Throughput-Optimal Joint Routing and Scheduling for Low-Earth-Orbit Satellite Networks

Olga Kondrateva*, Holger Döbler*, Hagen Sparka*, Andreas Freimann**, Björn Scheuermann*, and Klaus Schilling**
*Computer Engineering Group, Humboldt University of Berlin, Berlin, Germany
**Chair of Computer Science VII, University of Würzburg, Würzburg, Germany

***Abstract*—When optimizing communication patterns in wireless networks, routing and link scheduling cannot be handled separately but must be addressed jointly. Various linear programming formulations were proposed for static networks to optimize routing while ensuring schedulability of the achieved result. However, most of these approaches do not allow to obtain an optimal transmission schedule directly. Even if they do, they do not scale well to practically relevant network sizes. Node mobility further complicates the effort. Here, we consider satellite networks, which are characterized by time-varying, yet predictable topologies. We present a novel approach for the joint optimization of routing and link scheduling. It is based on linear programming and provides a constructive way to generate transmission schedules. To reduce the computational complexity, we decompose the problem and apply a series of optimization steps resulting in an optimal transmission schedule. As an exemplary use case we optimize the throughput of a network consisting of 18 low-earth-orbit satellites. Our evaluation results validate the optimality of our joint routing and scheduling approach and demonstrate its applicability to a real-world use case.**

I. Introduction

Continuing advances in space technology have led to increased popularity of miniaturized satellites in recent years. Many scientific and commercial missions based on multi-satellite systems have been proposed [1], [2]. Compared to traditional satellite systems consisting of a single large satellite, these systems are more robust, allow for new applications due to their spatial distribution and reduce costs for manufacturing and launch. Multi-satellite systems in Low Earth Orbit (LEO) are proposed for a variety of applications, e. g., for Earth observation and communication [3]–[5].

LEO satellites orbit the Earth several times a day, therefore the contact time to points on the Earth's surface is limited. Due to the relative movement of the satellites in their orbits, there is no continuous network connectivity between the satellites if relative distances exceed the communication range. These factors combined with the limited capabilities of small satellites present interesting challenges for the design and implementation of communication protocols [6].

A key to designing protocols for multi-satellite systems is to make use of the deterministic movement of the satellites. The topology of spatially distributed LEO satellite networks is highly time-varying, but also highly predictable. Inter-satellite links can be used to relay data to satellites within range of a ground station, following a Delay-Tolerant Networking (DTN) paradigm.

Corresponding author: O. Kondrateva (email: https://hu.berlin/ti-members).

In this paper, we consider application scenarios with predictable traffic demand, like for instance periodical sensor measurements. We assume that satellites act as relays to cooperatively transmit data to a number of interconnected ground stations. Based on these assumptions we propose a linear programming formulation for throughput optimization in multi-satellite systems.

Similar problems have been extensively studied in the past in the context of static wireless ad-hoc networks (see related work in Sec. II). Flow optimization in a wireless environment is a complex problem, which requires the joint consideration of interdependent sub-problems like routing, link scheduling, channel assignment and power control. In the present study our focus lies on the joint optimization of routing and link scheduling.

Basically, two kinds of linear programming formulations for wireless network flow optimization can be found in the literature. The first approach models each time slot directly and introduces a separate decision variable for each communication link in each slot [7]–[9]. This allows to directly compute a transmission schedule, but results in a prohibitively large number of decision variables. This makes the approach hardly applicable even for relatively small scenarios.

The second approach is based on finding sets of links that can be scheduled together without conflict [10], so-called independent sets. This information is then used in the linear program formulation. Although computing all not-interfering sets is NP-hard [10] and the corresponding linear program can also contain many decision variables, this formulation is preferable, as it is possible to reduce the computational complexity through heuristics (e. g. the so-called column generation method) [7], [11].

However, the main disadvantage of this second formulation is that it does not result in a complete transmission schedule. By solving a linear program in this formulation, it is possible to calculate the total scheduled time for a given set of links. However, optimal transmission order and duration of a single transmission are not provided. This information is crucial for applying the results in networks with time varying topologies, though.

