Optimizing ground station networks for free space optical communications: maximizing the data transfer

Mikaël Capelle, Marie-José Huguet, Nicolas Jozefowiez, Xavier Olive

ORIGINAL ARTICLE

Optimizing ground station networks for free space optical communications: maximizing the data transfer

Mikaël Capelle[1,2] | Marie-José Huguet[2] | Nicolas Jozefowiez[3] | Xavier Olive[1]

[1]Thales Alenia Space France, Toulouse, France

[2]LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France

[3]Université de Lorraine, LCOMS, F-57000 Metz, France

Correspondence
Marie-José Huguet, LAAS-CNRS, 7 av. du Colonel Roche, 31400 Toulouse, France
Email: marie-jose.huguet@laas.fr

Funding information

Free space optical communications are becoming a mature technology to cope with the needs of high data rate payloads for future low-earth orbiting observation satellites. However, they are strongly impacted by clouds. In this paper, we aim to find a network of optical ground stations maximizing the percentage of data acquired by a low-earth orbiting satellite that can be transferred to the Earth, taking into consideration cloud information. This problem can be separated in two parts and solved hierarchically: the selection of a network of optical ground stations and the assignment of downloads to visibility windows of the stations. We present theoretical and practical results regarding the complexity of the latter subproblem and propose a dynamic programming algorithm to solve it. We combine this algorithm with two methods for the enumeration of the stations, and compare them with a Mixed Integer Linear Program (MILP). Results show that even if the MILP can solve scenarios over small horizons, the hierarchical approaches outperform it in term of computation time while still achieving optimality for larger instances.

KEYWORDS
Combinatorial Optimization, Mixed-Integer Linear Programming, Dynamic Programming, Network Design,

1 | INTRODUCTION AND STATE OF THE ART

Free space optical communications are seen as a key technology [11, 13, 16] to cope with the needs of high data rate payloads for future low-earth orbiting observation satellites in replacement or in addition to current radio-frequency technologies. While the latter are very mature and well proven technologies which have been used for decades, the former may be able to offer data rates beyond the reach of radio-frequency technologies.

Current radio-frequency technologies mainly use X-Band for download and thus can currently provide up to a few gigabits per second (Gbps) [3]. Their main advantage is that X-Bands are not impacted by weather or atmospheric turbulences, thus allowing the establishment of communications at very low elevation angles (typically 5 degrees above the horizon). This leads to an increase in the contact duration between low-earth orbiting satellites and ground stations. One of their main drawbacks is their limited data rates and the need of frequency licensing in order to avoid interferences. This will be a major issue in the upcoming years due to the increase in the number of operational satellites and constellations orbiting around the Earth.

Free space optical communications offer data rates orders of magnitude higher than current radio-frequency ones: targeted data rates go from some tens of gigabits per second to several terabits per second (Tbps). Moreover, thanks to their very narrow beam, they do not require frequency licensing and are hard to intercept by malicious observers. Finally, they offer better power efficiency compared to radio-frequency technologies and reduced payload sizes which may prove very useful for nano and micro satellites. Unfortunately, free space optical communications are still experimental and are strongly impacted by weather, clouds, and atmospheric turbulence, and thus require new technologies to be used to cope with these issues [18, 19].

In order to evaluate the impact of free space optical communications on spatial imagery systems, our work focuses on the design of *Optical Ground Station Networks (OGSN)* for low-earth orbiting observation satellites. We focus on the optical downlink and we aim to find a subset of optical ground stations in order to maximize the percentage of data downloaded from satellites, taking cloud information into account using archived data from previous decades.

Several studies were conducted to analyze and evaluate the impact of such technologies for the design of optical ground station networks. The first ones [12, 21] considered optical ground station networks for a specific deep-space mission. Their objective was to find a network that would provide a required *availability*, i.e., a percentage of time where at least one station within the network is both visible from the deep-space probe and cloud free. One of the main constraints was that only a subset of the possible stations were visible at any time due to the orbit of the probe and the rotation of the earth.

The probability of having at least one cloud-free visibility window after a given amount of time has been analyzed for a hand-made worldwide network of optical ground stations and a low-earth orbiting satellite with a given orbit [20]. Statistical results were provided assuming constant and equal cloud covers for all stations. The average possible download volumes of various optical ground stations were also analyzed using orbital information from multiple existing low-earth orbiting observation satellites and assuming average clear sky probabilities [7]. Results showed that, for the considered data rates, mid-latitude stations with an average clear sky probability of 65% could handle between 7 and 26.1 terabits of data per day, while high-latitude stations with an average clear sky probability of 55% could handle between 21.9 and 81.9 terabits.

Studies regarding the optimization of a network of optical ground stations in Europe were conducted using three years of high-resolution cloud data over Europe, Africa and the Middle East [6, 10]. In these studies, an iterative

algorithm was proposed for determining a network of optical ground stations: a subset of locations was first selected in each one-degree iso-latitude strip depending on yearly cloud-free probabilities (without orbital information). Then, this subset was shrunk using monthly and yearly statistics. Finally, only some locations were kept depending on external requirements. The resulting networks were then analyzed using orbital information from various existing low-earth orbiting satellites (identical to [7]). It was shown that mid-latitude stations could handle between 5 and 9 terabits of data per day, while high-latitude stations could not handle more than 16 terabits of data per day. While these results were less optimistic than the ones proposed in a previous paper [7], comparisons showed that even small networks of optical ground stations using low data rates (10.5 gigabits per second) could outperform radio-frequency throughputs.

In 2010, the Inter-agency Operations Advisory Group (IOAG) established the *Optical Link Study Group (OLSG)* which led to two reports in 2012 [9, 17]. In these reports, the use of free space optical communications on various space systems was evaluated. Using a *"state-of-the-art cloud database"* (from [12, 21], not available), authors found that it was possible to transfer 95% of the data acquired by a low-earth orbiting satellite to the Earth using a network of seven optical ground stations.

In this paper, we deal with low-earth orbiting satellites, nevertheless, we refer to some recent works close to our problem but regarding geostationary satellites. In this context, studies regarding the availability of an optical ground station network in Europe for a geostationary satellite were proposed in [14, 15]. A greedy algorithm and an analysis of the *SAF-NWC* high-resolution cloud database were used to find efficient networks. The need to distribute optical ground stations over larger areas was later put forward by looking at the availability of networks over Germany, Europe and "extended" Europe using a probabilistic approach taking into account cloud correlations between ground stations [4] and various optimization methods [5]. Data from the SEVIRI[1] payload were used as inputs to the following approaches [5]: select the stations with best availabilities (without taking cloud correlation into account), select the best combinations of stations by computing the availabilities of all possible combinations, or select the best network by reducing the number of possible combinations using political and infrastructural constraints, together with statistics about cloudiness and correlation. Experiments showed that while a simple complete enumeration quickly failed to give results, a correctly tuned guided approach could provide good quality solutions. Results showed that networks of stations located only in Germany were quickly limited, while European networks enhanced with stations in Africa, Middle East and South America could reach availabilities near 100%. Finally, link availability of a 77°E geostationary satellite for various networks was analyzed using cloud information retrieved from satellite images taken between October 2013 and September 2014 [1]. Results showed that networks of one, two or three stations were able to achieve respectively 74.73%, 93.7% and 97.13% availability.

While these studies deal with the optimization of optical ground station networks, none propose a formal definition of the underlying problems, or any guarantee regarding optimality, except for a complete enumeration coupled with simulations that can be computationaly expensive to run. In this work, we first propose a formal definition of `MaxPDT`, the Optical Ground Stations Network optimization problem under some assumptions, together with a mathematical formulation. We then present exact hierarchical approaches based on a dynamic programming algorithm to solve it.

The paper is organized as follows. Section 2 defines formally the `MaxPDT` problem and one of its subproblems, $\texttt{MaxPDT}_{\mathcal{L}}$, and provides some results regarding their complexities. Then, Section 3 proposes a dynamic programming algorithm to solve $\texttt{MaxPDT}_{\mathcal{L}}$ and two exact hierarchical approaches combining this algorithm with different enumeration methods to solve the `MaxPDT` problem. Computational results regarding `MaxPDT` and $\texttt{MaxPDT}_{\mathcal{L}}$ are then presented in Section 4. Finally, conclusion and future research directions are discussed in Section 5.

[1] The *Spinning Enhanced Visible and Infrared Imager (SEVIRI)* is an optical imaging radiometer on-board *Meteosat Second Generation (MSG)* satellites.

2 | THE MAXIMUM PERCENT DATA TRANSFERRED PROBLEM

In this section, we explain the assumptions we consider to define the *Maximum Percent Data Transferred* (`MaxPDT`) problem. The `MaxPDT` problem is based on download points, which are simplification of visibility windows between the satellite and the optical ground stations. We first explain how we transform visibility windows into download points, then we formally define the `MaxPDT` problem and one of its subproblems that we will later use to solve `MaxPDT` and derive its complexity.

2.1 | Industrial context and assumptions

Considering a low-earth orbiting satellite and a given a set $\mathcal{R} = \{r_1, \ldots, r_N\}$ of possible locations for optical ground stations with associated costs p_r ($r \in \mathcal{R}$), we aim to find a subset $\mathcal{R}^* \subseteq \mathcal{R}$ having a total cost lower than K that **maximizes** the *Percent Data Transferred* (PDT) [9], i.e., the percentage of data acquired by the satellite that can be successfully transferred to the Earth.

We assume the satellite has a buffer of size $B \geq 0$ that is empty at the beginning of the time horizon and must be empty at the end. We assume that the time horizon $\mathcal{H} = [T_{start}, T_{end}]$ is divided into a set $\mathcal{S} = \{s_1, \ldots, s_M\}$ of successive *acquisition slots*, and that a given amount of data $a^s > 0$ is acquired at the beginning t^s of each slot $s \in \mathcal{S}$. There is no gap between two successive acquisition slots, thus the end time of slot s_i is the beginning time $t^{s_{i+1}}$ of slot s_{i+1}. By definition we have $t^{s_1} = T_{start}$ and for simplicity we assume $t^{s_{M+1}} = T_{end}$.

While orbiting around the Earth, the satellite is able to reach intermittently the various locations (of optical ground stations) during *visibility windows*. We define $\mathcal{V}$ as the set of all visibility windows. To each visibility window $v \in \mathcal{V}$ is associated a start $t_v^{sta} \geq T_{start}$, an end $t_v^{end} < T_{end}$, a unique location (or station) $\tau_v \in \mathcal{R}$, a data rate function $d_v : \left[t_v^{sta}, t_v^{end}\right] \to \mathbb{R}^+$ and a set γ_v of overlapping visibility windows:

$$\gamma_v = \left\{v' \in \mathcal{V} : t_v^{sta} \leq t_{v'}^{sta} < t_v^{end} \quad \text{or} \quad t_v^{sta} \leq t_{v'}^{end} < t_v^{end}\right\}$$

Two overlapping visibility windows cannot be both used for downloading data and are thus in mutual exclusion. For example, let us consider the instance presented in Figure 1 which consists of four slots and a network of three stations with eight visibility windows (*A* to *G*). Each horizontal line represents a different station with its set of visibility windows. Vertical dotted black lines represent the beginning of slots, with the acquisition volumes indicated above. The data rate functions of the visibility windows are not shown in the figure. In this example, some visibility windows are overlapping (such as *A* and *B* or *D* and *E*) and thus cannot be both used.

