instances, the optimality gaps were less than or equal to 10%. None of the CPU times exceeded 100 s, which is acceptable for users. In practical settings, users are typically more concerned with computational cost performance (i.e., the ratio between the optimality and computational costs). The results shown in Table 1 are satisfactory because the rounding heuristic based on CG could provide high-quality solutions in fairly short CPU times (100 s). This finding highlights the method's merit for actual applications. We can conclude from Table 1 that the problem is an oversubscribed problem in which the resource is too scarce to satisfy all imaging requests. This is because the UBs obtained by solving the linear relaxed MP with CG were less than the total profits of all strips. That is to say, not all strips could be imaged in a given time horizon. In our tests, the three EOSs could image at most 288 strips in 48 h or 488 in 96 h. These numbers provide a reference for the actual management of EOS resources.

We were also curious about why the heuristic based on CG could provide such good solutions. We propose the following: 1) The DP algorithm for each SP always provides an optimal integer column, which is beneficial for the entire schedule. As a result, this procedure could be very robust and feasible solutions could always be found 2) For simulated instances, the data may be regular or easily solvable. For example, the strips that split from the same polygon are mostly incompatible. This regulation may lead to a reduction in computational costs .

In order to check the stability and flexibility of the CG-based heuristic, we also conducted a parameter-sensitive (storage capacity) analysis. We selected instance D02–D10 under a 72 h scheduling horizon and ran the program with storage capacities of 500, 1000, 1500, 2000, 2500, and 3000 GB. The UBs and LBs are shown in Fig. 8.

As can be observed from Fig. 8, overall the UBs and LBs show a non-descending trend with increasing storage capacity. This finding indicates that more strips could be imaged with more resources. However, unlike the UBs, the LBs do not strictly follow this trend. This means that the B&P algorithm provides different solutions for the same instance with different storage capacities. Such a finding is not unreasonable because our heuristic may construct different feasible integer solutions, which may be selected as LBs, in accordance with the different columns generated during the CG process because of different storage capacities. The GAPs of instances D02–D09 were less than 10% for all storage capacities. This demonstrates the method achieved good flexibility for solving actual instances. The GAPs show a descending trend with increasing storage capacity in most instances. This can be explained as that the problem gets easier with more sufficient resources and hence a better LB can be found.