In this work, we close this gap by proposing a linear programming formulation which is based on calculating independent sets and provides a way to calculate a complete transmission schedule. Our main contributions are twofold. First, we propose a two-step linear optimization method to jointly generate throughput optimal routing and transmission schedules. Second, we propose a novel sliding-window-based approach that addresses the scalability issue of optimizing networks containing a high number of nodes. We evaluate these algorithms using satellite trajectories from a detailed orbit simulation model.

The remainder of this paper is organized as follows. In Sec. II, we review the literature on flow optimization in wireless networks. Sec. III describes our system model. The linear programming formulation for throughput maximization is explained in Sec. IV. In Sec. V, we then present our central contributions: the linear programming formulation and the sliding window-based approach. Finally, Sec. VII describes our evaluation results and Sec. VIII concludes this paper.

II. Related work

Flow optimization by means of routing and scheduling in dynamic wireless networks using linear programming has been studied extensively in the past. A short overview of selected related publications follows.

Kotnyek [12] gives a high-level overview of flow problems in networks that change over time (dynamic network flows). [13], [14], and [15] focus on optimal routing in dynamic networks following the DTN paradigm but do not consider scheduling simultaneously.

For the case of static networks, solutions for the joint routing and scheduling problem have been proposed using linear programming (LP) [10], [16], [17], or mixed integer linear programming (MILP) [7], [18], where the latter is applied if scheduling is to be considered on the granularity of individual packets. The linear column generation method has been used to approximately solve the linear programs [7], [17]. Using the non-linear column generation method, joint routing and scheduling may be considered w.r.t. non-linear optimization objectives [19].

More recently, joint routing and scheduling in dynamic wireless networks has been considered. In [20] the subgradient method is applied. It is assumed that all payload data is available at the node in advance, whereas our approach allows to incorporate payload to be generated at intermediate nodes at any time. In [8], [9] MILP is used in order to consider scheduling with respect to individual time slots. According to the results of [7], this approach comes at prohibitively high computation cost. As opposed to our work, [8], [9] do not consider secondary interference in their network model.

The study of optimal scheduling using ILP is not limited to the wireless multi-hop use-case but has recently also been applied in the context of (wired) time-sensitive software-defined networks [21]–[23].

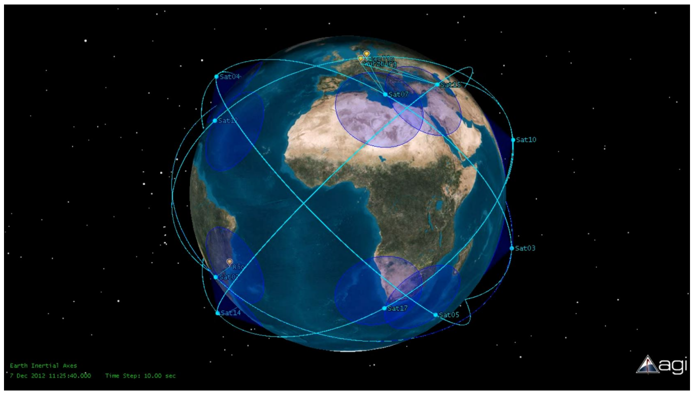


Image courtesy of Analytical Graphics, Inc. (www.agi.com)

Fig. 1: 3D model of the studied scenario consisting of 18 satellites and four ground stations

III. Problem Statement

A. Abstract Mission Model

We assume a constellation of LEO satellites that gather observation data, the payload, that is to be delivered to a network of ground stations (see Fig. 1).

The source traffic rate, i. e. the rate at which data is generated at the satellites, may differ between satellites and vary in time. In our evaluation we consider the case of homogeneous constant bit rate source traffic. Our formulation allows for different shapes of source traffic as well, like bursts, as long as the shape is known ahead of time. Any payload is considered "delivered" once received by any of the ground stations, thus payload is never transmitted from a ground station to a satellite. Satellites are equipped with radio transceivers and omnidirectional antennas which enable sat-to-sat as well as sat-to-ground communication. In addition we assume each satellite to carry a GPS receiver and antenna, which has been done in the past for nanosatellites [24]. We therefore assume the satellites' clocks to be sufficiently synchronized.