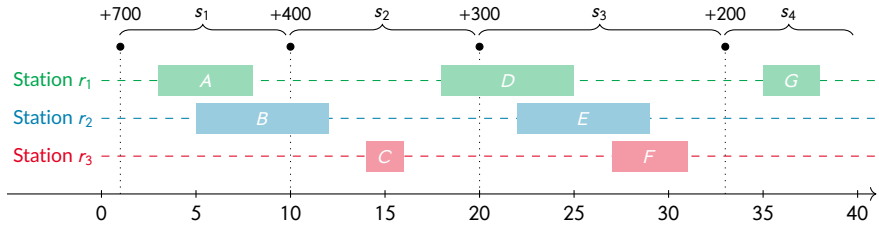


FIGURE 1 Example of an instance of the considered problem.

Characteristics of optical links during communications between satellites and optical ground stations are not well

known and multiple parameters, mainly cloud interferences, may influence the established link during a visibility window, thus creating very complicated data rate functions d_v. Furthermore, considered horizons $\mathcal{H}$ are very large (multiple decades) while slots and visibility windows require a granularity of the order of tens of minutes. To consider the difference in time steps, we assume that each visibility window can be reduced to an instantaneous download called a **download point**. We then consider a mapping $f_{v\to w}$ between each visibility window v and each download point w. For each visibility window $v \in \mathcal{V}$, the mapping function creates one download point $w = f_{v\to w}(v) = (\sigma_w, \tau_w, \rho_w, \gamma_w)$, defined as follows:

- σ_w is the slot within which the visibility window starts, i.e., σ_w is the **only** slot $s_i \in \mathcal{S}$ such that:

$$t^{s_i} \leq t_v^{sta} < t^{s_{i+1}}$$

- τ_w is the station associated to the visibility window, i.e., $\tau_w = \tau_v$;
- ρ_w is the download volume associated to the visibility window, which is computed beforehand using the data rate function d_v:

$$\rho_w = \int_{t_v^{sta}}^{t_v^{end}} d_v(t)dt$$

- γ_w is the set of download points **conflicting** with w, i.e., the set of download points associated to visibility windows overlapping v:

$$\gamma_w = \left\{ f_{v\to w}(v') \mid v' \in \gamma_v \right\}$$

We say that two download points are **in conflict** if their associated visibility windows are **overlapping**.

Data associated to the visibility windows of the example shown in Figure 1 are given on the left side of Table 1. On the right side, we provide the mapping of these data to obtain the associated download points.

We denote by Q the set of all download points:

$$Q = \{ f_{v\to w}(v) \ : \ v \in \mathcal{V} \}$$

For each slot $s \in \mathcal{S}$, we denote by Q^s the set of download points inside s:

$$Q^s = \{ w \in Q \ : \ \sigma_w = s \}$$

Similarly, for each location $r \in \mathcal{R}$, we denote by Q_r the set of download points associated with r:

$$Q_r = \{ w \in Q \ : \ \tau_w = r \}$$

Considering the example in Table 1, we obtain the following sets:

$$Q = \{ A_w, B_w, C_w, D_w, E_w, F_w, G_w \}$$

Visibility windows, $v \in \mathcal{V}$						Download points, $w = f_{v \to w}(v)$				
v	t_v^{sta}	t_v^{end}	τ_v	d_v	γ_v	w	σ_w	τ_w	ρ_w	γ_w
A	3	8	r_1	μ	$\{B\}$	A_w	s_1	r_1	100	$\{B_w\}$
B	5	12	r_2	μ	$\{A\}$	B_w	s_1	r_2	140	$\{A_w\}$
C	14	16	r_3	μ	$\emptyset$	C_w	s_2	r_3	40	$\emptyset$
D	18	25	r_1	μ	$\{E\}$	D_w	s_2	r_1	140	$\{E_w\}$
E	22	29	r_2	μ	$\{D, F\}$	E_w	s_3	r_2	140	$\{D_w, F_w\}$
F	27	31	r_3	μ	$\{E\}$	F_w	s_3	r_3	80	$\{E_w\}$
G	35	38	r_1	μ	$\emptyset$	G_w	s_4	r_1	60	$\emptyset$

TABLE 1 Example of mapping from visibility windows to download points for the instance shown in Figure 1. In this example, we consider a data rate function μ identical for all visibility windows: $\forall t \in [T_{start}, T_{end}],\ \mu(t) = 20$.

$$Q^{s_1} = \{A_w, B_w\}, \quad Q^{s_2} = \{C_w, D_w\}, \quad Q^{s_3} = \{E_w, F_w\}, \quad Q^{s_4} = \{G_w\}$$
$$Q_{r_1} = \{A_w, D_w, G_w\}, \quad Q_{r_2} = \{B_w, E_w\}, \quad Q_{r_3} = \{C_w, F_w\}$$

It can be noticed that sets Q^s define a partition of Q ($\bigcup_{s \in \mathcal{S}} Q^s = Q$ and $\forall s_i, s_j \in \mathcal{S},\ s_i \neq s_j \Rightarrow Q^{s_i} \cap Q^{s_j} = \emptyset$). Sets Q_r also define a partition of Q.

Based on the assumption that visibility windows can be reduced to instantaneous downloads (download points), optimizing a ground station network can be re-formulated as two interleaved selection problems. The first problem corresponds to the selection of ground station locations and the second problem corresponds to the selection of download points. The objective is to maximize the percent data transferred from the satellite to the Earth, and can be computed using a set of recursive equations depending on the set of chosen download points that are given later in this paper.

2.2 | The `MaxPDT` problem

The `MaxPDT` problem consists in selecting a subset $\mathcal{R}' \subseteq \mathcal{R}$ of locations and a subset $Q' \subseteq Q$ of download points, in order to maximize the *Percent Data Transferred (PDT)* from the satellite to the Earth.

Definition Instance of the `MaxPDT` problem

An instance of the `MaxPDT` problem is a 5-tuple $(K,\ B,\ \mathcal{R},\ \mathcal{S},\ Q)$ where $K > 0$ is the maximum cost allowed for opening stations, $B > 0$ is the size of the satellite buffer, $\mathcal{R}$ ($|\mathcal{R}| = N$) is the set of available locations, $\mathcal{S}$ ($|\mathcal{S}| = M$) is the set of slots and Q is the set of download points.

For each location $r \in \mathcal{R}$, $p_r \geq 0$ is the price of opening a station at r and $Q_r \subseteq Q$ is the set of download points associated with r.

For each slot $s \in \mathcal{S}$, $a^s \in [0, B]$ is the amount of data acquired at the beginning of s and $Q^s \subseteq Q$ is the set of download points associated with s.

For each download point $w \in Q$, $\tau_w \in \mathcal{R}$ is the location of w, $\sigma_w \in \mathcal{S}$ is the slot of w, $\rho_w \geq 0$ is the amount of data that can be downloaded for w and $\gamma_w \subset Q$ is the set of download points in conflict with w.

By construction, each download point $w \in Q$ is associated to a single location and a single slot:

$$\forall s_1, s_2 \in \mathcal{S} \ : \ Q^{s_1} \cap Q^{s_2} = \varnothing$$

$$\forall r_1, r_2 \in \mathcal{R} \ : \ Q_{r_1} \cap Q_{r_2} = \varnothing$$

Definition Feasible solution for the `MaxPDT` problem

Given an instance $(K,\ B,\ \mathcal{R},\ \mathcal{S},\ Q)$ of the `MaxPDT` problem, a feasible solution is a pair $(\mathcal{R}', Q')$ with $\mathcal{R}' \subseteq \mathcal{R}$ and $Q' \subseteq Q$ such that:

$$\sum_{r \in \mathcal{R}'} p_r \leq K \tag{2a}$$

$$Q' \subseteq \bigcup_{r \in \mathcal{R}'} Q_r \tag{2b}$$

$$\forall w \in Q', \quad \gamma_w \cap Q' = \varnothing \tag{2c}$$

Equation (2a) indicates that the total costs of the chosen locations (opened stations) is lower than the maximum K. Equation (2b) enforces that download points can only be used if their associated locations are chosen. Equation (2c) indicates that if a download point w is chosen, none of the download points in conflict with w are selected.

Definition Data loss of a solution for the `MaxPDT` problem

Given a solution $(\mathcal{R}', Q')$ of the `MaxPDT` problem, the amount of data loss associated is the amount of data acquired by the satellite that has not been successfully transferred to the Earth. Data loss occurs when there is not enough space in the buffer to store the data acquired by the satellite at the beginning of a slot.

The following equations describe formally the computation of the amount of data loss. Given a set $\mathcal{S}$ of M slots, a set of chosen download points Q' and a buffer size B:

$$losses(Q',\ \mathcal{S},\ B) = b^{s_M} + \sum_{s \in \mathcal{S}} l^s \tag{3}$$

where l^s is the amount of data loss *during* the slot s and b^s the amount of data in the buffer *at the end* of the slot s. By convention, b^{s_0} is the amount of data at the beginning of the temporal horizon. We assume that the amount of data in the buffer at the end of the temporal horizon, b^{s_M}, is lost. These can be computed using the following recursive equations:

$$b^{s_0} = 0 \tag{4a}$$

$$b^{s_i} = \max(0, \min(b^{s_{i-1}} + a^{s_i}, B) - \sum_{w \in Q^{s_i} \cap Q'} \rho_w), \qquad i \in \{1, \ldots, M\} \tag{4b}$$

$$l^{s_i} = \max(0, b^{s_{i-1}} + a^{s_i} - B), \qquad i \in \{1, \ldots, M\} \tag{4c}$$

At the beginning of the temporal horizon, the buffer is empty (4a). At the end of a slot s, the amount of data in the buffer is the amount of data in the buffer at the end of the previous slot, to which we add the acquisition volume of slot s and subtract the amount of data downloaded during the slot (4b). The amount of data loss during a slot s is the amount of data acquired at the beginning of the slot a^s that did not fit in the buffer (4c).

Definition Percentage of Data Transferred (PDT) for a `MaxPDT` solution

Given a solution $(\mathcal{R}', Q')$ of the `MaxPDT` problem, the percentage of data transferred is the amount of data acquired by the satellite during the time horizon that has been successfully downloaded:

$$pdt(Q', \mathcal{S}, B) = 1 - \frac{losses(Q', \mathcal{S}, B)}{\sum_{s \in \mathcal{S}} a^s} \tag{5}$$

The objective of the `MaxPDT` problem is to find a feasible solution $(\mathcal{R}', Q')$ which maximizes the percent data transferred, which is the same as minimizing the losses $losses(Q', \mathcal{S}, B)$.

2.2.1 | Mathematical model

To model the `MaxPDT` problem, we first consider two types of binary variables for the selection of stations and download points. For each station $r \in \mathcal{R}$, we define a binary variable $y_r = 1$ if and only if the station r is chosen. For each download point $w \in Q$, we define a binary variable $x_w = 1$ if and only if the download point w is used to download data.

Moreover, we introduce two real variables for each slot $s \in \mathcal{S}$ to model the amount of data in the buffer and the amount of data loss (similar to the intermediate variables used in Definition 3): $b^s \in \mathbb{R}^+$ represents the amount of data in the buffer *at the end of* slot s and $l^s \in \mathbb{R}^+$ represents the amount of data loss *during* slot s.