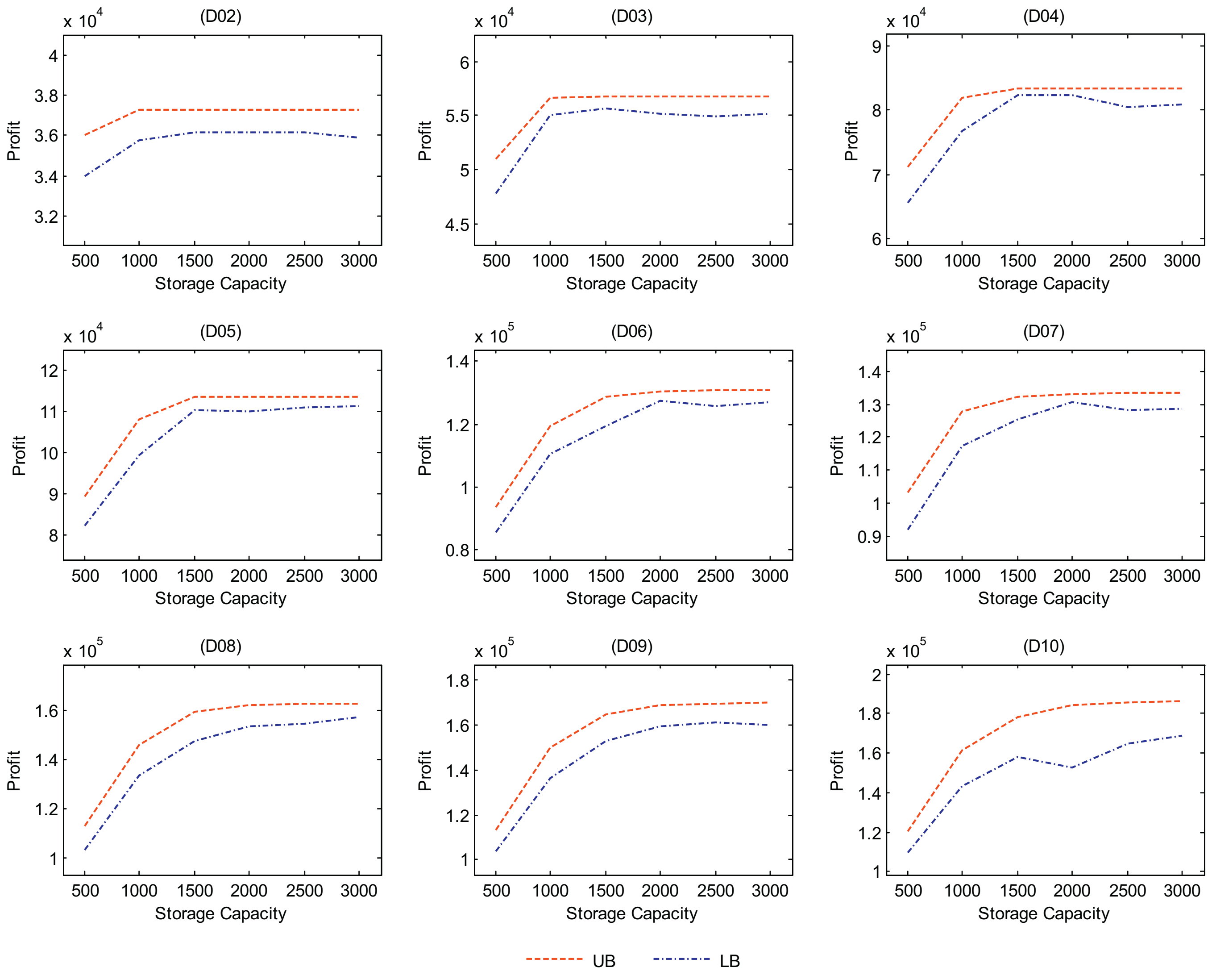

Fig. 8. The UBs and LBs for instances D02–D10 under 72 h with different storage capacitie.

6.2. Random instances

To further validate performance, we tested the B&P method on random instances. Although the CG-based heuristic provides a tight feasible LB, which could significantly reduce the optimality gap and multiple pruning strategies can be employed, carrying out a full search of B&P tree still consumes large computational resources. Therefore, we set the termination condition at “gap $\leq$ 5%” or “number of branching $=$ 50” instead of carrying out a full search. We produced 3 groups of random instances: D11–D40. In the first group (D11–D20), strips had fewer ITWs in each instance. On average, each strip had $\sim$2 ITWs. In the second group (D21–D30), strips had moderately-scaled numbers of ITWs in each instance. On average, each strip had $\sim$4 ITWs. In the third group (D31–D40), strips had sufficient ITWs in each instance. On average, each strip had $\sim$20 ITWs. The 3 groups of random instances corresponded to short-term, medium-term, and long-term scheduling problems, respectively.

The number of periods for all instances in the 3 groups were randomly produced. The first group included $\sim$50 periods and the other two groups included $\sim$100 periods. The profit and storage consumption for each strip were selected from 1–100. Hence, each strip consumes an average of $\sim$50 units of storage resources. The storage capacity was set to 500. Nearly 10 strips can be recorded for strips which consume 50 units of storage resources. The running results are shown in Table 2.

In instances D11-D20, the maximum number of incompatible ITWs reached 101,852. All optimality gaps were less than 5%. The numbers of nodes in BP trees were very small, such that the maximum NNS was 5. The included CPU times were also small, with a maximum time of 1.04 s. All UBs were less than the total profits (TPs). This indicates the method can find a tighter UB for the problem in these instances.

In instances D21–D30, the maximum number of incompatible ITWs reached 209,890, which is fairly high. More nodes need to be searched in B&P trees to reduce these gaps. The maximum NNS reached 103. However, there were 6 instances which were efficiently solved (D21–D24, D27, D28). The most time-consuming instance (D30) required 179.44 s to find a solution with an optimality gap of 5.25%. This is still computationally cost-effective. Except for instance D21, all UBs for the other instances were less than the total profits (TPs). Most gaps were less than 5% and all gaps were less than 10%. This indicates the B&P method can solve these instances to near optimal.

In instances D31–D40 (large-scale numbers in the ITWs), the maximum number of incompatible ITWs reached 926,888, which

Table 2
Statistical computational results for random instances D11–D40.