A sophisticated dissemination plan for the optimized transmission schedule is beyond the scope of this paper. We assume that during every ground station contact a marginal part of the available communication band width is used for bidirectional satellite control communication anyway. This may be used for dissemination as well. In addition, in-network contact plan dissemination techniques designed for contact graph routing [25] might be applicable.

B. Network Model

For a fixed point in time (the static scenario), communication opportunities are modeled by a connectivity graph $C = (N_C, L_C)$, in which each satellite and ground station is represented by a node (cf. Fig. 2a).

Inter-satellite and downlink communication opportunities are modeled as links of fixed capacity, where Cap_{ij} denotes the capacity of link $(i, j) \in L_C$. The process of payload generation is modeled by a virtual node s that acts as the only source and is connected to each satellite with a directed link whose capacity is the corresponding satellite's data generation rate. Analogously, each ground station is connected to a single virtual destination node d with a virtual link of unlimited capacity. This enables us to treat the situation of multiple real sources and multiple possible destinations efficiently as a single-source-single-destination problem.

We consider the simplified case of unlimited data storage capacity in each satellite. However, handling limited storage appears well feasible and will be a future extension.

C. Transmission and Interference Model

At each point in time, we assume that two satellites can either communicate with a fixed inter-satellite capacity, or cannot communicate at all. Likewise, a satellite can either send data to a ground station with a fixed downlink capacity, or cannot communicate to the ground station at all. These communication opportunities are modeled as directed links in the connectivity graph.

To incorporate interference, we adopt the approach from [10] and introduce a (undirected, unweighted) conflict graph $F = (L_C, L_F)$ whose vertices are the connectivity graph's links. Two vertices are connected by a conflict edge if and only if the corresponding connectivity links interfere with each other, i. e., if data cannot be sent simultaneously via both links. The conflict graph may incorporate primary interference, i. e., the restriction that two links interfere, if they share a common node. It can also include secondary interference, i. e., the interference of two disjoint links.

The connectivity graph as well as the conflict graph are determined by the transmission and interference model in use. To model wireless interference, the protocol model and the physical interference model are often employed [26]. Similar to [10] and [18], we will use the protocol model for our evaluation to reduce computation complexity. The physical model can be incorporated into our linear programming formulation as described in [7].

IV. Throughput Maximization for Mobile Networks

This section describes how a linear programming approach can be used to determine the maximum throughput of a mobile network. First, we review the formulation for static networks proposed by Jain et al. in [10]. Subsequently, we propose our generalization of this linear program to mobile networks.

A. Static Scenario

Recall our notation from Sec. III. For each link $(i, j) \in L_C$ we define a variable f_{ij} which denotes the amount of flow allocated to the corresponding link.

Let $N_C^+(i) := \{j \in N_C : (i, j) \in L_C\}$ denote the set of successors and let $N_C^-(i) := \{j \in N_C : (j, i) \in L_C\}$ denote the set of predecessors of node $i \in N_C$. Further, assume that the entire transmission time is equal to T. In the absence of interference, a linear program maximizing the throughput from $s \in N_C$ to $d \in N_C$ can be formulated as follows [10]:

$$\textbf{maximize} \quad \sum_{j \in N_C^+(s)} f_{sj} \tag{1}$$

subject to (1.1)–(1.5)

$$\sum_{j \in N_C^+(i)} f_{ij} = \sum_{j \in N_C^-(i)} f_{ji}, \qquad \forall i \in N_C \setminus \{s, d\} \tag{1.1}$$

$$\sum_{i \in N_C^-(s)} f_{is} = 0 \tag{1.2}$$

$$\sum_{i \in N_C^+(d)} f_{di} = 0 \tag{1.3}$$

$$f_{ij} \leq \mathrm{Cap}_{ij} \cdot T \qquad \forall (i, j) \in L_C \tag{1.4}$$

$$f_{ij} \geq 0 \qquad \forall (i, j) \in L_C. \tag{1.5}$$

The objective states that the outgoing flow of the source should be maximized. The first constraint guarantees flow conservation, i. e., that all data transmitted by the source will be delivered to the destination node. The second and the third constraints define that the incoming flows of the source and the outgoing flows of the destination have to be equal to zero. Otherwise the linear program could maximize the objective function by creating a circular flow through the source. The fourth constraint is necessary to ensure that capacities of the links are taken into account. Finally, the fifth constraint states that only non-negative flow values are allowed.