Using this set of variables and the parameters given in Definition 2.2 (summarized in Table 2), the mathematical model for the `MaxPDT` problem can be formulated as follows:

$$\text{min.} \quad \sum_{s \in \mathcal{S}} l^s \tag{6a}$$

$$\text{s.t.} \quad x_w \leq y_r, \qquad r \in \mathcal{R}, \quad w \in Q_r \tag{6b}$$

$$x_w + x_{w'} \leq 1, \qquad w \in Q, \; w' \in \gamma_w \tag{6c}$$

$$b^{s_i} + l^{s_i} \geq b^{s_{i-1}} + a^{s_i} - \sum_{w \in Q^{s_i}} x_w \rho_w, \qquad s_i \in \mathcal{S} \tag{6d}$$

$$0 \leq b^{s_i} \leq B - a^{s_{i+1}}, \qquad s_i \in \mathcal{S}, \; i \neq M \tag{6e}$$

$$b^{s_0} = b^{s_M} = 0 \tag{6f}$$

$$\sum_{r \in \mathcal{R}} p_r y_r \leq K \tag{6g}$$

$$x_w \in \{0,1\}, \qquad w \in Q \tag{6h}$$

$$y_r \in \{0,1\}, \qquad r \in \mathcal{R} \tag{6i}$$

$$b^s \geq 0, \quad l^s \geq 0, \qquad s \in \mathcal{S} \tag{6j}$$

Objective (6a) minimizes the amount of data loss. Constraints (6b) and (6c) prevent downloads on stations that are not chosen ($y_r = 0$) and on conflicting download points. Constraints (6d) and (6e) force the amount of data at the end of a slot s_i to be consistent with the amount at the beginning of s_i and s_{i+1}, and to be less than the buffer size B minus the acquisition of slot s_{i+1} (i.e., at the end of slot s_i, there must be at least $a^{s_{i+1}}$ free space in the buffer). Constraint (6f) says that the initial amount of data in the buffer is 0 and forces the final amount of data in the buffer to be 0. Constraint (6g) forces the total cost of the network to be less than the maximum cost allowed K. Constraints (6h)–(6j) define the domain of the decision variables.

2.3 | The download point selection problem, $\texttt{MaxPDT}_{\mathcal{L}}$

The MaxPDT problem can grow very quickly in size, especially when we consider very large horizon (multiple decades). In order to tackle it more efficiently, we chose to first focus on a subproblem: the selection of the download points. The next section will be dedicated to the analysis of this problem, which we called $\texttt{MaxPDT}_{\mathcal{L}}$. We will first formally define it from MaxPDT, and then provide some complexity results that can be extended to the MaxPDT problem.

Definition Instance of the $\texttt{MaxPDT}_{\mathcal{L}}$ problem

An instance of the $\texttt{MaxPDT}_{\mathcal{L}}$ problem is a triplet $(B,\ \mathcal{S},\ Q)$ where B, $\mathcal{S}$ and Q have the same meaning as for the MaxPDT problem (see Definition 2.2).

An instance of the $\texttt{MaxPDT}_{\mathcal{L}}$ problem is an instance of the MaxPDT problem where the selection of locations has been removed.

Definition Feasible solution for the $\texttt{MaxPDT}_{\mathcal{L}}$ problem

Given an instance $(B,\ \mathcal{S},\ Q)$ of the $\texttt{MaxPDT}_{\mathcal{L}}$ problem, a feasible solution is a subset of download points $Q' \subseteq Q$ (7a) such that no two download points are in conflict (7b).

$$Q' \subseteq Q \tag{7a}$$

$$\forall w \in Q', \quad \gamma_w \cap Q' = \varnothing \tag{7b}$$

Definition Optimal solution for the $\texttt{MaxPDT}_{\mathcal{L}}$ problem

Given an instance $(B,\ \mathcal{S},\ Q)$ of the $\texttt{MaxPDT}_{\mathcal{L}}$ problem, an optimal solution is a feasible solution with minimum losses.

An optimal solution for $\texttt{MaxPDT}_{\mathcal{L}}$ is similar to an optimal solution for MaxPDT. An optimal solution for the MaxPDT problem can be found by solving a $\texttt{MaxPDT}_{\mathcal{L}}$ problem for each feasible subset of locations, and then taking the best solution found.

2.3.1 | Mathematical model

The model for the $\texttt{MaxPDT}_{\mathcal{L}}$ problem can be obtained by removing variables y_r and constraints (6b) and (6g) from the model of the MaxPDT problem (6).

Inputs of the problem			
T_{start}	Start of the temporal horizon.	T_{end}	End of the temporal horizon.
$\mathcal{R}$	Set of possible locations.	N	Number of possible locations.
p_r	Cost of opening a station on location r.	K	Maximum allowed cost for the network of stations.
B	Size of the buffer.		
$\mathcal{S}$	Set of slots.	M	Number of slots.
t^s	Start of slot s.	a^s	Amount of data acquired at the beginning of slot s.
$\mathcal{V}$	Set of visibility windows.	τ_v	Location reachable during visibility window v.
t_v^{sta}	Start of visibility window v.	t_v^{end}	End of visibility window v.
d_v	Data rate function (evolution) during visibility window v.	γ_v	Set of visibility windows overlapping with visibility window v.
$f_{v\to w}$	Mapping function between visibility and download points.	Q	Set of download points.
Q^s	Set of download points associated with slot s.	Q_r	Set of download points associated with location r.
σ_w	Slot of download point w.	τ_w	Location of download point w.
ρ_w	Amount of data that can be downloaded using w.	γ_w	Set of download points conflicting with w.
Decision variables			
y_r	$y_r = 1$ if and only if location r is chosen.	x_w	$x_w = 1$ if and only if download point w is selected.
b^s	Amount of data in the buffer at the end of slot s.	l^s	Amount of data lost during slot s.

TABLE 2 Summary of notations for the definition of the MaxPDT and $\texttt{MaxPDT}_{\mathcal{L}}$ problems.

2.4 | Complexity results

In this section, we give some complexity results for the MaxPDT and $\texttt{MaxPDT}_{\mathcal{L}}$ problems. We then provide results for special cases of instances of the $\texttt{MaxPDT}_{\mathcal{L}}$ problem regarding the distribution of download points and conflicts between them.

Proposition 1 *The $\texttt{MaxPDT}_{\mathcal{L}}$ problem is strongly NP-hard.*

Proof Let us consider the decision variant of $\texttt{MaxPDT}_{\mathcal{L}}$: is it possible to find a solution Q' such that $losses(Q') \leq \phi$, ϕ being an arbitrary positive real value?

The proof is based on the reduction from the Weighted Independent Set problem (WIS), which is known to be NP-complete in the strong sense. The weighted independent set problem consists, given a graph $\mathcal{G} = (V, E)$ and weights $u : V \to \mathbb{Z}$, in finding a subset $S \subseteq V$ of vertices such that no two vertices in S are adjacent and such that the sum of the weights of vertices in S is greater than an arbitrary positive value Φ.

Obviously, $\texttt{MaxPDT}_{\mathcal{L}}$ is NP since, given a solution Q', $losses(Q')$ can be computed in linear time using the formula given in (3).

From a WIS instance $(\mathcal{G} = (V, E), \Phi)$, we build up an instance of $\texttt{MaxPDT}_{\mathcal{L}}$ in the following way: the instance contains a single slot s_1 ($\mathcal{S} = \{s_1\}$) with an associated acquisition volume $a^{s_1} = \sum_{v \in V} u_v$ and a set of download points $Q = Q^{s_1}$. The set Q contains a download point w_v for each vertex $v \in V$ with $\sigma_{w_v} = s_1$, $\rho_{w_v} = u_v$ and $\gamma_{w_v} = \{w_{v'} : (v, v') \in E\}$, meaning that two download points are in conflict if their corresponding vertices are connected in the graph $\mathcal{G}$. The buffer size is $B = a^{s_1}$.

A feasible solution Q' for this instance of $\texttt{MaxPDT}_{\mathcal{L}}$ is obviously a feasible solution S for the WIS instance due to the conflict constraints. Furthermore, finding a solution S_Φ such that $\sum_{v \in S_\Phi} u_v \geq \Phi$ is the same as finding a solution Q'_ϕ of the $\texttt{MaxPDT}_{\mathcal{L}}$ instance, with $\phi = B - \Phi$ such that $losses(Q'_\phi, \{s_1\}, B) \leq \phi = B - \Phi$ since from (3):

$$
\begin{aligned}
losses(Q'_\phi, \{s_1\}, B) &= l^{s_1} + b^{s_1} \\
&= \max(0, b^{s_0} + a^{s_1} - B) + b^{s_1} \\
&= b^{s_1} \\
&= \max(0, \min(b^{s_0} + a^{s_1}, B) - \sum_{w \in Q^{s_1} \cap Q'} \rho_w) \\
&= \max(0, a^{s_1} - \sum_{w \in Q'} \rho_w) \\
&= a^{s_1} - \sum_{w \in Q'} \rho_w \\
&= \sum_{v \in V} u_v - \sum_{v \in Q'} u_v = B - \Phi
\end{aligned}
$$

∎

Corollary 2 *The MaxPDT problem is strongly NP-hard.*

Proof The $\texttt{MaxPDT}_{\mathcal{L}}$ problem is a subproblem of the MaxPDT problem where stations have already been chosen. So it is thus trivial to reduce $\texttt{MaxPDT}_{\mathcal{L}}$ to MaxPDT, making MaxPDT strongly NP-hard. ∎

2.4.1 | Special cases

We propose here some complexity results for $\texttt{MaxPDT}_{\mathcal{L}}$ regarding instances with special distributions of download points and conflicts between download points.

Definition We define the class of **intra** instances as the class of instances within which there are no conflicts between download points that are not in the same slot, i.e., given an instance $(B,\ \mathcal{S},\ Q)$ of the **intra** class:

$$\forall w \in Q\ :\ \gamma_w \subset Q^{\sigma_w}$$

Definition We define the class of **interval** instances as the class of instances within which the set of conflicts represents intersection constraints between intervals, similar to interval or intersection graph.

Proposition 3 *Solving instances of $\texttt{MaxPDT}_{\mathcal{L}}$ that are in **both** the intra and the interval classes can be done in polynomial time.*

Proof Since there are no conflicts between download points not in the same slot, an optimal solution is a solution where the amount of data downloaded within each slot is maximized. Within any slot s, selecting a subset of stations maximizing the amount of data downloaded is the same as finding a maximum-weighted independent set in an interval graph with $|Q^s|$ vertices (see the reduction above). This can be done in $O(|Q^s|)$ time [8] if the download points are correctly sorted. Thus, finding the optimal solution can be done in $O(\sum_{s\in\mathcal{S}} |Q^s|) = O(|Q|)$ time if the download points are correctly sorted. Since sorting all the download points can be done in $O(|Q|\,.\log|Q|)$ time, the final complexity of the algorithm for instances in both the intra and the interval classes is $O(|Q|\,.\log|Q|)$. ∎

It is worth noticing that instances of the $\texttt{MaxPDT}_{\mathcal{L}}$ problem constructed from real scenarios fall within the **interval** class. Some of these might correspond to **intra** instances — this will mostly depend on the chosen stations and orbit of the satellite — but this will not be the general case.

3 | HIERARCHICAL APPROACH

In this section we will present hierarchical approaches to solve the MaxPDT problem based on a dynamic programming algorithm for solving the $\texttt{MaxPDT}_{\mathcal{L}}$ problem.

In real instances, the number N of possible locations for the stations is often very small (some tens) and the temporal horizon is large (some years). We propose to separate the decision process into two cooperative algorithms: a *master* algorithm that enumerates all possible subsets of stations $\mathcal{R}'$, and a *slave* algorithm that solves an instance of $\texttt{MaxPDT}_{\mathcal{L}}$ build from each subset.

3.1 | Algorithms for the enumeration of networks of ground stations

We propose two algorithms to enumerate the feasible subsets of locations. The first one is an exhaustive enumeration and the second algorithm is a branch-and-bound. Both of these algorithms use algorithm $\mathcal{A}$ to compute the maximum *PDT* of the selected networks. Algorithm $\mathcal{A}$ solves a $\texttt{MaxPDT}_{\mathcal{L}}$ instance and returns a list of selected download points (Q') together with the corresponding *PDT*. Such an algorithm is presented in Section 3.2. Table 3 gives a summary of notations used in the proposed algorithms.

