Scheduling Satellite Timetables using DCOP

Shai Krigman, Tal Grinshpoun[0000−0002−4106−3169], and Lihi Dery[0000−0002−8710−3349]

Department of Industrial Engineering and Management
Ariel Cyber Innovation Center
Ariel University, Ariel 4070000, Israel
shai.krigman@msmail.ariel.ac.il, talgr@ariel.ac.il, lihid@ariel.ac.il

Abstract. Earth observation satellites (EOS) are satellites equipped with optical sensors that orbit the Earth to take photographs of specific areas at the request of users. With the development of space technology, the number of satellites increases continuously. Yet still, the number of satellites cannot meet the explosive growth of applications. Thus, scheduling solutions are required to satisfy requests and obtain a high observation efficiency. While the literature on multi-satellite scheduling is rich, most of the solutions are centralized algorithms. However, due to their cost, EOS systems are often co-funded by several agents (e.g., countries, companies, or research institutes) and central solutions require that these agents will share their requests for observations with others. To date, there is no solution for EOS scheduling that protects the private information of the interested parties. In this study, we model the EOS scheduling problem as a distributed constraint optimization problem (DCOP). This modeling enables generating timetables for the satellites in a distributed manner without a priori sharing private information of the users with some central authority. For solving the resulting DCOP, we use the Distributed Stochastic Algorithm (DSA), which is a simple DCOP algorithm that is known to produce efficient solutions in a timely manner. The modeling together with the solving of the resulting DCOP constitute our new solution method, which we term Distributed Satellite Timetable Solver (DSTS). Experimental evaluation reveals that the DSTS method provides solutions of higher quality than a commonly-used Greedy algorithm.

Keywords: Earth observation satellites · Satellite timetables · DCOP.

1 Introduction

Earth observation satellites (EOSs) are sensor-equipped satellites that are designated to take photographs of special areas at the request of a user [39]. The satellites perform a cycle of orbits around the Earth over a period of several days. Each orbit slightly changes with respect to the preceding one but its trajectory is cyclic in the sense that the satellite recovers its initial position after a predefined number of orbits. Furthermore, a full cycle enables the satellite to view each

area of the planet. Most EOSs operate at low altitudes with the orbital periods varying from dozens of minutes to several hours. However, it takes several days for a single EOS to complete a full cycle and view the whole area of the Earth. During the course of one particular orbit, a satellite can take several photographs by rotating itself between consecutive shots. After capturing the photographs, the acquired data is stored in the on-board memory and transferred to a ground station when the satellites are in a feasible transferring range [49].

EOSs have been extensively employed in a wide range of tasks, such as Earth resource exploration, natural-disaster surveillance, environmental monitoring, and defense missions [6]. The demand for EOS services has risen over time due to some unique advantages, including an expansive coverage area, long-term surveillance, accurate and effective information access, and unlimited airspace borders [43].

With the development of space technology, the number of satellites continuously increases [43]. Yet, the number of EOSs still cannot meet the demand due to an explosive growth of applications that require observations. Consequently, scheduling solutions are needed to satisfy more requests and obtain a high observation efficiency. In particular, multi-satellite systems require dedicated scheduling solutions [3].

Due to their cost, EOS systems are often co-funded by several agents (e.g., countries, companies, or research institutes). Once constructed and made operational, the common property resource must be exploited and shared between the partners. Each party wants to fulfill its requirements for observations and a timetable should be developed to schedule all these requirements. Such a timetable needs to be (a) *efficient* in the sense that the satellites are maximally utilized with the highest priority tasks; and (b) considered *fair* by all parties [2].

While the literature on multi-satellite scheduling is rich, as summarized in [43], most of the solutions are centralized algorithms that assume all the requests reach a central entity that creates a timetable for them. Such solutions are prone to privacy issues since EOSs can be used for defense and security purposes and most of the parties do not want others to be informed of the way they are using the satellites. This raises the need for distributed scheduling solutions [35]. Since a given request can sometimes be satisfied by several satellites in more than one of their orbits, the problem is not separable by satellite nor by orbit. Instead, the scheduling process must be performed simultaneously for all satellites and orbits considered.

Distributed constraint optimization problem (DCOP) [15,9] is a powerful framework for representing and solving distributed combinatorial problems. DCOPs have been successfully applied in a variety of real-world problem domains, including meeting scheduling [28], traffic-light synchronization [18], sensor networks [7], and the Internet of Things [24]. Recently, Picard et al. [35] suggested the use of DCOPs as a potential approach for dealing with EOS scheduling problems. Following this, a DCOP-based solution has been proposed [34]. That work deals with the problem of coordinating users having reserved exclusive orbit portions and one central planner having several requests that may use some intervals of

these exclusives. Their solution enables the exclusive users to independently plan their own tasks; however, the rest of users' tasks are still scheduled in a central manner.

Contributions: Our paper goes one step further. We present a novel Distributed Satellite Timetable Solver that we name DSTS. The DSTS method works in two phases – it starts by mapping the EOS scheduling problem to a DCOP according to our new proposed modeling, and then it solves the DCOP using a DCOP algorithm. The DSTS method is beneficial for two main reasons. First, the EOS scheduling problem is distributed by nature, and as such, it can be modeled and solved in a distributed manner, such as DCOP. In particular, users do not need to a priori share their private information with some central authority. Second, DCOP is a well-established model that provides a wide palette of algorithms, as well as common metrics and simulation environments. Specifically, our method is modular as it is not limited to a specific algorithm.