This formulation is valid for static, wired networks. To adapt it for wireless communication, interference is taken into account by means of the conflict graph F. The key idea behind the conflict graph approach in the context of linear programming is to find all sets of links that can be active simultaneously. This corresponds to the graph theoretical problem of finding all maximal independent sets (MISs) in the conflict graph [10].

Let K denote the number of MISs in F, and let $I_k : k \in \{1, 2, \dots, K\}$ denote the MISs. For each I_k a new variable $\lambda_k \in [0, 1]$ is defined that denotes the amount of time for which the links in I_k may be active. Each link can be a member of multiple MISs. Therefore, the total amount of time in which a link may transmit data is the sum of the corresponding lambda variables. The capacity constraint (1.4) can hence be reformulated as follows:

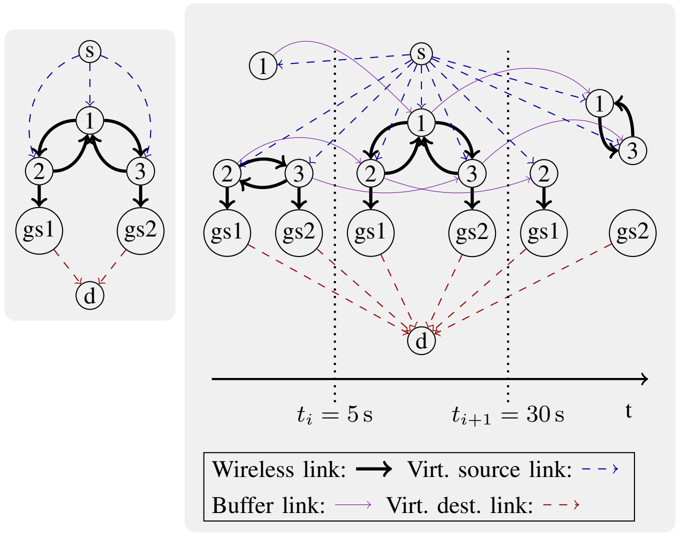
$$f_{ij} \leq \mathrm{Cap}_{ij} \cdot \sum_{\substack{k=1 \\ I_k \ni (i,j)}}^{K} \lambda_k \qquad \text{where} \qquad \sum_{k=1}^{K} \lambda_k = T.$$

The equality constraint here ensures that exactly one independent set is active at each time.

For simplicity, we assume that both f_{ij} and λ_k are continuous, thus allowing arbitrary short transmissions. A similar ILP formulation for scheduling transmissions with packet granularity is discussed in [7]. Applying our results to this formulation is straightforward.

B. Adaptation for Mobile Networks

In a LEO satellite constellation, the wireless connectivity and interference change over time. In our model, these dynamics are captured by changes in the connectivity graph and



(a) Static graph. (b) Time-expanded graph.

Fig. 2: Static and mobile network represented as graphs.

the conflict graph, respectively. Links in these graphs may appear or vanish at discrete, pre-known points in time.

In order to generalize the static formulation to the mobile network, we represent all changes in the topology as a time-expanded connectivity graph $C(p)$ just like the extended time-evolving graphs introduced in [9]. Consider the example shown in Fig. 2b. The entire transmission time $[t_0, t_M)$ is divided into a sequence of consecutive intervals $p = ([t_0, t_1), [t_1, t_2), \ldots, [t_{M-1}, t_M))$, referred to as "time frames", during which the topology, i. e., both connectivity and conflict graph, remain static.

For each time frame, we instantiate a copy of N_C as well as the connectivity links that capture the actual topology. To represent buffering of data across time frames, additional "buffer links" between instances of the same node in subsequent time frames are introduced. The virtual source and destination nodes are not duplicated. Instead, virtual links to all instances of the satellites and ground stations, respectively, are created.

The corresponding conflict graph is of much simpler structure. Neither do virtual links or buffer links interfere with each other or any other link, nor do connectivity links of different time frames interfere with each other. Therefore, the conflict graph corresponding to the time-expanded connectivity graph is simply a sequence of independent static conflict graphs corresponding to the sequence of time frames.