In order to obtain an instance of $\texttt{MaxPDT}_{\mathcal{L}}$ from an instance of MaxPDT for a subset of locations, we define the following operation:

Definition Projection of a set $\mathcal{S}$ of slots on a set $\mathcal{R}'$ of stations: $\mathcal{S} \downarrow \mathcal{R}'$

We define the projection of a set $\mathcal{S}$ of slots on a set $\mathcal{R}'$ of stations as an updated set $\mathcal{S}_P = \mathcal{S} \downarrow \mathcal{R}'$ where download points on stations not in $\mathcal{R}'$ have been removed. Given two related slots $s \in \mathcal{S}$ and $s^P \in \mathcal{S}_P$, the following hold:

$$a^{s^P} = a^s$$
$$Q^{s^P} = Q^s \cap Q_{\mathcal{R}'}$$

where $Q_{\mathcal{R}'}$ is the set of all download points on station in $\mathcal{R}'$:

$$Q_{\mathcal{R}'} = \bigcup_{r \in \mathcal{R}'} Q_r$$

The selection of feasible subsets of stations and the projection can be respectively seen as the enforcement of constraints (6g) and (6b) of the original MILP.

3.1.1 | Exhaustive enumeration

The first algorithm $\mathcal{EE}_{pdt}(\mathcal{A})$ is a simple exhaustive enumeration that tries every possible combinations of stations and uses algorithm $\mathcal{A}$ to compute the *PDT* of these combinations. Algorithm 1 gives an overview of $\mathcal{EE}_{pdt}(\mathcal{A})$:

Algorithm 1 Exhaustive enumeration algorithm

$pdt_{\max} \leftarrow 0, \quad \mathcal{R}^* \leftarrow \varnothing, \quad Q^* \leftarrow \varnothing$
for each $\mathcal{R}' \subseteq \mathcal{R}$ **do**
  if $\sum_{r \in \mathcal{R}'} p_r \leq K$ **then**
    $\mathcal{S}_P \leftarrow \mathcal{S} \downarrow \mathcal{R}'$
    $(pdt,\ Q') \leftarrow \mathcal{A}\,(B,\ \mathcal{S}_P,\ \bigcup_{r \in \mathcal{R}'} Q_r)$
    if $pdt > pdt_{\max}$ **then**
      $pdt_{\max} \leftarrow pdt$
      $\mathcal{R}^* \leftarrow \mathcal{R}'$
      $Q^* \leftarrow Q'$
    end if
  end if
end for

3.1.2 | Branch-and-bound algorithm

The second algorithm $\mathcal{BB}_{pdt}(\mathcal{A})$ we propose is a binary branch-and-bound algorithm that uses $\mathcal{A}$ to compute upper bounds on nodes and objective value of solutions.

In this algorithm, branching is done by imposing or forbidding the opening of a station at a possible location for which no decision has been made yet. On any given node, we split the set of stations $\mathcal{R}$ into three disjoint subsets $\mathcal{R}^+$, $\mathcal{R}^-$ and $\mathcal{R}^?$ corresponding respectively to chosen stations, not chosen ones and still undefined ones. To compute the lower bound on a given node, we use algorithm $\mathcal{A}$ with a projection of the slots on $\mathcal{R}^+ \cup \mathcal{R}^?$ — Stations still undefined are considered chosen since opening stations can only increase the objective value.

A *leaf* node is a node on which it is not possible to add any new stations to $\mathcal{R}^+$ without exceeding the maximum allowed cost:

$$leaf(\mathcal{R}^+, \mathcal{R}^-, \mathcal{R}^?) \Leftrightarrow \left(\forall r \in \mathcal{R}^?, \quad p_r + \sum_{r' \in \mathcal{R}^+} p_{r'} > K \right)$$

While processing a node, we branch on the station providing the largest amount of download. Given a solution Q' obtained using $\mathcal{A}$ for a projection of the slots on $\mathcal{R}^+ \cup \mathcal{R}^?$, we choose the station r^{next} such that:

$$r^{next} = \arg\max_{r \in \mathcal{R}^?} \sum_{w \in Q_r \cap Q'} \rho_w$$

We initialize our search tree with a single node with $\mathcal{R}^+ = \mathcal{R}^- = \emptyset$ and $\mathcal{R}^? = \mathcal{R}$. Nodes on the tree are processed in increasing order of their lower bounds.

Exhaustive Enumeration $\mathcal{EE}_{pdt}(\mathcal{A})$			
$\mathcal{S} \downarrow \mathcal{R}'$	Projection of $\mathcal{S}$ on a set $\mathcal{R}'$ of stations.	$pdt_{\max}$	Current maximum *PDT* up to now.
$\mathcal{R}^*$	Current best selection of stations.	Q^*	Current best selection of download points.
Branch & Bound algorithm $\mathcal{BB}_{pdt}(\mathcal{A})$			
$\mathcal{R}^+$	Set of already chosen stations.	$\mathcal{R}^-$	Set of already discarded stations.
$\mathcal{R}^?$	Set of yet undecided stations.		

TABLE 3 Summary of notations used in $\mathcal{EE}_{pdt}(\mathcal{A})$ and $\mathcal{BB}_{pdt}(\mathcal{A})$. Here $\mathcal{A}$ is the algorithm used to solve the $\texttt{MaxPDT}_{\mathcal{L}}$ problem for a selected set of locations.

3.2 | Dynamic programming algorithm definition to solve $\texttt{MaxPDT}_{\mathcal{L}}$

In the following, we present a dynamic programming algorithm, $DP_{pdt_{\mathcal{L}}}$ that can be used to solve $\texttt{MaxPDT}_{\mathcal{L}}$ for any class of instances. This algorithm can be used within the methods presented in Section 3.1.

The $DP_{pdt_{\mathcal{L}}}$ algorithm manages a set $\mathcal{H}$ of labels, where each label h is a tuple $(b_h,\ I_h,\ \Gamma_h,\ \mathcal{W}_h)$ with b_h the current amount of data in the buffer, I_h the accumulated amount of data loss, Γ_h the set of conflicting download points (i.e., the set of download points that cannot extend this label) and $\mathcal{W}_h$ the list of used download points.

The algorithm proceeds with the following steps:

1. a label $h_0 = (0,\ 0,\ \varnothing,\ \varnothing)$ is created and added to the initially empty set of labels $\mathcal{H}$;
2. the slots in $\mathcal{S}$ are processed in increasing starting time order. For each slot $s \in \mathcal{S}$, the following actions are performed:
   a. all existing labels are updated to take into account the amount of data acquired at the beginning of s;
   b. for each download point $w \in Q^s$ and for each existing label $h \in \mathcal{H}$, a new label h' is created if the download point is not in conflict with the label h ($w \notin \Gamma_h$);
3. labels in $\mathcal{H}$ are updated to take into account constraint (6f) (the data in the buffer at the end of the temporal horizon is lost) and the label with the minimum amount of data loss at the end is chosen.

Algorithm 2 shows a formal description of the algorithm.

3.2.1 | Dominance rule to prune labels

In order to prevent the set of labels $\mathcal{H}$ from growing too large, we need to remove labels that cannot lead to optimal solutions by means of a dominance rule between two labels.

Definition We say that a label h_1 dominates a label h_2 if $h_1 \neq h_2$ and:

$$\Gamma^+_{h_1} = \Gamma^+_{h_2} \quad and \begin{pmatrix} & b_{h_1} < b_{h_2} \ \wedge\ l_{h_1} \leq l_{h_2} & (8a)\\ or & b_{h_1} = b_{h_2} \ \wedge\ l_{h_1} < l_{h_2} & (8b)\\ or & b_{h_1} = b_{h_2} \ \wedge\ l_{h_1} = l_{h_2} \ \wedge\ \mathcal{W}_{h_1} \prec \mathcal{W}_{h_2} & (8c) \end{pmatrix}$$

where Γ^+_h is a subset of Γ_h containing only download points that have not yet been processed. This is due to the fact that labels can be compared only if they have the same sets of conflicts in the future.

Conditions (8a) and (8b) compare labels according to the amount of data loss (current objective value) and amount of data in the buffer. If there are less data loss and less data in the buffer in h_1 than in h_2, the solution for h_1 is better than the one for h_2. Since $\Gamma^+_{h_1} = \Gamma^+_{h_2}$, any choice possible for extending h_2 is also possible for h_1, thus h_1 dominates h_2. In fact, the only constraints for extension come from conflicting download points, so if h_1 is better than h_2, there will be at least one solution made from extending h_1 that will be better than any solution created by extending h_2.

Condition (8c) is only used to avoid having solutions with identical objective values: two solutions may have the same amount of data loss and amount of data in the buffer, keeping both of them would be inefficient, so we remove the one with the worst set of used download points ($\prec$ must be a **strict total order**).

Pruning of labels is done within step 2.2, at the beginning of the outer `for` loop (between line 7 and 8). This dominance rule guarantees that no label than can lead to an optimal solution will be pruned, insuring optimality of the algorithm. Furthermore, it guarantees that if all labels $h \in \mathcal{H}$ have an empty set of conflicts ($\Gamma_h = \varnothing$) and an empty buffer $b_h = 0$, they can be compared using (8b)–(8c) and a single one dominates all the other. This insures that at the end of the algorithm, a single label will remain.

We call the combinations of the $DP_{pdt_{\mathcal{L}}}$ with the above defined dominance rule $DP^+_{pdt_{\mathcal{L}}}$.

Algorithm 2 Dynamic programming algorithm $DP_{pdt_{\mathcal{L}}}$

$\mathcal{H} \leftarrow \{(0,\ 0,\ \varnothing,\ \varnothing)\}$ ▷ step 1
for each $s \in \mathcal{S}$ **do** ▷ step 2
  for each $h \in \mathcal{H}$ **do** ▷ step 2.1
    $b_h \leftarrow \min(b_h + a^s,\ B)$
    $l_h \leftarrow l_h + \max(0,\ b_h + a^s - B)$
  end for
  for each $w \in Q^s$ **do** ▷ step 2.2
    $\mathcal{H}^+ \leftarrow \varnothing$
    for each $h \in \mathcal{H}$ **do**
      if $w \notin \Gamma_h$ **then**
        $h' \leftarrow (\max(0,\ b_h - \rho_w),\ l_h,\ \mathcal{W}_h \cup \{w\},\ \Gamma_h \cup \gamma_w)$
        $\mathcal{H}^+ \leftarrow \mathcal{H}^+ \cup \{h'\}$
      end if
    end for
    $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{H}^+$
  end for
end for
for each $h \in \mathcal{H}$ **do** ▷ step 3
  $l_h \leftarrow l_h + b_h$
  $b_h \leftarrow 0$
end for

3.3 | Example of execution of the $DP^+_{pdt_{\mathcal{L}}}$ algorithm

In this section, we will carry out an example of $DP^+_{pdt_{\mathcal{L}}}$ on the instance presented in Section 2.1. We assume here that the three stations have been selected and that the buffer size is 1000. At each step of the algorithm, we display the current list of labels $\mathcal{H}$ in a table.