ITC	NST	NTW	NPR	NCF	TP	NSR	UB	LB	GAP	NNS	CPU Time (s)
D11	58	105	57	1973	2501	39	1858	1859	0.00%	1	0.12
D12	90	179	59	5687	4824	56	3826	3642	4.81%	3	0.57
D13	130	260	48	10,791	6617	49	3451	3303	4.29%	1	0.15
D14	160	299	29	14,364	8561	38	2847	2776	2.49%	3	0.32
D15	181	354	30	20,083	9374	43	3167	3045	3.85%	1	0.12
D16	247	439	39	29,917	12,564	53	4258	4128	3.05%	1	0.14
D17	285	520	38	40,138	13,853	52	4052	3985	1.65%	1	0.21
D18	328	590	40	48,404	16,299	58	4396	4287	2.48%	1	0.14
D19	368	696	38	65,067	19,050	53	4278	4078	4.68%	5	1.04
D20	409	902	62	101,852	20,616	74	6130	5962	2.74%	1	0.3
D21	45	166	71	2089	2334	45	2333	2334	0.00%	1	0.22
D22	104	413	107	14,022	5196	89	5032	4804	4.53%	1	0.83
D23	167	682	109	34,647	8398	111	7500	7130	4.93%	29	38.89
D24	249	1008	90	64,648	12,071	110	8272	7882	4.71%	17	13.75
D25	254	999	80	72,807	12,841	106	8181	7630	6.74%	73	60.67
D26	286	1132	98	81,108	13,392	124	8482	8029	5.34%	103	153.78
D27	349	1456	120	133,682	17,760	145	10,965	10,419	4.98%	9	34.35
D28	381	1527	80	145,445	18,857	108	8588	8161	4.97%	3	3.22
D29	402	1596	79	156,035	20,591	108	8910	8408	5.63%	73	54.02
D30	413	1716	111	209,890	20,323	140	10,820	10,252	5.25%	78	179.44
D31	48	933	71	10,857	2239	48	2239	2239	0.00%	1	1.6
D32	113	2198	86	62,223	5657	109	5657	5600	1.01%	1	6.68
D33	164	3182	77	119,040	8346	145	8346	8015	3.97%	1	8.64
D34	202	3858	92	174,420	10,602	173	10,602	10,143	4.33%	1	4.93
D35	234	4431	94	221,239	11,599	202	11,599	11,176	3.65%	11	96.49
D36	287	5558	120	358,139	14,686	244	14,686	13,957	4.96%	43	610.55
D37	302	5839	105	423,068	15,275	246	15,275	14,453	5.38%	103	1,313.73
D38	329	6326	109	492,987	16,963	268	16,963	15,851	6.56%	103	2,002.02
D39	342	6675	104	556,171	16,981	270	16,955	15,706	7.37%	103	1,988.27
D40	447	8662	95	926,888	22,323	303	20,990	19,006	9.45%	103	3,658.63

is extremely large. More nodes need to be searched in these B&P trees in order to reduce these gaps. The maximum NNS reached 103. However, there were still 6 instances (D31-D36) being solved efficiently. The most time consuming instance (D40) required 3,658.63 s to find a solution with an optimality gap of 9.45%.

Excluding instances D39 and D40, all UBs were equal to the total profits (TPs). As these instances correspond to long-term scheduling problems, which have large numbers of ITWs, the corresponding imaging opportunities are typically sufficient. The gaps for instances D31-D36 were less than 5% and all gaps were less than 10%. Therefore, one could conclude that for all random instances introduced in this paper, the B&P could always find high-quality solutions with a gap of less than 10%.The runtime required to solve a linear MP relaxation problem increases with increasing problem magnitude. This was also observed in the tests on simulated instances. Solving each node (branching) problem is similar with the fully linear relaxation problem. The runtime required by each node is close to that root on the B&P tree. For the most time-consuming instance (D40), which required 3,658.63 s to find a feasible LB with an optimality of 9.45%, only 103 nodes were searched. It can be concluded that carrying out a full search of the B&P tree is time-consuming in such long-term scheduling problems.

To verify the strength of the proposed B&P algorithm, we conducted comparison tests with two other commonly employed algorithms; namely, the greedy algorithm (GA), which is a heuristic, and the SA algorithm, which is a meta-heuristic based on neighbour searching. The GA's main idea was that at each step, the algorithm tried to make an observation plan for the strip with the highest indicator. This process was repeated until all strips were considered. The indicator for each strip was computed following Wang et al. (2011)'s method. With regard to the SA algorithm, a solution was defined as a strips' list with a specified order. The algorithm tried to make observation plans for the strips one by one, following the specified order. A neighbour solution was obtained by slightly modifying the solution; namely, by exchanging the sequences of two randomly selected strips. The SA algorithm conducted a continuous exploration of the neighbour solutions from a given initial solution. In order to obtain the best solution experience, the solution obtained by the GA was used as this initial solution. The SA algorithm explored 20 neighbour solutions for each iteration. The best solution (with maximal objective value) was selected to be the candidate solution. An acceptance function was used to determine whether or not to accept the candidate solution. The SA algorithm was run until it could provide no further improvements for objective values during 50 exploratory iterations. Table 3 provides an overview of the objective values computed by the three algorithms. With regard to the SA algorithm, the average, maximum, and minimum values were determined from a series of 20 trials.