Applying the linear program formulation (1) discussed above to mobile networks in this formulation requires only small changes:

1) Identify the time frames during which the connectivity and interference graphs are static.
2) For each time frame, introduce a copy of the set of network nodes.
3) Compute the independent sets within each time frame.

TABLE I: LP1 and LP2 solution for the network from Fig. 4

LP1		LP2	
$f_{1,2} = 20\,\mathrm{kB}$	$f_{s,2} = 60\,\mathrm{kB}$	$f_{1,2} = 15\,\mathrm{kB}$	$f_{s,2} = 60\,\mathrm{kB}$
$f_{2,1} = 0\,\mathrm{kB}$	$\lambda_1 = 4\,\mathrm{s}$	$f_{2,1} = 0\,\mathrm{kB}$	$\lambda_1 = 5\,\mathrm{s}$
$f_{2,\mathrm{gs}} = 80\,\mathrm{kB}$	$\lambda_2 = 0\,\mathrm{s}$	$f_{2,\mathrm{gs}} = 75\,\mathrm{kB}$	$\lambda_2 = 0\,\mathrm{s}$
$f_{s,1} = 20\,\mathrm{kB}$	$\lambda_3 = 16\,\mathrm{s}$	$f_{s,1} = 15\,\mathrm{kB}$	$\lambda_3 = 15\,\mathrm{s}$
Total throughput: 80 kB		Total throughput= 75 kB	

4) From s, create unidirectional data generation links to each satellite in each time frame; likewise, create data collection links from each ground station in each time frame to d. As these links are virtual, they cannot be in conflict with any other links, so they are part of every MIS of the corresponding time frame.

The linear program for this extended graph is then structurally equivalent to (1) above. The resulting formulation is below referred to as LP1.

V. Deriving Routing Information

The linear program discussed in the previous section is designed to calculate the maximum throughput of a wireless network. In addition, information about the total amount of data transmitted over each link (f_{ij}) and the total amount of time for which each independent set of links should be active (λ_k) is computed. However, for scheduling the order in which the independent sets transmit and the lengths of their respective transmission slots have to be determined as well. Even in simple cases, optimally choosing these quantities can be hard. Consider the example in Fig. 4.

Nodes 1 and 2 both transmit data to node gs. Only node 2 has a direct connection to gs and serves as relay for node 1. The nodes s and d are added to model the process of data generation and collection as discussed in the previous section. It is assumed that nodes are not able to receive and transmit simultaneously (primary interference), which means that all links except for the outgoing links of the node s are in conflict with each other. The MISs are shown in Fig. 4. We solve the linear program as presented above for the following set of values: $\mathrm{Cap}_{1,2} = \mathrm{Cap}_{2,1} = \mathrm{Cap}_{2,\mathrm{gs}} = 5\,\mathrm{kB/s}$, $\mathrm{Cap}_{s,1} = \mathrm{Cap}_{s,2} = 3\,\mathrm{kB/s}$. Furthermore, it is assumed that the total transmission time is 20 s and that the topology remains stable for the whole transmission time.

Solving the LP1 linear program for this setting provides the flows and time slot lengths shown in Tab. I. The maximal achievable throughput is equal to 80 kB. This value coincides with the amount of flow on the link $(2, \mathrm{gs})$, because gs is the only receiver in the network. Due to constraint (1.1) all generated data will be delivered, so 20 kBs generated at the node 1 and 60 kB generated at node 2 will be transmitted to gs.

Although the length of the assigned times slots (λ_k) fit the flow variables (f_{ij}), this information is not sufficient to schedule the transmissions accordingly. For example, assume that node 1 is chosen to transmit first and the length of its

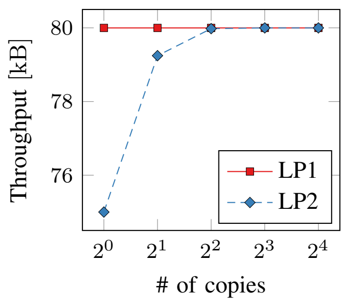


Fig. 3: LP1 and LP2 solution; the latter with N copies of independent sets.