The rest of the paper is organized as follows. Section 2 provides an overview of relevant existing studies. Section 3 includes formal definitions of the EOS scheduling problem, which is at the focus of this study. Section 4 provides the definition of DCOP and the DCOP modeling of the EOS scheduling problem. An experimental evaluation is presented in Section 5, followed by the conclusions in Section 6.

2 Related Work

A number of EOS scheduling solutions have been proposed. A detailed literature review is presented in [43]. That review classifies the algorithms into four classes: exact methods, heuristics, meta-heuristics, and machine learning based algorithms. Exact methods can provide optimal or near optimal solutions but they are limited to relatively small scale instances. Heuristic methods are employed when exact methods cannot be used; these are further classified into constructive heuristics and time-efficient heuristics. The methods are easily implemented and have relatively short computational time. However, they are specifically designed and there is no guarantee of solution quality. Meta-heuristics are general procedures that find, generate, or select a search algorithm that may provide a sufficiently high-quality solution to an optimization problem. Evolutionary algorithms and single-point search algorithms are two main examples of this category. Machine learning methods have been recently proposed for solving the EOS scheduling problem. These include deep reinforcement approaches [44,17] and competitive learning strategies [26]. The downside of such methods is the requirement for large amount of data to provide good solutions.

All of the above are *static* solutions in the sense that all problem inputs are given in advance. Another research direction deals with the requirement for *dynamic* scheduling [41]. Some examples in which dynamic scheduling is necessary include incoming emergency tasks [36], impact of clouds [40,42], and impact of transition time between observations [45,46].

Most of the known solutions for EOS scheduling problems are centralized algorithms in which the assumption is that satellite constellations are shared resources managed by a central mission control that receives all the requests for observations and schedules them. Recently, a few distributed methods have been proposed [25,4]. These works distribute the scheduling problem between the satellites in attempt to provide rapid response to dynamic changes. However, such form of distribution does not deal with the privacy issue of the users. In a pioneer work, Picard [34] investigates the use of multi-agent allocation techniques for solving the EOS scheduling problem. One of the proposed techniques is DCOP-based. It works under the assumption that some users (termed exclusive users) have reserved exclusive orbit portions. A central planner collects the requests of non-exclusive users and sends them to the exclusive users. The exclusive users employ a distributed procedure that attempts to sequentially schedule those requests, each time creating a DCOP instance in an attempt to add a new request. In this case, the exclusive users retain their privacy while the non-exclusive users still need to share their tasks with a central mission control and thus lose their privacy. To the best of our knowledge, there are no studies that fully distribute the problem among all users.

In order to obtain a fully distributed solution for all users, we suggest to model the whole problem as a DCOP. A major benefit of modeling a problem as a DCOP is that once modeled, the problem can by solved by a wide variety of algorithmic approaches, either complete [31,32,11,48] or incomplete [50,8,20]. DCOPs are NP-hard. Therefore, the use of incomplete approaches is required when solving medium- to large-scale problems.

3 Problem Definition

This section provides the core definitions of the problem we investigate. Consider a set of independent users, each of which generates a set of observation requests. Each request requires viewing a specific area for a specific duration of time, i.e., a request for a specific latitude-longitude-altitude position (LLA) at a certain time interval. These requests potentially yield several observation opportunities per request. In order to define the set of opportunities for each request, the following parameters are required: the LLA to observe, the satellite's location (defined by its orbit plan), and the attitude adjustment capabilities of the satellite's camera. A user can generate a set of observation opportunities for each of its requests using these three parameters. Each such opportunity can be served by a specific satellite at a specific time. The duration of an opportunity is determined by the duration of the request with the addition of the setup time required by the satellite. The extent of the setup time depends on the previous viewing angle of the satellite before it attended to the opportunity at hand. Each user assigns a certain utility, herein termed reward, to its requests. The goal is to find a timetable in which the sum of rewards due to handled requests is maximal. Based on the core definitions in [34], we present the following definitions.

Definition 1 (Earth Observation Satellite Scheduling Problem). *An Earth Observation Satellite Scheduling Problem $\mathcal{P}$ is defined by a tuple*

$$\mathcal{P} =< \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} >$$

where $\mathcal{S}$ is a set of satellites, $\mathcal{U}$ is a set of users, $\mathcal{R}$ is a set of observation requests, and $\mathcal{O}$ is a set of observation opportunities for fulfilling the requests in $\mathcal{R}$.

Definition 2 (Satellite). *A satellite $s \in \mathcal{S}$ is defined as a tuple*

$$s =< OP_s, \kappa_s, ST_s >$$

where OP_s is the orbit plan of the satellite. $\kappa_s \in \mathbb{N}^+$ is the satellite capacity (i.e. the maximum number of observations during its orbit plan). The transition time between two given observations is $ST_s \colon \mathcal{O} \times \mathcal{O} \to \mathbb{R}$.

For simplicity, we assume that the timeline is divided into discrete steps; τ denotes the time period index, see [43].

Definition 3 (User). *A user $u \in \mathcal{U}$ is defined by its set of requests $\mathcal{R}_u \subset \mathcal{R}$.*

Definition 4 (Request). *A request $r \in \mathcal{R}$ is a tuple*

$$r =< t_r^{start}, t_r^{end}, \Delta_r, \rho_r, p_r, u_r, \Theta_r >$$

where the request validity time window is $t_r^{start} \in \mathcal{T}$ and $t_r^{end} \in \mathcal{T}$, $\Delta_r \in \mathcal{T}$ is the required observation duration, $\rho_r \in \mathbb{R}$ is the reward obtained once the request is fulfilled, p_r is the latitude-longitude-altitude position (LLA) to observe, $u_r \in \mathcal{U}$ is the requesting user, and finally, $\Theta_r \subseteq \mathcal{O}$ is the list of opportunities to fulfill the request.