As can be observed from Table 3, the minimum objective values computed by the SA algorithm are larger than the objective values computed by the GA because the solutions obtained by the latter were used as initial solutions for the former. Moreover, the objective values computed by the B&P algorithm are larger than the maximum objective values computed by the SA algorithm in nearly all instances. Please also note that even when more CPU time was available, the GA and SA algorithm could not provide solutions as good as those of the B&P algorithm for most tested instances. The findings demonstrate the proposed B&P algorithm's strength in terms of its ability to find better solutions.

We also conducted a parameter (storage capacity) sensitive analysis for instances D22–D30. These storage capacities were 100, 200,···, 1000 (units). The corresponding UBs/LBs are shown in Fig. 9.

The UBs increased with increasing storage capacity in all instances. The LBs also increased with increasing storage capacity in most instances. However, the LBs occasionally decreased with increasing storage capacity, which is significantly different from the

Table 3
A comparison of computational results among the B&P algorithm, the GA, and the SA algorithm for instances D11-D40.

ITC	B&P		GA		SA			
	OBJ	CPU Time(s)	OBJ	CPU Time(s)	MEAN	MAX	MIN	CPU Time(s)
D11	**1859**	0.12	1675	0.00	1812	1812	1812	0.06
D12	3642	0.57	3112	0.00	3766	**3807**	3727	0.16
D13	**3303**	0.15	2483	0.00	3022	3046	2978	0.28
D14	**2776**	0.32	1784	0.00	2459	2569	2404	0.35
D15	**3045**	0.12	2039	0.00	2781	2861	2689	0.49
D16	**4128**	0.14	2766	0.00	3794	3973	3634	0.83
D17	**3985**	0.21	2290	0.01	3549	3760	3264	1.10
D18	**4287**	0.14	2631	0.01	4007	4080	3942	1.30
D19	**4078**	1.04	2374	0.01	3755	3843	3601	1.60
D20	**5962**	0.3	3760	0.01	5354	5461	5265	2.67
D21	**2334**	0.22	**2334**	0.00	**2334**	**2334**	**2334**	0.09
D22	4804	0.83	4219	0.01	4911	**4975**	4822	0.55
D23	**7130**	38.89	5433	0.01	6801	6895	6718	1.52
D24	**7882**	13.75	5070	0.02	7150	7300	7067	3.33
D25	**7630**	60.67	5323	0.01	7134	7272	7044	3.15
D26	**8029**	153.78	5696	0.01	7322	7414	7142	4.03
D27	**10419**	34.35	7507	0.02	9407	9575	9287	6.89
D28	**8161**	3.22	5676	0.03	7349	7502	7237	7.19
D29	**8408**	54.02	5407	0.03	7726	7847	7611	7.95
D30	**10252**	179.44	6652	0.03	9256	9381	9070	8.99
D31	**2239**	1.6	**2239**	0.00	**2239**	**2239**	**2239**	0.68
D32	5600	6.68	**5657**	0.03	**5657**	**5657**	**5657**	6.24
D33	**8015**	8.64	6950	0.07	7338	7346	7320	16.78
D34	**10143**	4.93	8393	0.10	9404	9539	9312	27.45
D35	**11176**	96.49	10406	0.15	11035	11103	10987	42.14
D36	**13957**	610.55	11775	0.22	13477	13705	13143	61.12
D37	**14453**	1313.73	11512	0.27	13453	13541	13337	70.57
D38	**15851**	2002.02	12968	0.30	14735	14806	14641	84.29
D39	**15706**	1988.27	12637	0.35	14673	14869	14434	98.64
D40	**19006**	3658.63	13789	0.69	17176	17301	17092	201.68

UBs. The magnitude of the problem increased from instance D22–D30. Accordingly, the GAPs show an increasing trend. This also indicates that larger problems are more difficult to optimally solve. All GAPs are less than 10% and most of them fluctuate by up to 5%. This indicates that the method could provide high-quality solutions. In each instance, the GAP variation trend was irregular. This is likely because the GAPs are strongly influenced by LBs, which do not always grow with increasing storage capacity. Therefore, the optimality of the B&P method is parameter (storage capacity) dependent.

6.2.1. *Computational costs*

We also conducted an analysis of computational costs for instances D22–D30 with different storage capacities. The storage capacities were 100, 200,$\cdots$, 1000. The CPU time and number of searched nodes are shown in Fig. 10.

With all gaps less than 10% (Table 6), the runtimes did not exceed 300 s in instances D22–D30 (Fig. 9). A larger NNS requires more runtime. Computational cost has a weak relationship with problem magnitude and storage capacity. The problem magnitude grows from instance D22-D30. However, the CPU time and NNS did not show a clearly increasing trend. In each instance, the computational cost shows an irregular variation trend with increasing storage capacity. Therefore, the computational costs of the B&P method are also parameter (storage capacity) dependent.