3.3.1 | Step 1

We start by initializing $\mathcal{H}$ with the initial label:

	b_h	l_h	$\mathcal{W}_h$	Γ_h
h_1	0	0	$\varnothing$	$\varnothing$

3.3.2 | Step 2

We start processing the first slot s_1 ($a^{s_1} = 700$) by updating all labels inside $\mathcal{H}$ according to step 2.1:

	b_h	l_h	$\mathcal{W}_h$	Γ_h
h_1	700	0	$\varnothing$	$\varnothing$

We process the first download point A_w ($\rho_{A_w} = 100$ and $\gamma_{A_w} = \{B_w\}$). This creates a new label h_2 by extending h_1:

	b_h	l_h	$\mathcal{W}_h$	Γ_h
h_1	700	0	$\varnothing$	$\varnothing$
h_2	600	0	$\{A_w\}$	$\{B_w\}$

We process the second download point B_w ($\rho_{B_w} = 140$ and $\gamma_{B_w} = \{A_w\}$). Since Γ_{h_2} contains B_w, we can only extend h_1 and create a new label h_3:

	b_h	l_h	$\mathcal{W}_h$	Γ_h
h_1	700	0	$\varnothing$	$\varnothing$
h_2	600	0	$\{A_w\}$	$\{B_w\}$
h_3	560	0	$\{B_w\}$	$\{A_w\}$

We start processing the second slot s_2 ($a^{s_2} = 400$) and update all labels inside $\mathcal{H}$. Since the buffer size is limited to 1000, some data are lost when updating h_1 ($b_{h_1} + a^{s_2} = 700 + 400 > 1000$):

	b_h	l_h	$\mathcal{W}_h$	Γ_h
h_1	1000	100	$\varnothing$	$\varnothing$
h_2	1000	0	$\{A_w\}$	$\{B_w\}$
h_3	960	0	$\{B_w\}$	$\{A_w\}$

Before processing the third download point, we apply our dominance rule (at the beginning of step 2.2). Since A_w and B_w have already been processed, we have $\Gamma^+_{h_1} = \Gamma^+_{h_2} = \Gamma^+_{h_3}$. We can see that h_3 dominates h_1 and h_2 due respectively to equations (8b) and (8a).

	b_h	l_h	$\mathcal{W}_h$	Γ^+_h
h_3	960	0	$\{B_w\}$	$\varnothing$

If we continue the process (considering download points C_w, D_w, E_w, F_w and G_w), we obtain the following set of labels before step 3:

	b_h	l_h	$\mathcal{W}_h$	Γ^+_h
h_7	1000	200	$\{B_w, C_w, D_w, F_w\}$	$\varnothing$
h_8	940	200	$\{B_w, C_w, D_w, F_w, G_w\}$	$\varnothing$

3.3.3 | Step 3

Labels are updated at the end of the algorithm to remove data in the buffer (at the end of the time horizon, the buffer must be empty, i.e., $b_h = 0,\ \forall h$):

	b_h	l_h	$\mathcal{W}_h$	Γ_h^+
h_7	0	1200	$\{B_w, C_w, D_w, F_w\}$	$\varnothing$
h_8	0	1140	$\{B_w, C_w, D_w, F_w, G_w\}$	$\varnothing$

And finally, h_8 dominates h_7, thus the optimal solution is:

$$Q^* = \{B_w, C_w, D_w, F_w, G_w\}$$

3.4 | Complexity of the $DP^+_{pdt_{\mathcal{L}}}$ algorithm for special classes of instances

Proposition 4 *The $DP^+_{pdt_{\mathcal{L}}}$ algorithm is a fixed-parameter tractable (FPT) algorithm, for instances of the intra class, considering the maximum number of download points inside a slot as a parameter of the problem.*

Proof At the beginning of the algorithm, the set $\mathcal{H}$ contains a single label. We will show that, for instances of the *intra* class, if there is a single label in $\mathcal{H}$ at the beginning of the processing of a slot s, then a single label will remain in $\mathcal{H}$ at the end of the processing of s (due to the dominance rule).

Assume there is a single label $h = (b_h, l_h, \Gamma_h, \mathcal{W}_h)$ in $\mathcal{H}$ at the beginning of the processing of a slot s: first, b_h and l_h are updated (no new labels are created), then the download points inside s are processed and new labels are created. All these labels have the same updated value of l_h (see algorithm 2).

At the end of the processing of the last download point inside s, the pruning happens. Since there cannot be conflict between download points not in the same slot, all labels will have the same reduced set of conflicts $\Gamma^+ = \varnothing$. Since all labels have the same amount of losses l_h, a single label in $\mathcal{H}$ dominates all the others, thus this label will be the single one remaining in $\mathcal{H}$ at the end of the processing of s. For any slot s, the processing of the download points inside Q^s is done in $O(2^{|Q^s|})$ time, thus the complexity of the $DP^+_{pdt_{\mathcal{L}}}$ algorithm for instances of the *intra* class is:

$$O(|\mathcal{S}|.2^{\alpha_{max}})$$

where:

$$\alpha_{max} = \max_{s \in \mathcal{S}} |Q^s|$$

∎

4 | COMPUTATIONAL RESULTS

In this section, we present some computational results obtained by solving randomly generated instances for the $\texttt{MaxPDT}_{\mathcal{L}}$ problem and realistic instances for the MaxPDT problem.

4.1 | Computation results for the $\texttt{MaxPDT}_{\mathcal{L}}$ problem

4.1.1 | Instances

We generated random instances for the $\texttt{MaxPDT}_{\mathcal{L}}$ problem, grouped into 3 categories depending on their conflicts:

- *Int*: Conflicts can only occur within a slot.
- *Adj*: Conflicts can only occur between successive download points.
- *All*: Conflicts are not constrained.

For each category, instances were generated using a given number of slots, a random number (within a given range) of download points per slot, a fixed buffer size, a probability of conflict between download points, and randomly generated acquisition and download volumes. Instances of type *All* are much harder to solve than instances of type *Int* or *Adj* (see Table 4), which is why the parameters used to generate these instances are different from the parameters used for the *Int* and *Adj* types:

- for instances of types *Int* and *Adj*, the following parameters were used:
  - number of slots per instance: 100, 500, 1000, 2000, 3000, 4000, 5000
  - number of download points per slot (minimum / maximum): 0/5, 5/10, 10/20, 20/40
- for instances of type *All*, the following parameters were used:
  - number of slots per instance: 10, 50, 100, 200, 300, …, 900, 1000
  - number of download points per slot (minimum / maximum): 0/5, 5/10, 10/20

For each combination of parameters, four probabilities of conflict were used: 0.2, 0.4, 0.6, 0.8, and four different buffer sizes (see below).

Download volumes of download points were uniformly distributed between 100 and 200 gigabits for all instances. The acquisition volumes were uniformly distributed between $150 * M$ and $250 * M$ where M is the maximum possible number of download points per slot. The buffer sizes used were multiples of the average acquisition volumes $\hat{a} = 200 * M$: $B = \hat{a}, 2\hat{a}, 4\hat{a}, 8\hat{a}$.

These parameters have been chosen to test the performance of $DP^{+}_{pdt_{\mathcal{L}}}$ compared to the $MILP$ model and are unlikely to appear in real instances of the MaxPDT problem. Although the number of slots is somehow realistic (slots are typically one hour long, so 5000 slots represent a bit more than 6 months), it is unlikely to encounter that many download points per slot or that much conflicts in real applications.

For each combination of these parameters (category, number of slots, number of download points per slot, probability of conflict and buffer size), 4 instances were generated, for a total of 5888 instances.

4.1.2 | Computational context

The $DP^{+}_{pdt_{\mathcal{L}}}$ and the mathematical model for the $\texttt{MaxPDT}_{\mathcal{L}}$ problem were both implemented in C++. The $MILP$ model was solved using *CPLEX 12.7*. All experiments were run using a single thread on a cluster with 2.3 GHz processor and 6 gigabytes of RAM per thread. A time limit of one hour was used.

4.1.3 | Results

Table 4 shows, for both methods, the number of instances where the method found the optimal solution, or a feasible solution, or failed to provide a solution. Both methods have no problem with instances of the *Adj* category. The $DP^{+}_{pdt_{\mathcal{L}}}$ algorithm manages to solve all instances of the *Int* category, even those with up to 40 download points per slot, while the $MILP$ fails to solve some of these. For instances of the *All* category, while the $MILP$ provide solutions (even if non-optimal) for about 80% of the instances, $DP^{+}_{pdt_{\mathcal{L}}}$ only manages to provide solutions for about 18% of the instances, and even if the number of instances solved optimally is slightly larger for $DP^{+}_{pdt_{\mathcal{L}}}$, some instances solved to optimality by the $MILP$ are not solved up to optimality by $DP^{+}_{pdt_{\mathcal{L}}}$. Since it is trivial to find a feasible solution for any kind of instances (select no stations), instances for which the methods did not manage to provide solutions are instances for which the system ran out of memory and killed the program.

In the following, we look in detail at the performance of both methods when solving instances of the *Adj* and *Int* categories, and give some information regarding the performance of the methods for the *All* category.

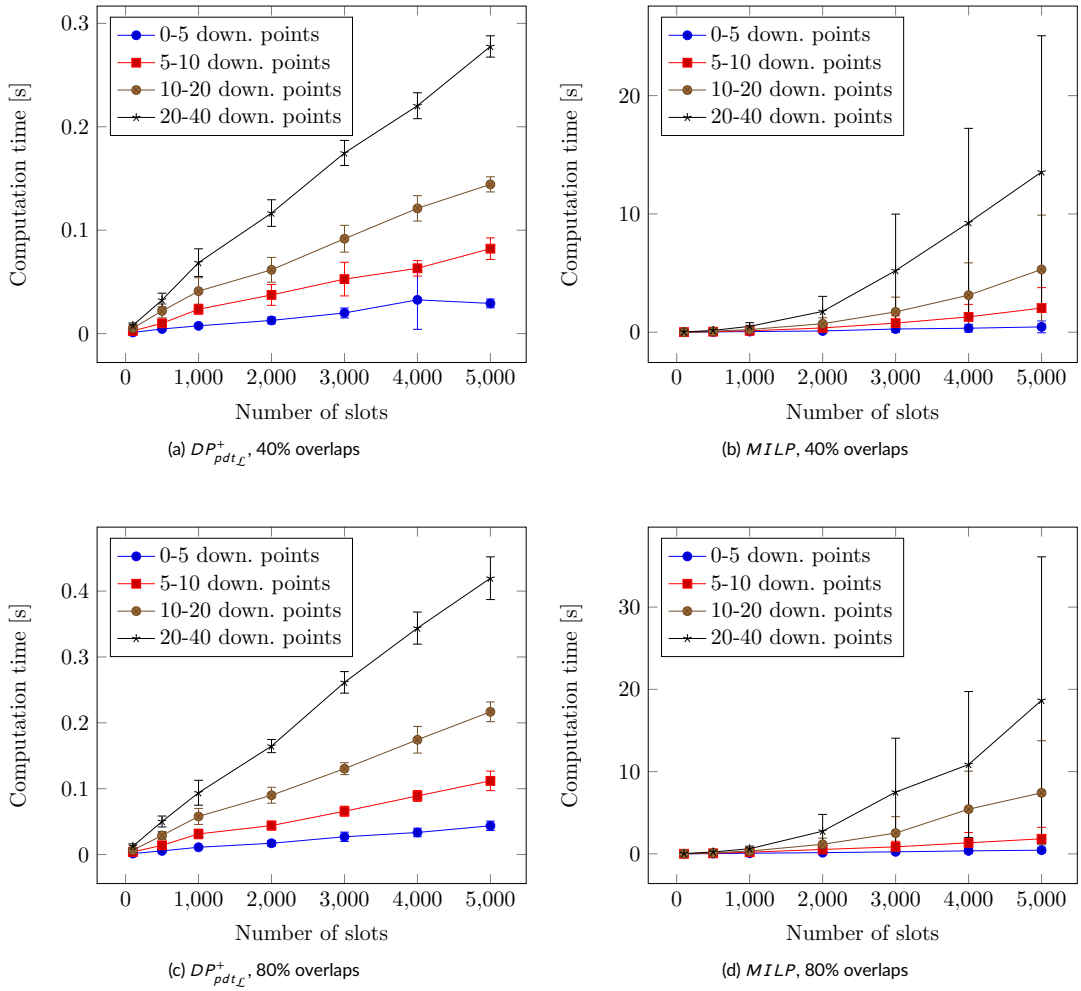
Solver	Category	# Instances	# Optimal	# Non optimal	# Not found
$DP^{+}_{pdt_{\mathcal{L}}}$	*Adj*	1792	1792	0	0
$DP^{+}_{pdt_{\mathcal{L}}}$	*All*	2304	369	48	1887
$DP^{+}_{pdt_{\mathcal{L}}}$	*Int*	1792	1792	0	0
$MILP$	*Adj*	1792	1792	0	0
$MILP$	*All*	2304	336	1496	472
$MILP$	*Int*	1792	1587	165	40

TABLE 4 Comparison of the number of instances solved by $DP^{+}_{pdt_{\mathcal{L}}}$ and the $MILP$, for each category of instances. The third column indicates the number of instances tested per category, and the three right-most columns indicate respectively the number of instances for which no solutions were found (time or memory limit reached), a feasible solution was found, or an optimal solution was found.