Definition 5 (Opportunity). *An opportunity is defined as a tuple*

$$o =< t_o^{start}, \Delta_o, r_o, \rho_o, s_o >$$

where $t_o^{start} \in \mathcal{T}$ is the opportunity start time, $\Delta_o \in \mathcal{T}$ is the opportunity duration, r_o is the request to which this opportunity contributes, ρ_o is the reward that this opportunity will provide (it is based on ρ_r with modifications according to the observation angle, weather conditions, etc.), s_o is the satellite on which this opportunity can be scheduled.

Definition 6 (Solution). *A solution to a EOS scheduling problem is a feasible subset of opportunities $\mathcal{M} = \{o \in \mathcal{O}\}$, so there is at most one observation per request, and the overall reward (i.e., the sum of the rewards of all scheduled observations) is maximized:*

$$\arg\max_{\mathcal{M}} \sum_{o \in \mathcal{O}} \rho_o x_o$$

subject to

$$x_o \in \{0,1\}\ \forall o \in \mathcal{O} \tag{1}$$

$$\sum_{o \in \Theta_r} x_o \leq 1\ \forall r \in \mathcal{R} \tag{2}$$

$$s_{o_i} \neq s_{o_j} \vee [int_{o_i} + ST_{s_{ij}}] \cap [int_{o_j} + ST_{s_{ji}}] = \emptyset\ \forall o_i, o_j \in \mathcal{M} \tag{3}$$

$$int_o \coloneqq [t_o^{start}, t_o^{start} + \Delta_o] \tag{4}$$

where x_o is the decision variable that defines whether opportunity o is included in $\mathcal{M}$ or not. Equation 2 ensures at most one opportunity for each request is included in the solution. Finally, Equations 3 and 4 avoid overlaps between opportunities, by verifying that the opportunities are either in different satellites or their time intervals including transition time (int_o) do not overlap.

4 DCOP Modeling of the Problem

Now we model the problem as a DCOP. We first provide the formal definition of a DCOP, followed by our proposed DCOP modeling of the EOS scheduling problem. This modeling is the first phase of the proposed DSTS method.

4.1 DCOP Definition

A *DCOP* is a tuple $< \mathcal{A}, \mathcal{X}, \alpha, \mathcal{D}, \mathcal{R} >$, where: $\mathcal{A}$ is a finite set of agents $A_1, A_2, ..., A_n$; $\mathcal{X}$ is a finite set of variables $X_1, X_2, ..., X_m$; $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ maps each variable to one agent; $\mathcal{D}$ is a set of domains $D_1, D_2, ..., D_m$, where each domain D_i consists of a finite set of values that can be assigned to variable X_i; $\mathcal{R}$ is a set of relations (constraints), where each constraint $C \in \mathcal{R}$ is a function $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}_+$ that defines a non-negative *cost* for every possible value combination of a set of variables. A *complete assignment* consists of assignments to all variables in $\mathcal{X}$. An *optimal solution* to a DCOP is a complete assignment of minimal cost.

The advantage of DCOP representation over the classical constraint optimization problems is in the ability to solve the problem in a distributed manner. DCOP researchers have proposed a wide variety of solution approaches, such as search-based algorithms [10,31,48], logic programming [22], inference-based algorithms [33], and other methods [30]. In all of these approaches, each agent handles its own variables and exchanges messages with the other agents in order to determine the final variable values. No centralized entity takes part in the solving process.

In this work we used a search-based approach, specifically, the Distributed Stochastic Algorithm (DSA) [50]. DSA is a standard incomplete DCOP algorithm. This algorithm operates as follows. In each step, the state of every variable (the current value assignment) is shared with its neighbors (the variables with which it is constrained) by sending and receiving messages. After receiving the

updated states of its neighbors, each agent decides whether to change the current state of the variable in an attempt to reduce its costs – this decision is the most fundamental step in DSA. If the agent cannot find a new value in its domain to improve its current state, it has no reason to change its current value. However, in case there is such a value that improves the state, the agent may or may not change to the new value based on a stochastic scheme that prevents scenarios of infinite alteration loops. The probability of parallelism, p, controls how frequently neighboring variables can change their values. The p probability is set as a parameter of the DSA algorithm. The algorithm stops its execution after pre-defined number of iterations.

4.2 EOS Scheduling as a DCOP model

We first provide the mapping of the EOS scheduling problem onto a DCOP model, and then present an example that illustrates the mapping. The mapping is performed in the following manner: A user is mapped to an agent in the DCOP: $\mathcal{A} \coloneqq \{u \in \mathcal{U}\}$ Each user maps its requests to variables in the DCOP: $\mathcal{X} \coloneqq \{r \in \mathcal{R}\}$. Because $\mathcal{X} = \mathcal{R}$, we use these notations interchangeably. The α mapping is performed according to the set of requests $\mathcal{R}_u$ of each user u. The domain $\mathcal{D}$ of each variable is defined by the set of available opportunities Θ_r to fulfill the request r with the addition of the value 0. $\mathcal{D}\colon = \{\Theta_r \cup \{0\} \mid \forall r \in \mathcal{R}\}$. Each value represents one available opportunity for this request. The additional 0 value represents situations in which this request cannot be fulfilled.