7. Conclusion and future work

In this paper, we conducted a dedicated study of the EOSCIDIS problem. First, the scheduling problem was described and the structure of the problem was thoroughly analyzed based on period splitting. We proposed a primal MILP formulations for the problem accordingly. Second, we decomposed the primal program into a master problem and several pricing problems employing DW decomposition method. The MP has an extremely complex constraint structure while the pricing problems can easily be solved to optimal. An efficient DP algorithm was developed for the pricing problems. The complicated MP constraint structure resulted in an extremely slow convergence when using the CG method. As such, we employed PDCGM to solve the MP and embedded the CG process into a B&B framework to develop a B&P algorithm for the problem. In addition, the generated columns are sub-schedules, which are helpful for constructing feasible integer solutions. Hence, we developed a greedy-based heuristic embedded in the B&B framework to construct feasible solutions which could help prune B&P tree branches. Finally, we conducted numerical tests to validate the proposed algorithm. Our numerical tests included both simulated and random instances. The simulated instances were constructed according to the Chinese Gaofen-1 satellite. Tests showed that the use of PDCGM plus a heuristic could provide high-quality solutions for most simulated instances. We developed 3 types of random instances: short, medium, and long-term scheduling instances. The tests on random instances show that the proposed B&P method could compute near-optimal solutions of the 3 types of instances. Overall, the B&P achieved very good computational performance.

The EOS constellation imaging and downloading integrated scheduling problem is an interesting optimization problem in engineering. This paper provided a fundamental formulation for solving the problem. Several variations of the problem could be solved based on extensions of the method introduced in this paper. Various other realistic limitations could be added, such as energy consumption. Most EOSs can pitch cameras along the scanning direction. As such, they are becoming more flexible for performing imaging operations. In this variation, each strip could be imaged in longer ITWs. Therefore, the beginning imaging time is not fixed, which could significantly increase the solution space. The decomposition method used in this paper may be viable for such variations because the sub-problems can still be viewed as a DP problem. However, this needs further work because Algorithm 2 intro-