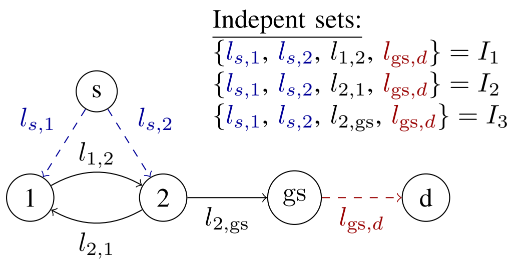

Indepent sets:
$\{l_{s,1}, l_{s,2}, l_{1,2}, l_{\mathrm{gs},d}\} = I_1$
$\{l_{s,1}, l_{s,2}, l_{2,1}, l_{\mathrm{gs},d}\} = I_2$
$\{l_{s,1}, l_{s,2}, l_{2,\mathrm{gs}}, l_{\mathrm{gs},d}\} = I_3$
s
$l_{s,1}$
$l_{s,2}$
$l_{1,2}$
1
2
gs
d
$l_{2,\mathrm{gs}}$
$l_{\mathrm{gs},d}$
$l_{2,1}$

Fig. 4: Example network. Nodes 1 and 2 transmit data to node gs. Nodes s and d are added to model data generation and data collection.

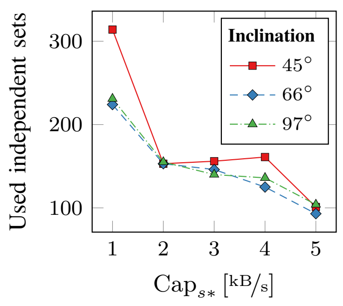


Fig. 5: Number of independent sets that are utilized in the schedule in the solution of LP1.

transmission slot is set to 4 s (according to the value of λ_1). Because the data generation rate is equal to 3 kB/s, only 12 kB instead of 20 kB (the value of $f_{1,2}$) will be available for transmission. A similar problem arises, if the second node transmits first.

The reason for this discrepancy lies in the fact that the linear program does not put transmissions into any chronological order. Instead, the values of the flow variables are restricted only by capacity and flow conservation constraints. For this reason, the LP1 formulation represents a best case bound where all data which needs to be forwarded by a particular node has already reached this node by the time the corresponding link is scheduled for transmission. This example shows that it is not always possible in practice to achieve the exact value calculated based on LP1. However, one can get very close to the maximum by scheduling the independent sets in the correct order and choosing the optimal lengths for transmission slots.

In theory, this could be mitigated by keeping the transmission slots very short. However, this quickly comes to limits in practice due to a lack of synchronization between the nodes and because of technical limitations in real communication systems. Moreover, for mobile networks—like our satellite use-case—additional factors such as varying topology and potentially short contact periods should be taken into account. Thus, the need of a better solution becomes apparent. In this work, we incorporate time dependencies into the aforementioned linear program. This allows to compute the optimal scheduling order of independent sets as well as the required lengths of transmission slots as part of the linear program.

In order to discuss the independent sets' scheduling with respect to temporal order we assume that the independent sets are arranged in some arbitrary but fixed order $\mathcal{I} = (I_1, \dots, I_K)$ and further assume that I_i is scheduled for transmission before I_j if $i < j$. For each link (i, j), consider the ordered list K_{ij} of indices corresponding to all independent sets containing (i, j). Further, we define a variable f_{ij}^k for $k \in K_{ij}$ denoting the amount of data sent via the link l_{ij} during the transmission period of the independent set I_k. The value of the f_{ij} variables in LP1 can then be calculated by summing up all f_{ij}^k variables:

$$f_{ij} = \sum_{k \in K_{ij}} f_{ij}^k.$$

Therefore, the reformulation of the linear program from the previous section with the help of f_{ij}^k variables is straightforward.