Two types of constraints are defined for each user for each of its variables:

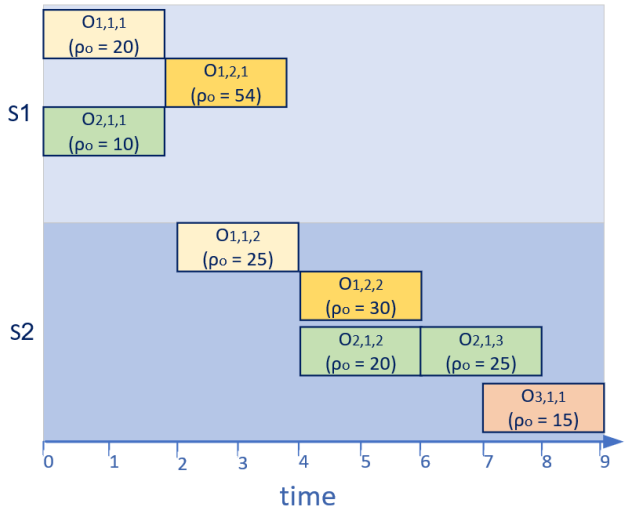
1. A **unary constraint** assigns a cost for each value of the variable based on the reward of the corresponding opportunity. DCOPs are commonly used as minimization problems and require non-negative constraints. Accordingly, we define the cost of opportunity o as follows:

$$cost_o = maxCost - \rho_o \tag{5}$$

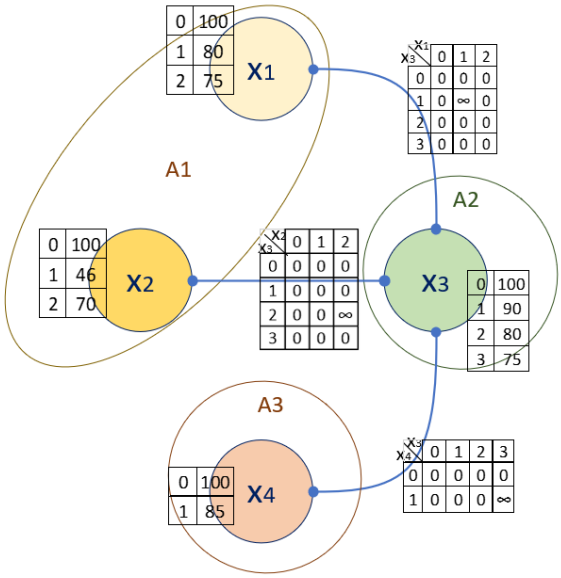
   where $maxCost$ is greater than the reward of any opportunity:

$$maxCost > \max_o \rho_o$$

   Specifically, the cost of 0 value is $maxCost$. Thus, the algorithm will assign a 0 only when there is no possibility to assign any of the opportunities, i.e., the request cannot be fulfilled.
2. **Binary constraints** are defined between two variables if there is overlap between the opportunities they represent. Overlap occurs when the two opportunities are served by the same satellite and their schedule times overlap. The cost of each combination of values is ∞ if there is overlap between the opportunities that these values represent; otherwise, the cost is zero. The setup time between the opportunities, as defined in the ST_s table of the satellite, is considered when computing the overlap. The cost of the 0 value is zero for any combination with the other variable's values.



(a) An EOS scheduling problem. S1 and S2 are two different satellites, rectangles represent opportunities, and the color of a rectangle relates to the request that the opportunity can fulfill.



(b) The DCOP modeling of the above problem. A_1, A_2, and A_3 are the agents (users) and $x_1, \ldots, x_4$ are the variables (requests). The tables represent the constraints.

Fig. 1: An example of an EOS scheduling problem mapped to a DCOP.

An example of an EOS scheduling problem and its DCOP modeling are shown in Figure 1. Figure 1a depicts an EOS scheduling problem with two satellites (s_1 and s_2) and three users (u_1, u_2, and u_3). User u_1 has two requests: $r_{1,1}$ and $r_{1,2}$ each of which has two opportunities: $o_{1,1,1}$ and $o_{1,1,2}$ for $r_{1,1}$ and $o_{1,2,1}$ and $o_{1,2,2}$ for $r_{1,2}$. The second user, u_2, has a single request, $r_{2,1}$ with three opportunities: $o_{2,1,1}$, $o_{2,1,2}$, and $o_{2,1,3}$. The third user, u_3, has a single request, $r_{3,1}$ with a single opportunity $o_{3,1,1}$. In the figure, the opportunities are shown with their schedules and the rewards.

Figure 1b describes the DCOP modeling of this problem. This DCOP contains three agents (A_1, A_2, and A_3) representing the users (U_1, U_2, and U_3). Four variables (x_1, x_2, x_3, and x_4) correspond to the four requests and their domains are defined according to the number of opportunities that can fulfill each request. The domain of x_1 consists of three values: two values (1 and 2) for the opportunities that can fulfill it and the 0 value for the option of not scheduling this request. For the same reasons the domain of x_2 has three values, the domain of x_3 has four values, and the domain of x_4 has two values. Each variable has a unary constraint that assigns a cost to each value. The cost of the 0 value is always $maxCost$ ($maxCost = 100$ in this example). The costs of the other values are defined by Equation 5. For example, $o_{1,1,1}$ has a reward of $\rho_o = 20$, so the cost of value 1 of x_1 is 80. Binary constraints are defined between requests that their opportunities overlap. For example, there is a constraint between x_1 and x_3 because $o_{1,1,1}$ and $o_{2,1,1}$ are scheduled at the same time and on the same satellite (s_1). The cost of the combination is ∞. The other combinations have zero cost since there are no conflicts between these potential opportunities. Both constraint types are illustrated in the tables. The unary constraints tables contain rows for each of the opportunities and an additional row for the 0 value. Each row contains a single value: the unary cost of this opportunity. For example, the unary constraint table of X_1 contains three rows (two for the two opportunities and one for the 0 value) with the value 100 in the first row for the 0 value, 80 in the second row as the cost value of $o_{1,1,1}$ and 75 in the third row as the cost value of $o_{1,1,2}$. The binary constraints are illustrated by two-dimensional tables that are linked to an arc that connects two variables. Each cell in the tables represents the possibility of overlap between two opportunities. For example, in the binary constraints table of $x_1 - x_3$, all the cells have zero value, except for the cell that represents the combination of the $o_{1,1,1}$ and $o_{2,1,1}$ opportunities; that cell has the ∞ value since only this combination generates an overlap.