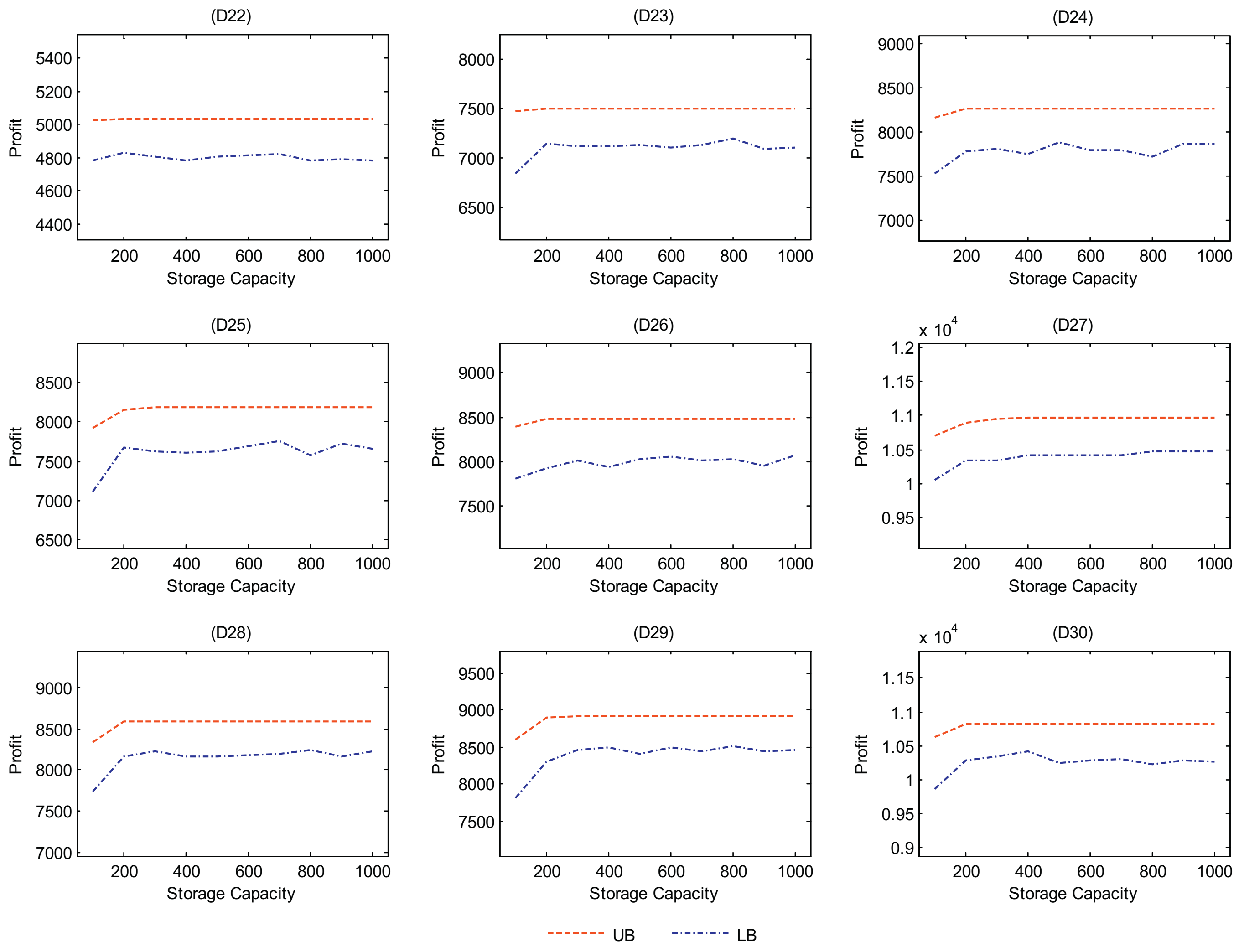

Fig. 9. The UBs and LBs for instances D22-D30 with different storage capacities.

duced in this paper should be adjusted for solving the new pricing problems in this variant.

Acknowledgement

The authors thank the editors and the anonymous referees for their constructive suggestions which have enabled the authors to significantly improve the paper. The authors also thank Elsevier Web Shop for English language editing. This study was funded by the National Natural Science Foundation of China under Grant 71671059, 71521001 and 71401048. The authors declare that there is no conflict of interests regarding the publication of this paper. This article does not contain any studies with human participants performed by any of the authors. Informed consent was obtained from all individual participants included in the study.

References

Álvarez, A.J.V., Erwin, R.S., 2015. An introduction to optimal satellite range scheduling (vol. 106). Springer.
Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H., 1998. Branch-and-price: column generation for solving huge integer programs. Oper. Res. 46 (3), 316–329.
Bensana, E., Lemaitre, M., Verfaillie, G., 1999. Earth observation satellite management.. Constraints 4 (3), 293–299.
Bianchessi, N., Righini, G., 2008. Planning and scheduling algorithms for the COSMO-skymed constellation.. Aerosp. Sci. Technol. 12 (7), 535–544.
Cordeau, J.F., Laporte, G., 2005. Maximizing the value of an earth observation satellite orbit.. J. Oper. Res. Soc. 56 (8), 962–968.
Dantzig, G.B., Wolfe, P., 1960. Decomposition principle for linear programs.. Oper. Res. 8 (1), 101–111.
Desrosiers, J., Lübbecke, M.E., 2005. A primer in column generation.. In Column Generation (pp. 1–32). Springer US.
Furini, F., Iori, M., Martello, S., Yagiura, M., 2015. Heuristic and exact algorithms for the interval min–max regret knapsack problem.. INFORMS J. Comput. 27 (2), 392–405.
Gabrel, V., 2006. Strengthened 0–1 linear formulation for the daily satellite mission planning.. J. Comb. Optim. 11 (3), 341–346.
Gabrel, V., Moulet, A., Murat, C., Paschos, V.T., 1997. A new single model and derived algorithms for the satellite shot planning problem using graph theory concepts.. Ann. Oper. Res. 69, 115–134.
Gabrel, V., Murat, C., 2003. Mathematical programming for earth observation satellite mission planning.. In Operations Research in Space and Air (pp. 103–122). Springer US..
Gabrel, V., Vanderpooten, D., 2002. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite.. Eur. J. Oper. Res. 139 (3), 533–542.
Globus, A., Crawford, J., Lohn, J., Pryor, A., 2004. A comparison of techniques for scheduling earth observing satellites.. In AAAI (pp. 836–843). July
Gondzio, J., González-Brevis, P., 2015. A new warmstarting strategy for the primal–dual column generation method.. Math. Program. 152 (1–2), 113–146.
Gondzio, J., González-Brevis, P., Munari, P., 2013. New developments in the primal–dual column generation technique.. Eur. J. Oper. Res. 224 (1), 41–51.
Gondzio, J., González-Brevis, P., Munari, P., 2016. Large-scale optimization with the primal-dual column generation method.. Math. Program. Comput. 8 (1), 47–82.
Habet, D., Vasquez, M., Vimont, Y., 2010. Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite.. Comput. Optim. Appl. 47 (2), 307–333.
Janiak, A., Kovalyov, M.Y., Lichtenstein, M., 2015. On a single machine-scheduling