Figures 2 and 3 show computation times of CPLEX and $DP^{+}_{pdt_{\mathcal{L}}}$ for instances of the *Adj* and *Int* classes. Each curve shows data averaged over the four instances of each group (same set of parameters) and over the various buffer sizes used, since we noticed that the buffer size had little influence on the computational time of the algorithms. Instances of the *Int* class for which the $MILP$ did not find a solution (or did not find the optimal one) are included in these graphs with a computation time of 3600 seconds (the time limit used) so that each pair of plots correspond to the same set of instances. We can see that for these two classes of instances, $DP^{+}_{pdt_{\mathcal{L}}}$ is orders of magnitude faster than the $MILP$ solver.

For instances of the *Adj* category, Figures 2(a)–2(d) show a near-linear behavior for $DP^{+}_{pdt_{\mathcal{L}}}$ regarding the number of slots and the number of download points per slot, but a slightly exponential behavior for the $MILP$. Furthermore, $DP^{+}_{pdt_{\mathcal{L}}}$ seems to be less impacted by minor changes in the input instances than the $MILP$ model. For both methods, the number of conflicts (percentage of overlaps) has little impact on the computation times, which is why results are only provided for 40% and 80% probability of conflicts.

For instances of the *Int* category, we can see on Figures 3(a), 3(c) and 3(e), that the computation time of $DP^{+}_{pdt_{\mathcal{L}}}$



(a) $DP^{+}_{pdt_{\mathcal{L}}}$, 40% overlaps

(b) $MILP$, 40% overlaps

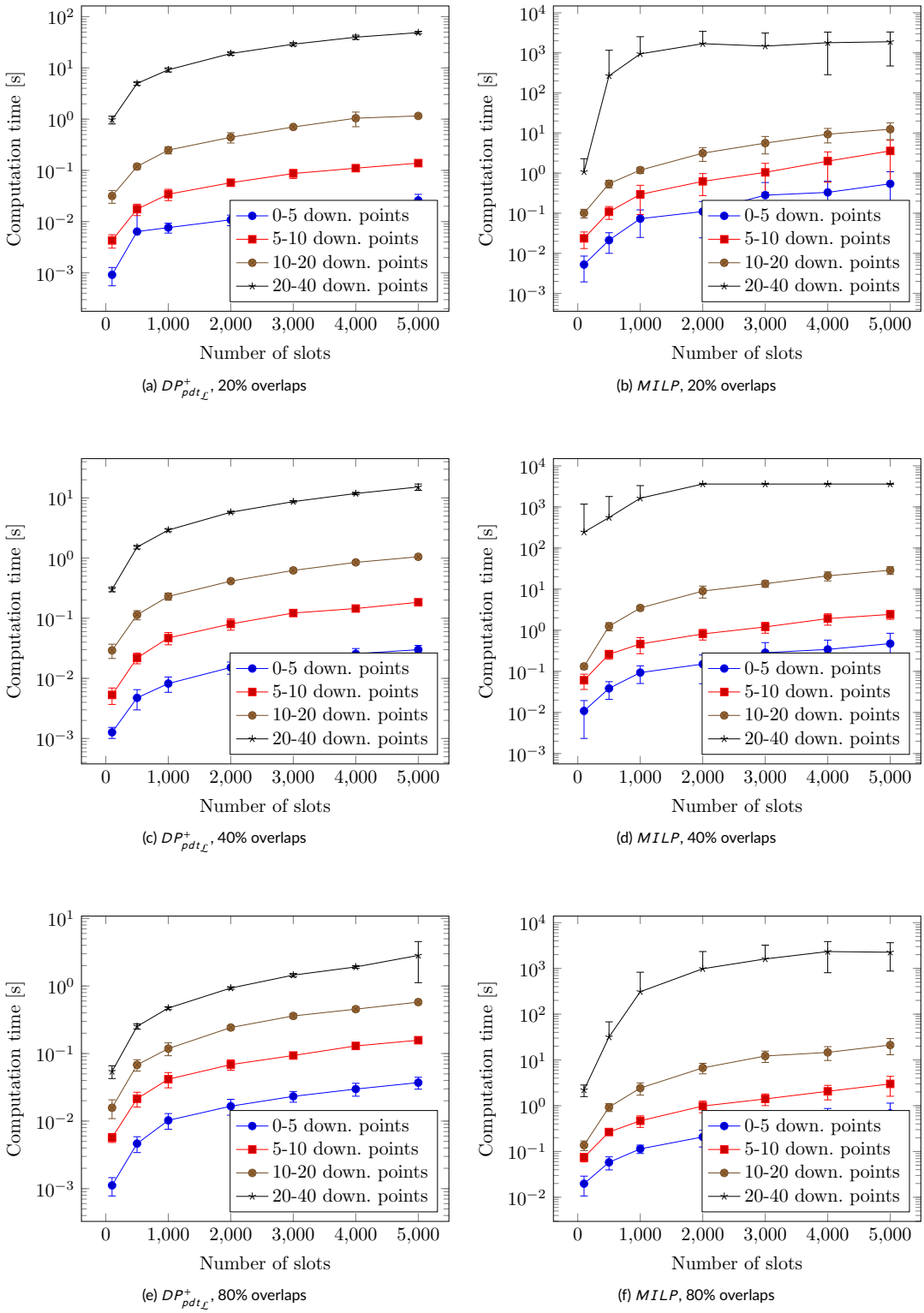
(c) $DP^{+}_{pdt_{\mathcal{L}}}$, 80% overlaps

(d) $MILP$, 80% overlaps

FIGURE 2 Computation time for instances of the *Adj* category using $DP^{+}_{pdt_{\mathcal{L}}}$ or the $MILP$.

seems to stabilize as the number of slots becomes larger. This behavior is explained by the *FPT* behavior of $DP^{+}_{pdt_{\mathcal{L}}}$ on instances of the *Int* category — the fixed parameter is the maximum number of download points per slot, which is constant for each curve in these figures. These plots also show that $DP^{+}_{pdt_{\mathcal{L}}}$ is more efficient at solving instances with a large number of overlaps. It can easily be explained by the fact that if a label is in conflict with a download point, the label is not duplicated. Thus, the more conflicts there are, the less label duplications occur. Figures 3(b), 3(d) and 3(f) show a still visible but weaker behavior regarding the number of conflicts for the $MILP$ solver.

Table 5 shows the number of instances from the *All* category solved (to optimality or not). Similar to instances of the *Adj* and *Int* categories, this shows that $DP^{+}_{pdt_{\mathcal{L}}}$ is able to solve more instances when the percentage of conflicts is high. The $MILP$ seems to have issue solving instances with a large number of conflicts — while the number of instances solved to optimality is slightly higher for instances with a large number of conflicts, the number of instances without a solution is also higher.



(a) $DP^{+}_{pdt_{\mathcal{L}}}$, 20% overlaps

(b) $MILP$, 20% overlaps

(c) $DP^{+}_{pdt_{\mathcal{L}}}$, 40% overlaps

(d) $MILP$, 40% overlaps

(e) $DP^{+}_{pdt_{\mathcal{L}}}$, 80% overlaps

(f) $MILP$, 80% overlaps

FIGURE 3 Computation time for instances of the *Int* category using $DP^{+}_{pdt_{\mathcal{L}}}$ or the $MILP$.

	$DP^{+}_{pdt_{\mathcal{L}}}$			$MILP$		
Conflict	Optimal	Non optimal	Not found	Optimal	Non optimal	Not found
20%	32	0	544	64	459	53
40%	65	0	511	80	395	101
60%	96	1	479	86	346	144
80%	176	47	353	106	296	174

TABLE 5 Comparison of the number of instances of the *All* category solved by $DP^{+}_{pdt_{\mathcal{L}}}$ and the $MILP$ for different probabilities of conflicts. For each probability, 576 instances were tested, and the columns respectively represent the number of instances for which no solutions were found (time or memory limit reached), a feasible solution was found, or an optimal solution was found.

These experiments show that $DP^{+}_{pdt_{\mathcal{L}}}$ is much faster at solving instances of $\texttt{MaxPDT}_{\mathcal{L}}$ than the $MILP$ model, even if both methods fail to solve complex instances of $\texttt{MaxPDT}_{\mathcal{L}}$ where there are a large number of conflicts and slots, and where conflicts are not confined.

4.2 | Computation results for the `MaxPDT` problem

4.2.1 | Instances

We generated custom instances for a low-earth orbiting satellite using concepts of operations from [9, 7, 10]: data rate (before cloud impact) $\mathcal{D}_{\mathcal{R}} = 10.5$ Gbps, buffer size $B = 2300$ gigabits, an acquisition (slot) every hour and a constant acquisition volume of $a^s = 500$ gigabits. Visibility windows were computed using the *Systems Tool Kit (AGI, STK)* software, for a sun-synchronous low-earth orbiting satellite with an altitude of 700 kilometers and a local time of 10:30 A.M, between 1990 and 2010. We used the *ERA Interim* cloud database [2] (freely available) to approximate the cloud cover $c_v \in [0, 1]$ during any visibility window v, using cloud data from the previous decades (between 1990 and 2010). Based on these approximated cloud covers, we assumed that the data rate function d_v of a visibility window v was constant during the visibility window but proportional to the cloud cover c_v over the station τ_v at the beginning of the visibility window:

$$d_v(t) = \mathcal{D}_{\mathcal{R}} * (1 - c_v), \quad \forall t \in \left[t_v^{sta},\ t_v^{end}\right]$$

The download volume ρ_w of a download point w associated to a visibility window v can thus be computed as:

$$\rho_w = \int_{t_v^{sta}}^{t_v^{end}} \mathcal{D}_{\mathcal{R}}(1 - c_v)dt = \mathcal{D}_{\mathcal{R}}(1 - c_v)(t_v^{end} - t_v^{sta})$$

We discarded download points that were too small ($\rho_w < 1$ gigabits). We used two different networks composed of 11 ($\mathcal{N}_{11}$) and 16 ($\mathcal{N}_{16}$) possible locations and since we could not find realistic information for the costs of the different locations, we choose to simply select fixed numbers K of stations (between 1 and 16). We generated instances of various durations of 1, 2, 4, 5, 10 and 20 years. All instances were generated using the same 20 year horizon, meaning that there are respectively 20, 10, 5, 4, 2, 1 instances of each duration.

4.2.2 | Computational context

The $MILP$ model and both the $\mathcal{EE}_{pdt}(\mathcal{A})$ and the $\mathcal{BB}_{pdt}(\mathcal{A})$ algorithms were implemented in C++. The $MILP$ solver used was *CPLEX 12.7*. All experiments were run using 1 and 8 threads on a cluster with 2.3 GHz processor and 3.5 gigabytes of RAM per thread. For both networks, we looked for networks of $K = 1, \ldots, N$ stations (with $N = 11$ or 16 depending on the network), for a total of 1134 instances that were run using the 3 algorithms with both 1 and 8 threads (6804 runs).