5 Evaluation

We first describe our experimental setup (Section 5.1) and then the obtained results (Section 5.2).

5.1 Experimental Setup

In order to evaluate DSTS, we generated EOS scheduling problems. For the first phase of DSTS, we mapped the generated problems into DCOPs, as described

in Section 4.2. For the second phase of DSTS, in which the resulting DCOP model should be solved by some DCOP algorithm, we used the DSA algorithm (as detailed in Section 4.1). DSA is suitable for our purposes for several reasons. First, DSA has a low communication and computation overhead, which is required when dealing with problems with hundreds of variables. Second, DSA operates simultaneously by all agents with no pre-defined ordering or structure[1], which provides some sense of fairness.

We compare our DSTS method to a greedy benchmark algorithm (denoted GREEDY), which is used by most satellite and constellation operators [5]. GREEDY sorts all opportunities in increasing order according to their start time and then by their reward (in decreasing reward order). Then, for each opportunity in this sorted list, if there is a free slot for it on its satellite then it is scheduled and all other opportunities of the same request are deleted; otherwise, this opportunity is deleted.

Following [34], we generated two sets of experiments:

1. **Highly conflicting problems**: small-scale problems (5 minutes planning horizon) with 3 satellites, 8 users emitting 2 to 20 requests each ($|R_u| = 2, 4, \ldots, 20$), and 10 observation opportunities per request. The requests validity time window varied in the range $[10 : 20]$ and its duration was set to $\tau = 5$, with reward $\rho_o = [10 : 50]$ meaning a reward was sampled uniformly at random between 10 and 50. This set of problems yields many overlaps between observation opportunities, thus it produces tight problems.

2. **Realistic problems**: large-scale problems in a 6-hour planning horizon. We generated instances with 8 satellites, 6 users emitting 10 to 100 requests each ($|R_u| = 10, 20, \ldots, 100$), and 5 observation opportunities per request. The requests validity time window varied in the range $[40 : 60]$ and its duration was set to $\tau = 20$, with reward $\rho_o = [10 : 50]$. This set of problems has many observation opportunities (up to 3000) but with less overlaps between them (a sparse setting).

We generated 100 instances of each setting, which resulted in a total of 2000 generated problems. All experiments were performed on the 'AgentZero' simulator [27].[2] All experiments were run on a standard laptop (Lenovo T14 with Intel(R) Core(TM) i7-10610U CPU running at 1.80GHz) with Win10 OS and took a few minutes in total.

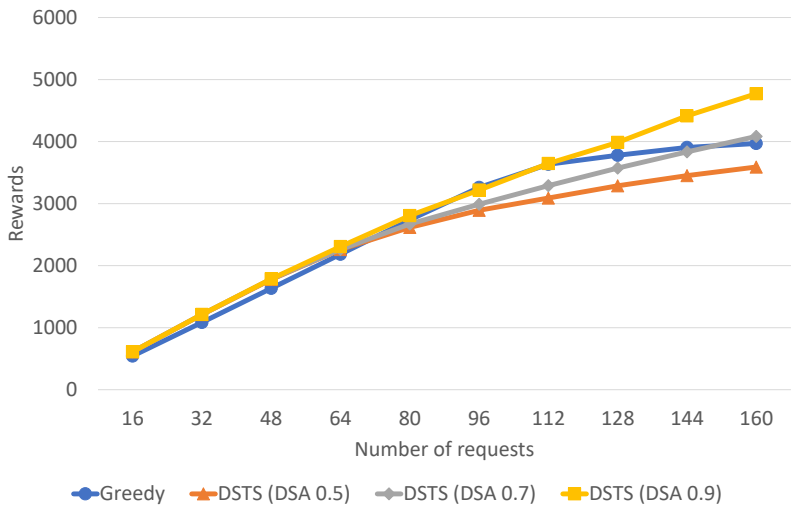


Fig. 2: Rewards of highly conflicting problems

5.2 Experimental Results

We first examined the rewards of highly conflicting problems. Figure 2 presents a comparison of DSTS and Greedy. We used the DSA algorithm to solve the DCOP. Specifically, we used the DSA-B version [50] with three different p values, $p = \{0.5, 0.7, 0.9\}$, and ran it for 10 iterations. Axis x presents the number of requests, from 16 (eight users with two requests each) to 160 (eight users with 20 requests each). Axis y presents the sum of the rewards over all scheduled requests.

For a low number of requests the results of the two compared methods are similar, but for a high number of requests DSTS (using DSA with $p = 0.9$) outperforms Greedy. The best results for DSTS are obtained with $p = 0.9$. These findings are consistent with known results regarding the probability of parallelism in DSA [50]; higher parallelism usually leads to higher quality solutions until a 'phase transition' is reached, after which the solution quality drops drastically. In particular, Zhang et al. [50] showed that for the DSA-B version the phase transition commonly occurs when $p > 0.9$, in consistence with our results.