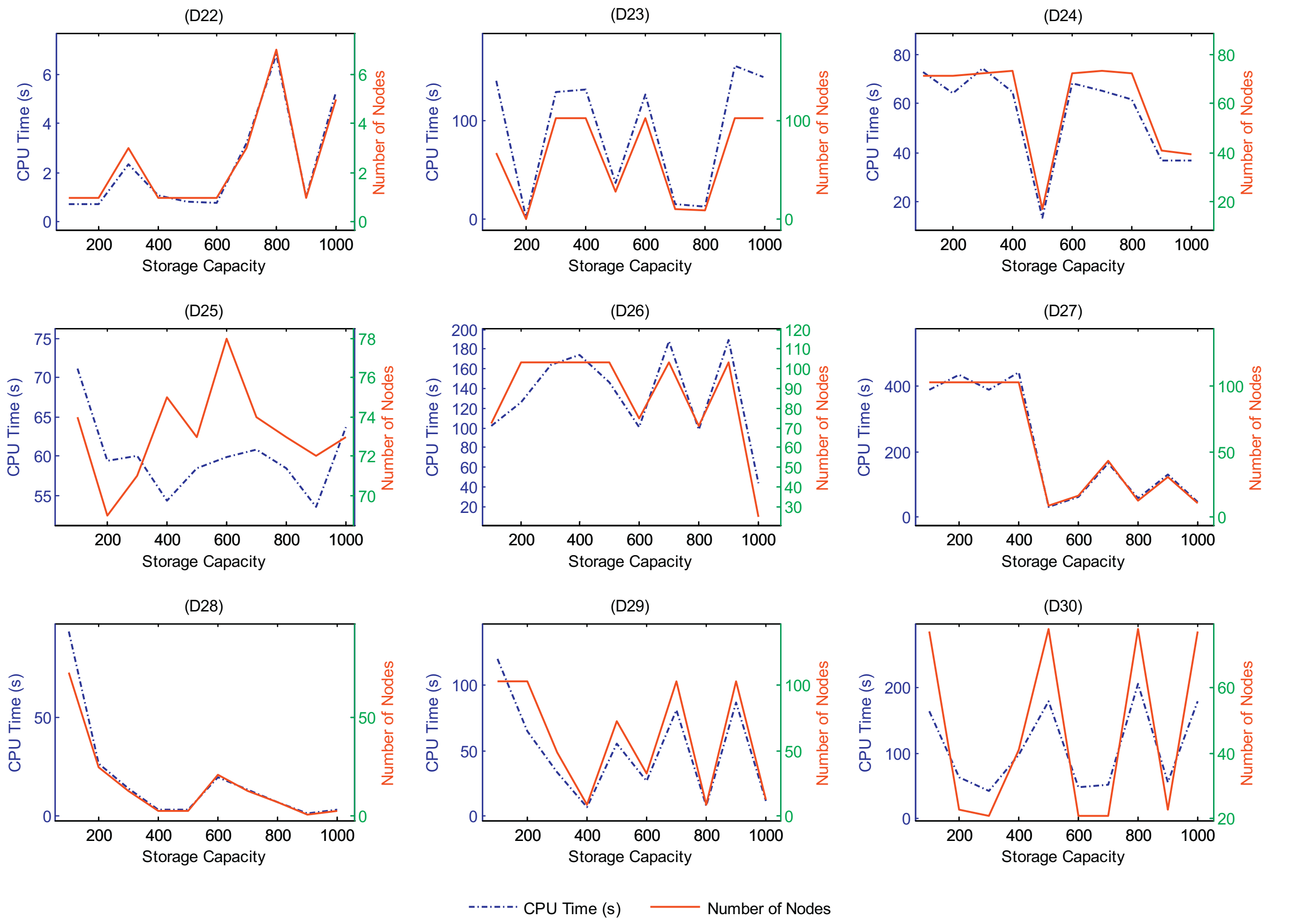

Fig. 10. CPU time and number of nodes for instances D22-D30 with different storage capacities.