4.2.3 | Results

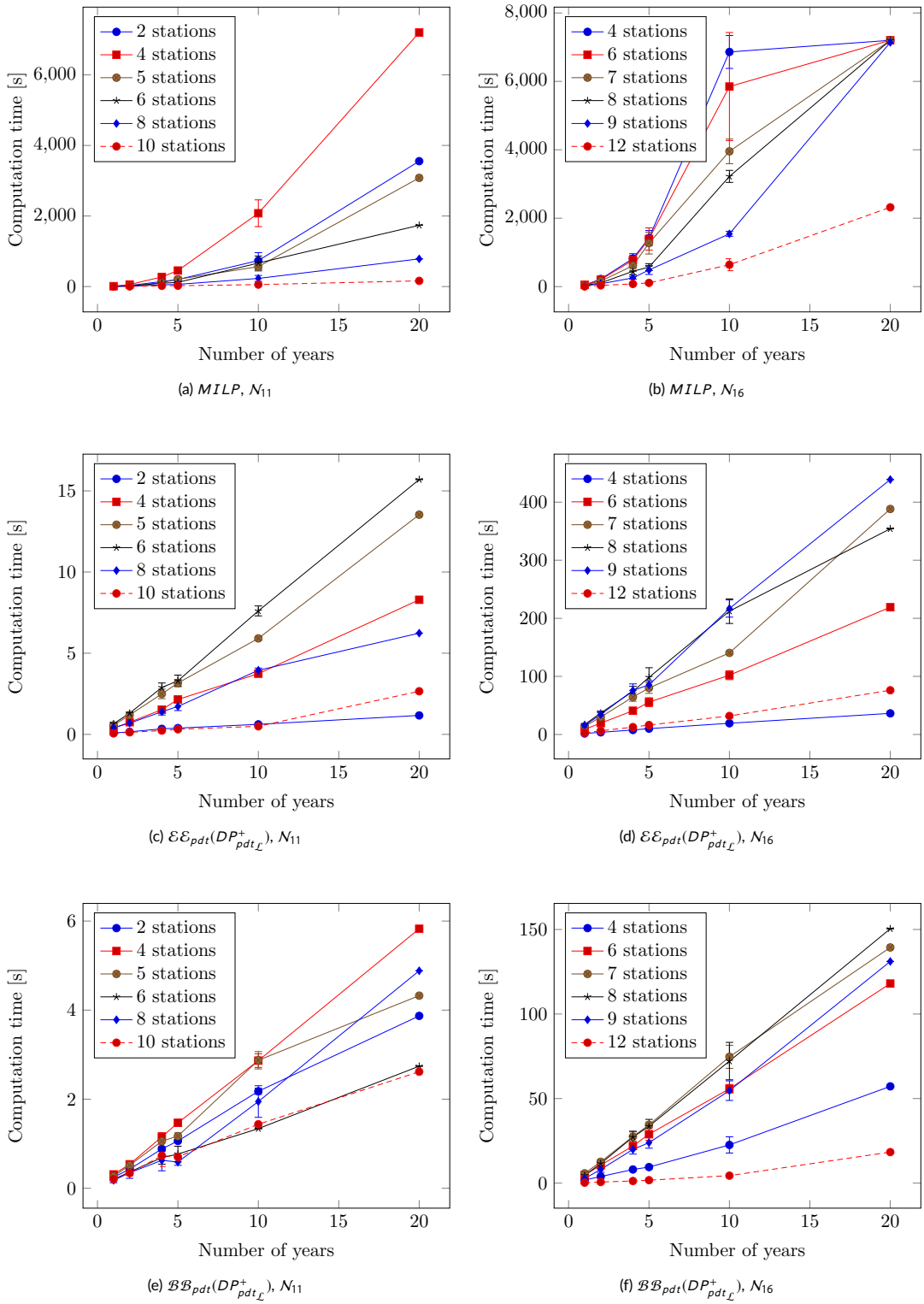
Figure 4 shows the computation times (means and standard deviations) for the $MILP$ solver and both $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ and $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ regarding the temporal length of the instances for various choices of K (number of chosen stations). Figures 4(a)–4(b) show that the $MILP$ solver either does not succeed to provide a solution, or does not manage to prove optimality (the time limit of 7200 seconds is reached) for some instances with a large horizon (20 years for $\mathcal{N}_{11}$ and 10 or 20 years for $\mathcal{N}_{16}$). These figures also show an exponential behavior regarding both the length and the number of stations chosen in the instances for the $MILP$ model. Figures 4(c)–4(f) show that both our enumeration methods have a near-linear behavior regarding the length of the instances. This can be explained by the FPT behavior of the $DP^{+}_{pdt_{\mathcal{L}}}$ on instances with no conflict between slots — in real instances, there are few conflicts between slots and the number of download points per slot does not depend on the length of the instance. Figure 4 shows that our $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ combination is faster than our $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ combination for the generated instances, and that both combinations outperform the $MILP$ solver. Moreover, the standard deviations of the computation time for our hierarchical approaches are lower than the standard deviations for the $MILP$ solver, meaning that our methods are not really impacted by slight differences in the input instances.

Figure 5 shows the computation times (averages and standard deviations) for the $MILP$ solver and both algorithms regarding the number of stations chosen for various temporal lengths of the instances (1 and 2 years were very similar, like 4 and 5 years). These plots present similar information as Figure 4 but in a different way. Figures 5(c)–5(d) show an exponential behavior for the exhaustive enumeration $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$. As expected, instances where the number of chosen stations is close to half the number of available stations are the hardest to solve. While Figure 5(f) depicts the same exponential behavior for the $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ algorithm, Figure 5(e) shows a more chaotic behavior for $\mathcal{N}_{11}$. This behavior can be explained by the low solution times — such small solution times can depict strange behavior with a multi-threaded algorithm. Figure 6 shows that the exponential behavior of the $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ algorithm using one thread is similar to the behavior of the $MILP$ solver using 8 threads, enforcing this hypothesis.

Figures 5(a)–5(b) also show an exponential behavior for the $MILP$ solver but slightly shifted to the left — the hardest instances seem to be those where the number of chosen stations is slightly less than half the number of available locations.

Table 6 shows the computation time difference when using either 1 or 8 threads for the $MILP$ solver and both algorithms, averaged for instances with a temporal length of 5 years.

For complex instances (when K is close to half the number of available stations), the $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ clearly takes advantage of the 8 threads, while the $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ only has a ratio close to 8 for the hardest instances. When the instances are simpler to solve, the $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ fails to take advantage of the threads compared to the $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$.



(a) $MILP$, $\mathcal{N}_{11}$

(b) $MILP$, $\mathcal{N}_{16}$

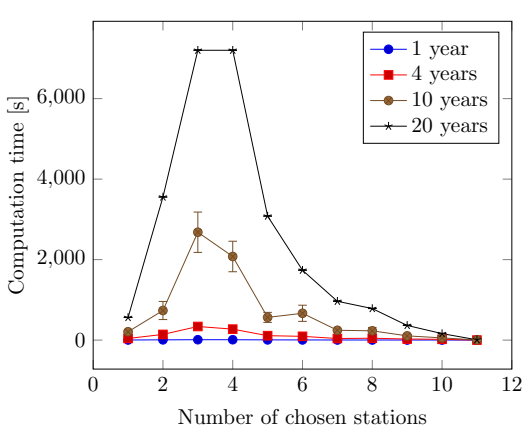
(c) $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{11}$

(d) $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{16}$

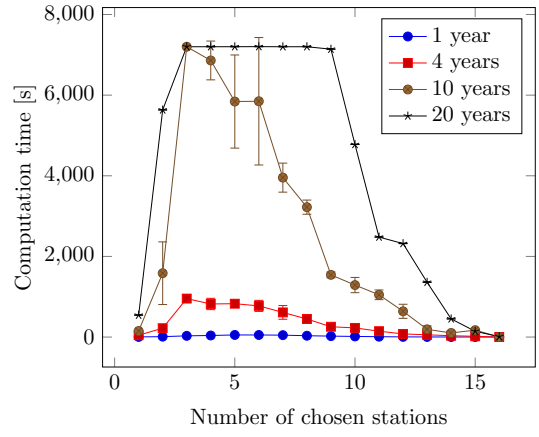
(e) $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{11}$

(f) $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{16}$

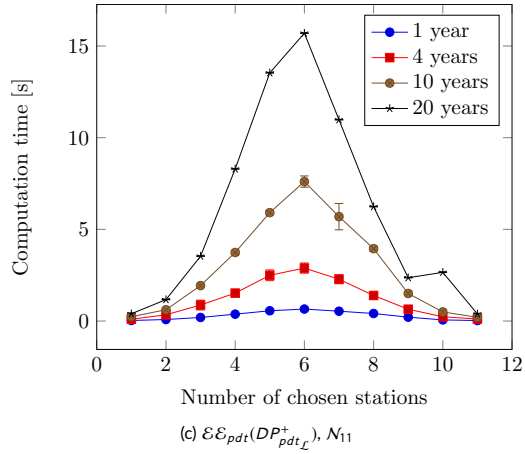
FIGURE 4 Computation times for each methods using 8 threads, depending on the temporal horizon length.



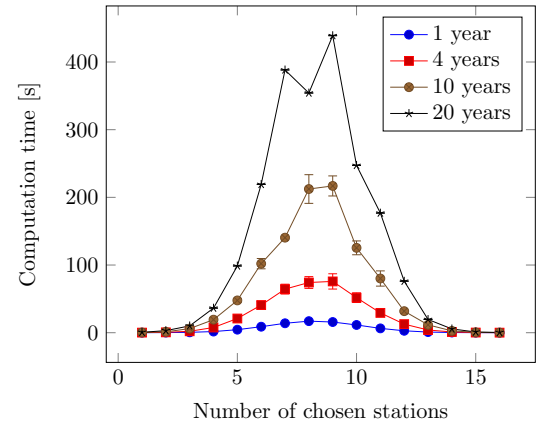
(a) $MILP$, $\mathcal{N}_{11}$



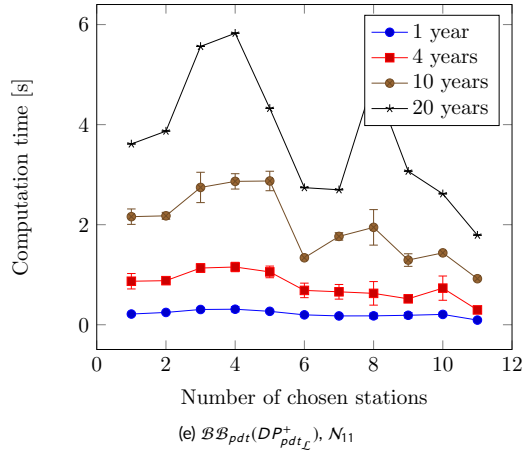
(b) $MILP$, $\mathcal{N}_{16}$



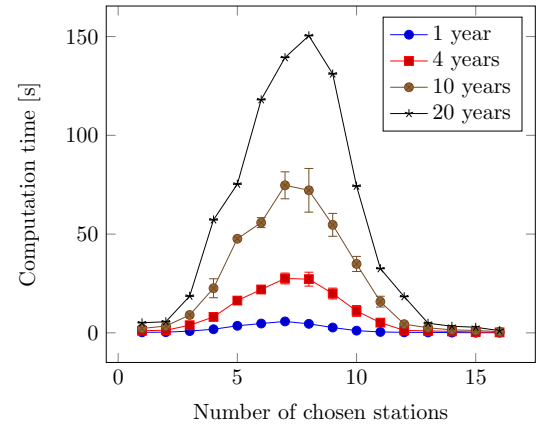
(c) $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{11}$



(d) $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{16}$



(e) $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{11}$



(f) $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{N}_{16}$

FIGURE 5 Computation times for each methods using 8 threads, depending on the number of chosen stations.