[1] Some DCOP algorithms like SyncBB [15] and AFB [11] maintain a pre-defined ordering of the agents, while others, such as DPOP [32] and BnB-ADOPT [48], operate on a tree structure.

[2] AgentZero is a Java-based programming framework for research and implementation of multi-agent problems, and particularly DCOPs. It enables to generate various types of multi-agent problems, test distributed algorithms, and collect various performance statistics.

Table 1 shows the average number of messages exchanged during the run of DSTS using DSA with $p = 0.9$; GREEDY is a centralized algorithm, thus it does not exchange messages. Table 2 compares the run-times of DSTS (using DSA with $p = 0.9$) and GREEDY. As can be seen, DSTS is more affected by the size of the problem. Yet, it is still very fast (less than 100 milliseconds for the largest problems in this set).

Requests	16	32	48	64	80	96	112	128	144	160
DSTS	327	1782	5349	11390	19758	30054	42813	56748	73287	92281

Table 1: Number of messages in highly conflicting problems

Requests	16	32	48	64	80	96	112	128	144	160
DSTS	0.33	0.92	1.8	3.92	8.6	16.3	24.2	36	52	97
GREEDY	0.15	0.12	0.13	0.16	0.22	0.32	0.36	0.42	0.49	0.69

Table 2: Run-time of highly conflicting problems (milliseconds)

Next, we examined realistic problems. We used the DSA algorithm to solve the DCOP with the same three p values as before. The results are displayed in Figure 3. Axis x presents the number of requests, from 60 (six users with ten requests each) to 600 (six users with 100 requests each). Again, axis y presents the sum of the rewards over all scheduled requests. Here, all three versions of DSA obtain similar rewards. However, these rewards are $\sim 6\%$ better than those of GREEDY . These findings can be explained by the relative sparsity of constraints in this set, which results in problems that are more easily solved by both GREEDY and the various DSTS versions. Still, DSTS outperforms GREEDY, since even in such sparse settings there are a few conflicts that GREEDY fails to resolve due to its simplistic and obviously greedy nature.

Table 3 shows the average number of messages exchanged during the run of DSTS using DSA with $p = 0.9$ and Table 4 presents the run-time comparison with GREEDY. Here, the run-time of DSTS is only slightly higher than that of GREEDY. Note that realistic problems produce less messages and run faster than the conflicting problems (cf. Tables 1 and 2); this further indicates that density is an extremely important factor. Another conclusion that can be drawn from these results is that due to their sparseness, realistic problems of much larger sizes (in terms of numbers of requests) could be solved in practice using DSTS.

6 Conclusions

In this work we proposed DSTS, a novel method that models *EOS* scheduling problems as DCOPs and solves them using standard DCOP algorithms. Our

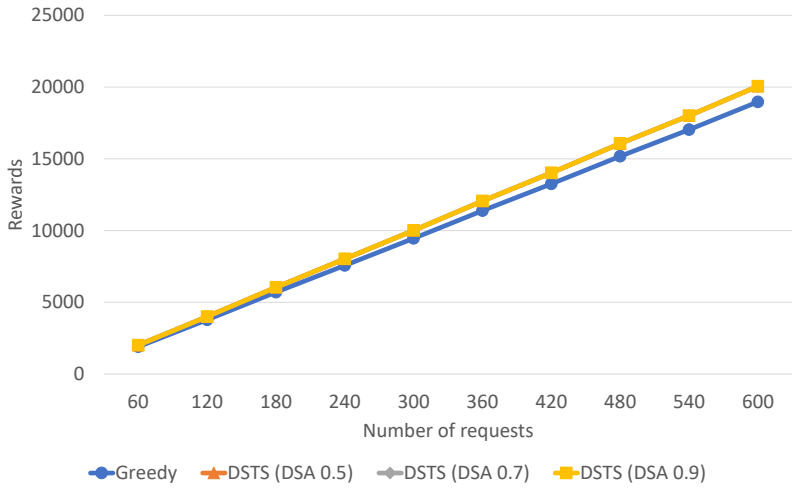


Fig. 3: Rewards of realistic problems

Requests	60	120	180	240	300	360	420	480	540	600
DSTS	18.6	77.7	181.5	321.4	514.6	765	1056	1393	1817	2283

Table 3: Number of messages in realistic problems

experiments reveal that DSTS fastly solves the EOS scheduling problems and provides higher quality solutions than the Greedy benchmark algorithm currently used.

Modeling the EOS scheduling problem as a DCOP is natural since the problem is inherently distributed. Moreover, by applying an algorithm with no predefined ordering or structure, such as DSA, all users "have the same starting point" when participating in the solution process, which is considered fair [1]. It should be noted that even though the users do not need to a priori share their private information with some central authority, some private information may be leaked during the DCOP solving process [29,12]. In situations where privacy is an important issue, one may resolve to using a privacy-preserving DCOP algorithm in phase two of DSTS. One may choose a complete privacy-preserving algorithm (e.g., [23,13]) or preferably an incomplete one (e.g., [38,14]) for appli-

Requests	60	120	180	240	300	360	420	480	540	600
DSTS	0.17	0.2	0.32	0.37	0.47	0.53	0.65	0.85	0.84	1.12
Greedy	0.23	0.16	0.24	0.26	0.29	0.35	0.43	0.52	0.58	0.72

Table 4: Run-time of realistic problems (milliseconds)

cability reasons, as privacy preservation comes with a price tag of considerably higher overheads.