problem with separated position and resource effects.. Optimization 64 (4), 909–911.
Jarrah, A.I., Qi, X., Bard, J.F., 2014. The destination-loader-door assignment problem for automated package sorting centers.. Transp. Sci. 50 (4), 1314–1336.
Karapetyan, D., Minic, S.M., Malladi, K.T., Punnen, A.P., 2015. Satellite downlink scheduling problem: a case study.. Omega 53, 115–123.
Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.M., Bataille, N., 2002. Selecting and scheduling observations of agile satellites.. Aerosp. Sci. Technol. 6 (5), 367–381.
Lew, A., Mauch, H., 2006. Dynamic programming: a computational tool (vol. 38). P 64 - 66. Springer.
Liu, X., Bai, B., Yingwu, C., Feng, Y., 2014. Multi satellites scheduling algorithm based on task merging mechanism.. Appl. Math. Comput. 230, 687–700.
Liu, X., Laporte, G., Chen, Y., He, R., 2017. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time.. Comput. Oper. Res. 86, 41–53.
Lübbecke, M.E., Desrosiers, J., 2005. Selected topics in column generation.. Oper. Res. 53 (6), 1007–1023.
Madry, S., 2013. Introduction and history of space remote sensing. In Handbook of Satellite Applications [pp. 657–666). Springer New York..
Malladi, K.T., Minic, S.M., Karapetyan, D., Punnen, A.P., 2016. Satellite constellation image acquisition problem: a case study.. In Space Engineering (pp. 177–197). Springer, Cham.
Malladi, K.T., Minic, S.M., Punnen, A.P., 2017. Clustered maximum weight clique problem: algorithms and empirical analysis.. Comput. Oper. Res. 85, 113–128.
Mansour, M.A., Dessouky, M.M., 2010. A genetic algorithm approach for solving the daily photograph selection problem of the SPOT5 satellite.. Comput. Ind. Eng. 58 (3), 509–520.
Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C., 2016. Branch-and-bound algorithms: a survey of recent advances in searching, branching, and pruning.. Discrete Optim. 19, 79–102.
Pelton, J.N., Madry, S., Camacho-Lara, S., 2012. Handbook of satellite applications. Springer Publishing Company, Incorporated.
Peng, G., Wen, L., Feng, Y., Baocun, B., Jing, Y., 2011. Simulated annealing algorithm for EOS scheduling problem with task merging. Inmodelling, identification and control (ICMIC). In: Proceedings of 2011 International Conference on (pp. 547–552). IEEE. June
Sarkheyli, A., Bagheri, A., Ghorbani-Vaghei, B., Askari-Moghadam, R., 2013. Using an effective tabu search in interactive resources scheduling problem for LEO satellites missions.. Aerosp. Sci. Technol. 29 (1), 287–295.
Song, J.S., Xiao, L., Zhang, H., Zipkin, P., 2016. Optimal policies for a dual-sourcing inventory problem with endogenous stochastic lead times. Operations Research.
Tangpattanakul, P., Jozefowiez, N., Lopez, P., 2015. Biased random key genetic algorithm for multi-user earth observation scheduling. In recent advances in computational optimization (pp. 143-160). Springer International Publishing.
Tangpattanakul, P., Jozefowiez, N., Lopez, P., 2015. A multi-objective local search heuristic for scheduling earth observations taken by an agile satellite.. Eur. J. Oper. Res. 245 (2), 542–554.
Vasquez, M., Hao, J.K., 2003. Upper bounds for the SPOT 5 daily photograph scheduling problem.. J. Comb. Optim. 7 (1), 87–103.
Verfaillie, G., Lemaître, M., Schiex, T., 1996. Russian doll search for solving constraint optimization problems.. In AAAI/IAAI 1, 181–187. August
Waiming, Z., Xiaoxuan, H., Wei, X., Peng, J., 2017. A two-phase genetic annealing method for integrated earth observation satellite scheduling problems.Soft Comput, https://doi.org/10.1007/s00500-017-2889-8.
Wang, J., Demeulemeester, E., Qiu, D., 2016. A pure proactive scheduling algorithm for multiple earth observation satellites under uncertainties of clouds.. Comput. Oper. Res. 74, 1–13.
Wang, J., Zhu, X., Yang, L.T., Zhu, J., Ma, M., 2015. Towards dynamic real-time scheduling for multiple earth observation satellites.. J. Comput. Syst. Sci. 81 (1), 110–124.
Wang, L., Zheng, X.L., Wang, S.Y., 2013. A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem.. Knowl. Based Syst. 48, 17–23.
Wang, P., Reinelt, G., Gao, P., Tan, Y., 2011. A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation.. Comput. Ind. Eng. 61 (2), 322–335.
Wolfe, W.J., Sorensen, S.E., 2000. Three scheduling algorithms applied to the earth observing systems domain.. Manage. Sci. 46 (1), 148–166.
Wu, G., Liu, J., Ma, M., Qiu, D., 2013. A two-phase scheduling method with the con-

sideration of task clustering for earth observing satellites.. Comput. Oper. Res. 40 (7), 1884–1894.

Xu, R., Chen, H., Liang, X., Wang, H., 2016. Priority-based constructive algorithms for scheduling agile earth observation satellites with total priority maximization.. Expert Syst. Appl. 51, 195–206.

Zhang, D., Guo, L., Cai, B., Sun, N., Wang, Q., 2013. A hybrid discrete particle swarm optimization for satellite scheduling problem.. In: In Conference Anthology, IEEE (pp. 1–5). IEEE. January

Zhang, Z., Zhang, N., Feng, Z., 2014. Multi-satellite control resource scheduling based on ant colony optimization.. Expert Syst. Appl. 41 (6), 2816–2823.