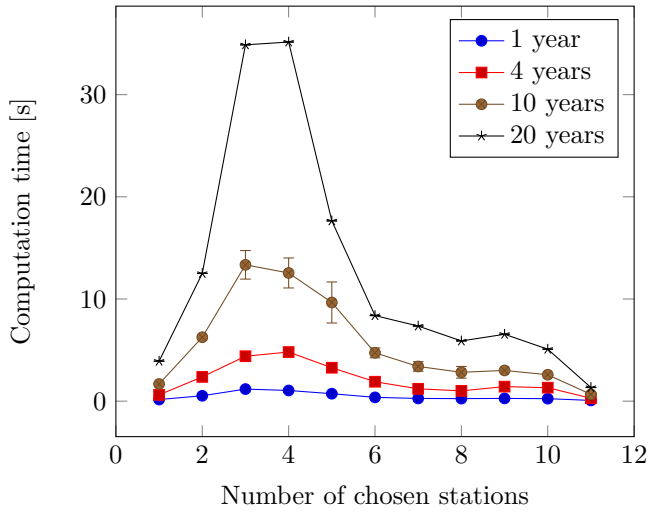


FIGURE 6 Computation time for the $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ algorithm using 1 thread for $\mathcal{N}_{11}$.

This behavior is easily explained by the complexity of multi-threading a branch-and-bound compared to an exhaustive enumeration — in both enumerations, each run of $DP^{+}_{pdt_{\mathcal{L}}}$ is done on a separate thread, so 8 instances of $\texttt{MaxPDT}_{\mathcal{L}}$ can be solved simultaneously, but the synchronization process is more complex for $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ than for $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$. While the $MILP$ solver is also faster when using 8 threads, the ratio is not close to the expected value of 8, meaning the relative gap between the $MILP$ solver and our enumerations is greater when using 8 threads.

These results show that solving realistic instances of `MaxPDT` by combining an enumeration method and $DP^{+}_{pdt_{\mathcal{L}}}$ is much faster than using the $MILP$ model. Experiments also show that the branch-and-bound enumeration is often faster than the complete enumeration, even if it takes less advantage of multi-threading.

		$\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$			$\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$			$MILP$		
	K	1 th.	8 th.	ratio	1 th.	8 th.	ratio	1 th.	8 th.	ratio
$\mathcal{N}_{11}$	2	3.16	1.06	2.98	1.49	0.39	3.85	2135.00	200.56	10.65
$\mathcal{N}_{11}$	4	6.11	1.47	4.16	14.64	2.15	6.81	1871.52	453.79	4.12
$\mathcal{N}_{11}$	5	4.19	1.17	3.58	24.78	3.16	7.84	932.53	206.17	4.52
$\mathcal{N}_{11}$	6	2.03	0.77	2.65	28.71	3.30	8.70	291.33	118.65	2.46
$\mathcal{N}_{11}$	8	1.23	0.59	2.08	10.26	1.72	5.96	64.77	62.29	1.04
$\mathcal{N}_{11}$	10	1.16	0.70	1.65	0.89	0.31	2.84	28.12	23.66	1.19
$\mathcal{N}_{16}$	4	64.51	9.48	6.81	62.17	10.04	6.19	5593.90	1421.25	3.94
$\mathcal{N}_{16}$	6	187.33	28.92	6.48	373.87	55.66	6.72	4802.77	1392.55	3.45
$\mathcal{N}_{16}$	7	274.81	34.46	7.98	709.01	80.08	8.85	3521.69	1271.03	2.77
$\mathcal{N}_{16}$	8	226.76	33.60	6.75	636.93	98.31	6.48	2157.92	568.72	3.79
$\mathcal{N}_{16}$	9	144.35	23.92	6.03	843.58	85.47	9.87	1491.61	486.14	3.07
$\mathcal{N}_{16}$	12	7.72	1.73	4.47	108.35	16.25	6.67	328.47	108.56	3.03

TABLE 6 Comparison of solution time between $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ and the MILP for instances with a temporal horizon of 5 years, on $\mathcal{N}_{11}$ and $\mathcal{N}_{16}$, for various number of selected stations K. For each method, the first two columns represent the averaged solving time (in seconds) when using respectively 1 or 8 threads, while the third one shows the improvement ratio when using 8 threads compared to 1 thread.

5 | CONCLUSIONS

In this paper, we consider the problem MaxPDT of finding a network of ground stations maximizing the amount of data transferred from a low-earth satellite satellite to the Earth using optical communications. We model this problem using an aggregation of visibility windows into download point which allows any model to be used for the computation of the data rate during communications. We analyze the $\texttt{MaxPDT}_{\mathcal{L}}$ problem within the MaxPDT problem and give complexity results regarding its general form. We propose mixed-integer linear programs for both MaxPDT and $\texttt{MaxPDT}_{\mathcal{L}}$. To solve $\texttt{MaxPDT}_{\mathcal{L}}$, we propose a dynamic programming algorithm $DP^{+}_{pdt_{\mathcal{L}}}$ that exhibits an FPT behavior for realistic instances. We combine this algorithm with two enumeration methods, $\mathcal{EE}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$ and $\mathcal{BB}_{pdt}(DP^{+}_{pdt_{\mathcal{L}}})$, in order to solve the MaxPDT. We showed that for realistic instances, both enumerations methods outperform the $MILP$ solver.

Additional research efforts are needed for investigating the complexity of the $\texttt{MaxPDT}_{\mathcal{L}}$ and MaxPDT problems for real instances where conflicts within visibility windows can be represented as an intersection graph with possible overlaps between slots. A more sophisticated way of taking overlaps between visibility windows into account could also be considered. While we consider that two overlapping visibility windows are in total conflict, it could be interesting to split these into sub-windows to allow partial usage of both conflicting visibility windows. Finally, a version of the MaxPDT problem on a rolling horizon could be considered, such as the minimization of the maximum amount of data loss over each month.

references

[1] E. Chen, H. Mei, C. Zhang, and C. Chang, *The link availability analysis of GEO satellite-to-ground laser communication*, Proceedings SPIE 9619, 2015 International Conference on Optical Instruments and Technology: Optoelectronic Devices and Optical Signal Processing, August 2015.

[2] D.P. Dee, S.M. Uppala, A.J. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M.A. Balmaseda, G. Balsamo, P. Bauer, P. Bechtold, A.C.M. Beljaars, L. van de Berg, J. Bidlot, N. Bormann, C. Delsol, R. Dragani, M. Fuentes, A.J. Geer, L. Haimberger, S.B. Healy, H. Hersbach, E.V. Hólm, L. Isaksen, P. Kållberg, M. Köhler, M. Matricardi, A.P. McNally, B.M. Monge-Sanz, J.J. Morcrette, B.K. Park, C. Peubey, P. de Rosnay, C. Tavolato, J.N. Thépaut, and F. Vitart, *The ERA-Interim reanalysis: Configuration and performance of the data assimilation system*, Q. J. Royal Meteorological Soc. **137** (2011), 553–597.

[3] B. Eilertsten and P. Hyvönen, *Ground station networks vs. geo relay satellites for polar orbiting satellites data download*, Proceedings of the 12th International Conference on Space Operations (SpaceOps), 2012.

[4] C. Fuchs and F. Moll, *Ground station network optimization for space-to-ground optical communication links*, IEEE/OSA J. Optical Commun. Networking **7** (December 2015), 1148–1159.

[5] D. Giggenbach, E. Lutz, J. Poliak, R. Maa-Calvo, and C. Fuchs, *A high-throughput satellite system for serving whole Europe with fast internet service, employing optical feeder links*, Proceedings of Broadband Coverage in Germany, 9th ITG Symposium, April 2015.

[6] A. Guérin, F. Lacoste, A. Laurens, J. Berthon, C. Periard, and D. Grimal, *Optimisation of an optical ground station network*, Proceedings of the 5th ESA International Workshop on Tracking, Telemetry and Command Systems for Space Applications (TTC, ESA-ESTEC), 2010.

[7] A. Guérin, G. Lesthievent, and J.L. Issler, *Evaluation of new technological concepts for high data rate payload telemetry*, Proceedings of the 5th ESA International Workshop on Tracking, Telemetry and Command Systems for Space Applications (TTC, ESA-ESTEC), 2010.

[8] J.Y. Hsiao, C.Y. Tang, and R.S. Chang, *An efficient algorithm for finding a maximum weight 2-independent set on interval graphs*, Informat. Process. Lett. **43** (Oct. 1992), 229–235.

[9] IOAG, *Optical link study group final report*, Tech. report IOAG.T.OLSG.2012.V1, Interagency Operations Advisory Group (IOAG), Optical Link Study Group (OLSG), June 2012.

[10] F. Lacoste, A. Guérin, A. Laurens, G. Azema, C. Periard, and D. Grimal, *FSO ground network optimization and analysis considering the influence of clouds*, Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP), 2011.

[11] A. Le Kernec, M. Sotom, B. Bénazet, J.B. Gonzalez, L.P. Quesada, M. Maignan, I. Esquivias, F. Lopez, and N. Karafolas, *Space evaluation of optical modulators for microwave photonic on-board applications*, Proceedings of the 8th International Conference on Space Optics (ICSO), October 2010.

[12] R. Link, M.E. Craddock, and R.J. Alliss, *Mitigating the impact of clouds on optical communications*, Proceedings of the 2005 IEEE Aerospace Conference, March 2005.

[13] H. Porte, A. Le Kernec, L.P. Quesada, I. Esquivias, H. Brahimi, J.B. Gonzalez, A. Mottet, and M. Sotom, *Optimization and evaluation in space conditions of multi-GHz optical modulators*, Proceedings of the 10th International Conference on Space Optics (ICSO), October 2014.

[14] S. Poulenard, M. Crosnier, and A. Rissons, *Ground segment design for broadband geostationary satellite with optical feeder link*, IEEE/OSA J. Optical Commun. Networking **7** (April 2015), 325–336.

[15] S. Poulenard, M. Ruellan, B. Roy, J. Riedi, F. Parol, and A. Rissons, *High altitude clouds impacts on the design of optical feeder link and optical ground station network for future broadband satellite services*, Procceedings SPIE 8971, Free-Space Laser Communications and Atmospheric Propagation XXVI, March 2014.

[16] B. Roy, S. Poulenard, S. Dimitrov, R. Barrios, D. Giggenbach, A.L. Kernec, and M. Sotom, *Optical feeder links for high throughput satellites*, Proceedings of the IEEE International Conference on Space Optical Systems and Applications (ICSOS), October 2015, pp. 1–6.

[17] K.J. Schulz and J. Rush, *Results of the optical link study group*, Proceedings of the 12th International Conference on Space Operations (SpaceOps), 2012.

[18] Z. Sodnik, J.P. Armengola, R.H. Czichyb, and R. Meyerc, *Adaptive optics and ESA's optical ground station*, Proceedings SPIE 7464, Free-Space Laser Communications IX, August 2009.

[19] Z. Sodnik and M. Sans, *Extending EDRS to laser communication from space to ground*, Proceedings of the International Conference on Space Optical Systems and Applications (ICSOS), October 2012.

[20] Y. Takayama, M. Toyoshima, and N. Kura, *Estimation of accessible probability in a low-earth orbit satellite to ground laser communications*, Radioengineering **19** (June 2010), 249–253.

[21] G.S. Wojcik, H.L. Szymczak, R.J. Alliss, and R.P. Link, *Deep-space to ground laser communications in a cloudy world*, Proceedings SPIE 5892, Free-Space Laser Communications V, August 2005.