An interesting feature of the problem at hand is that users do not know in advance with whom they are constrained, which is considered trivial information in other problem domains (e.g., meeting scheduling [28]). This problem can be handled in a privacy-preserving manner by employing standard multi-party computation methods. Basically, each pair of users has to construct a Boolean or arithmetic circuit that represents the time intervals of the observation opportunities. That circuit can be evaluated using techniques of cryptography (e.g., garbled circuits [47]) to obtain mutual information of the overlaps without learning anything else. Despite the cryptographic workload, such a preprocessing stage is performed only once by each pair of users ($\mathcal{O}(n^2)$) and, therefore, does not heavily influence the overall performance. Another solution for the preprocessing stage is to delegate these computations to a set of external mediators. Recent studies in the DCOP field showed that such mediators may perform the computations in an oblivious manner, without gaining access neither to the problem inputs nor to its outputs [37,21] (this is in contrast to the centralized approach in which the central authority is exposed to the inputs and outputs). An interesting direction for future work is to devise a mediation-based solution in which the mediators can perform both the preprocessing stage and the DCOP-solving stage.

In this work we do not assume that the satellites have limited capacity; however, some real-world satellites do have such limitation. Applying this limitation to our model is not trivial since it requires adding global constraints, which are known to impair the performance. Nonetheless, a solution to a similar problem has been successfully applied recently, including the development of new variation of DSA that focuses on problems with limited capacity [19]. Therefore, employing a similar solution in DSTS is another prospect for future work. Yet another issue is that of dynamic changes; satellites are often affected by environmental changes (e.g., clouds) or may receive emergency requests. We, therefore, plan to follow recent advances on dynamic DCOPs [16] to enable dynamic modifications of the timetables.

Acknowledgments This work was partially supported by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office.

References

1. Abdulkadiroğlu, A., Sönmez, T.: Random serial dictatorship and the core from random endowments in house allocation problems. Econometrica **66**(3), 689–701 (1998)
2. Bataille, N., Lemaitre, M., Verfaillie, G.: Efficiency and fairness when sharing the use of a satellite. In: Artificial Intelligence, Robotics and Automation in Space. vol. 440, p. 465 (1999)

3. Bianchessi, N., Cordeau, J.F., Desrosiers, J., Laporte, G., Raymond, V.: A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites. European Journal of Operational Research **177**(2), 750–762 (2007)
4. Braquet, M., Bakolas, E.: Greedy decentralized auction-based task allocation for multi-agent systems. IFAC-PapersOnLine **54**(20), 675–680 (2021)
5. Cho, D.H., Kim, J.H., Choi, H.L., Ahn, J.: Optimization-based scheduling method for agile earth-observing satellite constellation. Journal of Aerospace Information Systems **15**(11), 611–626 (2018)
6. Denis, G., Claverie, A., Pasco, X., Darnis, J.P., de Maupeou, B., Lafaye, M., Morel, E.: Towards disruptions in earth observation? new earth observation systems and markets evolution: Possible scenarios and impacts. Acta Astronautica **137**, 415–433 (2017)
7. Farinelli, A., Rogers, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: AAMAS. pp. 639–646 (2008)
8. Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2. pp. 639–646 (2008)
9. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research **61**, 623–698 (2018)
10. Gershman, A., Meisels, A., Zivan, R.: Asynchronous forward-bounding for distributed constraints optimization. In: Proc. ECAI-06. pp. 103–107. Lago di Garda (August 2006)
11. Gershman, A., Meisels, A., Zivan, R.: Asynchronous forward bounding. Journal of Artificial Intelligence Research **34**, 25–46 (2009)
12. Greenstadt, R., Pearce, J.P., Tambe, M.: Analysis of privacy loss in distributed constraint optimization. In: AAAI. vol. 6, pp. 647–653 (2006)
13. Grinshpoun, T., Tassa, T.: P-SyncBB: A privacy preserving branch and bound DCOP algorithm. Journal of Artificial Intelligence Research **57**, 621–660 (2016)
14. Grinshpoun, T., Tassa, T., Levit, V., Zivan, R.: Privacy preserving region optimal algorithms for symmetric and asymmetric DCOPs. Artificial Intelligence **266**, 27–50 (2019)
15. Hirayama, K., Yokoo, M.: Distributed partial constraint satisfaction problem. In: CP. pp. 222–236 (1997)
16. Hoang, K.D., Hou, P., Fioretto, F., Yeoh, W., Zivan, R., Yokoo, M.: Infinite-horizon proactive dynamic DCOPs. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 212–220 (2017)
17. Huang, Y., Mu, Z., Wu, S., Cui, B., Duan, Y.: Revising the observation satellite scheduling problem based on deep reinforcement learning. Remote Sensing **13**(12), 2377 (2021)
18. Junges, R., Bazzan, A.L.: Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2. pp. 599–606 (2008)
19. Khakhiashvili, I., Grinshpoun, T., Dery, L.: Course allocation with friendships as an asymmetric distributed constraint optimization problem. In: 2021 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). IEEE (2021)
20. Kiekintveld, C., Yin, Z., Kumar, A., Tambe, M.: Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In: AAMAS. vol. 10, pp. 133–140 (2010)