With the help of f_{ij}^k it is possible to ensure that all data scheduled for transmission in a particular independent set will be available at the source node in time. The corresponding constraint can be formulated as follows:

$$\forall i \in N_C \setminus \{s\}\, \forall j \in N_C^+(i)\, \forall k \in K_{ij}:$$
$$f_{ij}^k \leq \sum_{m \in N_C^-(i)} \sum_{\substack{k' \in K_{mi} \\ k' \leq k}} f_{mi}^{k'} - \sum_{\substack{m \in N_C^+(i) \\ m \neq j}} \sum_{\substack{k' \in K_{im} \\ k' \leq k}} f_{im}^{k'} - \sum_{\substack{k' \in K_{ij} \\ k' < k}} f_{ij}^{k'} \qquad (2)$$

This constraint ascertains that the amount of data, which can be transmitted over a link when a particular independent set is scheduled, cannot exceed the total amount of data received by the node so far minus the total amount of data already transmitted by the node. The LP1 formulation plus this additional constraint is referred as LP2. The LP2 solution corresponding to the example shown in Fig. 4 is presented in Tab. I.

One last limitation remains to be addressed and overcome, though. In contrast to LP1, the scheduling order of the independent sets is now determined by $\mathcal{I}$, i. e., is itself an input to the linear program LP2 and thus can so far not be optimized. For this example the node 1 was chosen to transmit first. This explains the lower values of the variables $f_{s,1}$, $f_{1,2}$ and $f_{2,\mathrm{gs}}$, compared to the results of LP1. LP2 takes into account that the buffer of node 1 is empty and that the data is transmitted from node 1 to node 2 in one slot, which lasts 5 s. Therefore, only 15 kB can be transmitted within this slot.

Clearly, LP2 does not yet fully achieve the goals formulated above. This results from two unintended limitations in LP2's formulation. First, the order of the independent sets is chosen arbitrary. Second, it is assumed that each independent set can be scheduled only once. It puts certain restrictions on the length of transmission slots.

In order to overcome these limitations, we add additional copies of the independent sets to the linear program. The list $\mathcal{I}$ of all MISs is replaced by $\mathcal{I}' = N \times \mathcal{I}$, i. e., by a concatenation of N copies of $\mathcal{I}$. This effectively turns the corresponding connectivity graph into a multigraph, where each pair of connected nodes is now connected by N copies of the link, yielding the possibility to change the effective order of MISs to compute a better solution.

Effectively, the original independent sets can be utilized in an order differing from $\mathcal{I}$ in the process of solving the linear program. The flow variables of the "misplaced" independent sets will be set to zero in the resulting solution. Furthermore, shorter transmission slots are now possible, allowing for better utiliziation of link capacities. Thus, both the order of independent sets and the lengths of transmission slots can now be computed as a part of the linear program solution. The graph from Fig. 3 shows that, as more copies of $\mathcal{I}$ are added to the linear program, the total throughput converges quickly towards the optimal, ordering-agnostic value computed with the help of LP1.

The proposed approach achieves an optimal solution by adding redundancy. This inevitably leads to an increase of the size of the linear program and hence limits scalability. In order to make this approach practically applicable, the number of independent sets should be kept as small as possible. This can be achieved by solving LP1 first. If the value of some flow variable is equal to zero, the corresponding link can be safely removed from the connectivity graph. The evaluation results in Sec. VII show that the linear program size can be reduced considerably. The overall procedure is as follows:

1) Compute LP1 and remove from the connectivity graph all the links that are not used in the solution.
2) Compute the optimal solution with LP2 by adding more copies of the independent sets until the total throughput reaches the maximal value calculated by solving LP1.

VI. Sliding Window approach

The main drawback of the time-expanded graph is its increased size which in turn causes an increase of the corresponding linear program's size. This limits the scalability of the proposed approach. A naïve solution would be to split the time-expanded graph into a number of blocks and to compute them separately. However, this method ignores buffer links between consecutive blocks and therefore results in a significant decrease in throughput. To mitigate this effect, we propose a sliding-window-based method.

The key idea is to solve a series of linear programs for overlapping time intervals and to combine the results into one overall solution.