21. Kogan, P., Tassa, T., Grinshpoun, T.: Privacy preserving DCOP solving by mediation. In: International Symposium on Cyber Security Cryptography and Machine Learning. Springer (2022)
22. Le, T., Son, T.C., Pontelli, E., Yeoh, W.: Solving distributed constraint optimization problems using logic programming. Theory and Practice of Logic Programming **17**(4), 634–683 (2017)
23. Léauté, T., Faltings, B.: Protecting privacy through distributed computation in multi-agent decision making. Journal of Artificial Intelligence Research **47**, 649–695 (2013)
24. Lezama, F., Palominos, J., Rodríguez-González, A.Y., Farinelli, A., Munoz de Cote, E.: Agent-based microgrid scheduling: An ict perspective. Mobile Networks and Applications **24**(5), 1682–1698 (2019)
25. Liu, L., Dong, Z., Su, H., Yu, D.: A study of distributed earth observation satellites mission scheduling method based on game-negotiation mechanism. Sensors **21**(19), 6660 (2021)
26. Liu, Y., Chen, Q., Li, C., Wang, F.: Mission planning for earth observation satellite with competitive learning strategy. Aerospace Science and Technology **118**, 107047 (2021)
27. Lutati, B., Gontmakher, I., Lando, M., Netzer, A., Meisels, A., Grubshtein, A.: AgentZero: A framework for simulating and evaluating multi-agent algorithms. Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks pp. 309–327 (2014)
28. Maheswaran, R.T., Tambe, M., Bowring, E., Pearce, J.P., Varakantham, P.: Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In: AAMAS. pp. 310–317. New York, NY, USA (2004)
29. Maheswaran, R.T., Pearce, J.P., Bowring, E., Varakantham, P., Tambe, M.: Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications. Autonomous Agents and Multi-Agent Systems **13**(1), 27–60 (2006)
30. Mailler, R., Lesser, V.: Asynchronous Partial Overlay: A New Algorithm for Solving Distributed Constraint Satisfaction Problems. Journal of Artificial Intelligence Research **25**, 529–576 (April 2006), http://mas.cs.umass.edu/paper/397
31. Modi, P.J., Shen, W., Tambe, M., Yokoo, M.: Adopt: asynchronous distributed constraints optimization with quality guarantees. Artificial Intelligence **161**(1-2), 149–180 (2005)
32. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: IJCAI. pp. 266–271. Edinburgh, Scotland, UK (2005)
33. Petcu, A., Faltings, B.: ODPOP: An algorithm for open/distributed constraint optimization. In: AAAI. pp. 703–708. Boston, MA, USA (2006)
34. Picard, G.: Auction-based and distributed optimization approaches for scheduling observations in satellite constellations with exclusive orbit portions. arXiv preprint arXiv:2106.03548 (2021)
35. Picard, G., Caron, C., Farges, J.L., Guerra, J., Pralet, C., Roussel, S.: Autonomous agents and multiagent systems challenges in earth observation satellite constellations. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021). pp. 39–44 (2021)
36. Sun, H., Xia, W., Wang, Z., Hu, X.: Agile earth observation satellite scheduling algorithm for emergency tasks based on multiple strategies. Journal of Systems Science and Systems Engineering **30**(5), 626–646 (2021)
37. Tassa, T., Grinshpoun, T., Yanai, A.: PC-SyncBB: a privacy preserving collusion secure DCOP algorithm. Artificial Intelligence **297**, 103501 (2021)

38. Tassa, T., Grinshpoun, T., Zivan, R.: Privacy preserving implementation of the Max-Sum algorithm and its variants. Journal of Artificial Intelligence Research **59**, 311–349 (2017)
39. Walker, J.G.: Satellite constellations. Journal of the British Interplanetary Society **37**, 559 (1984)
40. Wang, J., Demeulemeester, E., Hu, X., Wu, G.: Expectation and saa models and algorithms for scheduling of multiple earth observation satellites under the impact of clouds. IEEE Systems Journal **14**(4), 5451–5462 (2020)
41. Wang, J., Zhu, X., Yang, L.T., Zhu, J., Ma, M.: Towards dynamic real-time scheduling for multiple earth observation satellites. Journal of Computer and System Sciences **81**(1), 110–124 (2015)
42. Wang, X., Gu, Y., Wu, G., Woodward, J.R.: Robust scheduling for multiple agile earth observation satellites under cloud coverage uncertainty. Computers & Industrial Engineering **156**, 107292 (2021)
43. Wang, X., Wu, G., Xing, L., Pedrycz, W.: Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. IEEE Systems Journal **15**(3), 3881–3892 (2020)
44. Wei, L., Chen, Y., Chen, M., Chen, Y.: Deep reinforcement learning and parameter transfer based approach for the multi-objective agile earth observation satellite scheduling problem. Applied Soft Computing **110**, 107607 (2021)
45. Wei, L., Xing, L., Wan, Q., Song, Y., Chen, Y.: A multi-objective memetic approach for time-dependent agile earth observation satellite scheduling problem. Computers & Industrial Engineering **159**, 107530 (2021)
46. Xiang, S., Xing, L., Wang, L., Zhou, Y., Peng, G.: Enhanced pigeon inspired optimisation approach for agile earth observation satellite scheduling. International Journal of Bio-Inspired Computation **17**(3), 131–141 (2021)
47. Yao, A.C.: Protocols for secure computation. In: FOCS. pp. 160–164 (1982)
48. Yeoh, W., Felner, A., Koenig, S.: BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. Journal of Artificial Intelligence Research **38**, 85–133 (2010)
49. Yifang, B., Gong, P., Gini, C.: Global land cover mapping using earth observation satellite data: Recent progresses and challenges. ISPRS journal of photogrammetry and remote sensing (Print) **103**(1), 1–6 (2015)
50. Zhang, W., Wang, G., Xing, Z., Wittenburg, L.: Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. Artificial Intelligence **161**(1-2), 55–87 (2005)