This iterative algorithm (for a schematic visualization see Fig. 6) is parametrized by a window size W and a step size S, both in units of time frames and satisfying $W > S$. For ease of introduction, we will here explain it on the example of LP1. As this method is unaffected by the duplication of independent sets it can as well be applied to LP2. For initialization (iteration 0) the linear program corresponding to

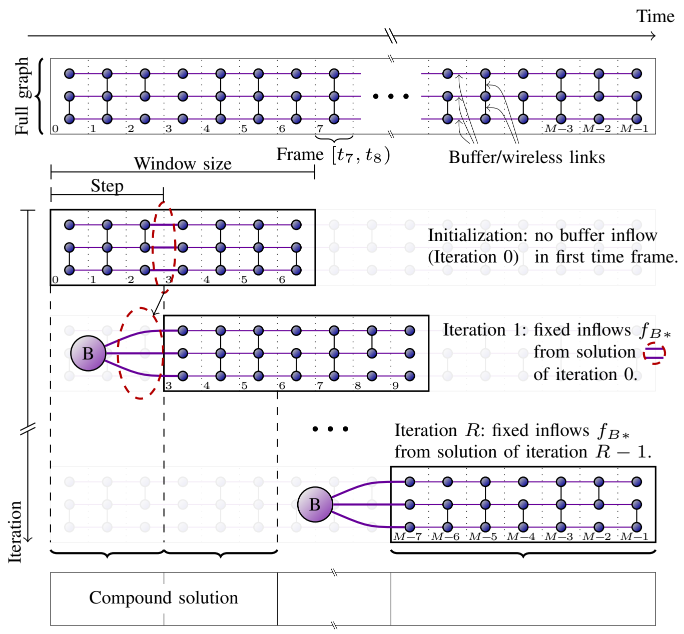

Time
Full graph
Frame $[t_7, t_8)$
Buffer/wireless links
Window size
Step
Initialization: no buffer inflow (Iteration 0) in first time frame.
Iteration 1: fixed inflows f_{B*} from solution of iteration 0.
Iteration R: fixed inflows f_{B*} from solution of iteration $R-1$.
B
Iteration
Compound solution

Fig. 6: The sliding window approach.

the window $[t_0, t_W)$ is solved. Then, $R = \lceil \frac{M-W}{S} \rceil$ iterations are performed, where M is the total number of time frames. In the i^{th} iteration, $i \in \{1, 2, \ldots, R\}$, the linear program for $[t_{iS}, t_{iS+W})$ is solved, not excluding the buffer links at time t_{iS}, but representing them as links of *fixed flow* from a virtual buffered-data-node B to each node in the frame $[t_{iS}, t_{iS+1})$. The corresponding fixed flow values are taken from the result of iteration $i-1$ (compare Fig. 6, dashed ellipses). In the final R^{th} iteration the (possibly shorter) window $[t_{RS}, t_M)$ is considered.

The overall solution is composed as follows: from the solution of each iteration $i = 0, \ldots, R-1$, the first S frames are used. The final R^{th} iteration's solution is used completely.

The proposed method can effectively cope with both memory and computation speed issues. The parameter W limits the size of the linear programs and can be chosen to address memory restrictions. The parameter S controls the total number of linear programs that are computed and, thus, affects the total computation speed. It is also important to note that results are produced beginning with solutions for early time frames, which will *not* be changed during subsequent iterations. For this reason it is possible to make use of each linear optimization result as soon as it is available—without the need to wait for the completion of the overall computation.

VII. Evaluation

In this section, the approach presented above is applied to a LEO satellite network. We assume a satellite system consisting of 18 satellites equally distributed on six orbital planes, resulting in a so-called $45° : 18/6/0$ Walker constellation, although other orbital inclinations are considered as well. By this orbit configuration and a typical altitude for LEO satellites of $600\,\mathrm{km}$ a high global coverage is achieved.

Due to this high global coverage, Walker constellations are widely used in earth observation missions, e. g. in the Fire Observation Constellation [27], or for positioning services such as Galileo [28] or NAVSTAR/GPS [29]. We assume data to be transmitted to four ground stations located in Berlin, Rio de Janeiro, Tokyo and Würzburg. To enable communication in absence of a direct connection with a ground station, all satellites serve as relays for each other.

Assuming a transmission range $R = 5662\,\mathrm{km}$ which corresponds to the maximum range at which the line of sight between two satellites is not obstructed by the Earth, an inter-satellite link (i, j) of the connectivity graph exists if $\|\vec{x}_i - \vec{x}_j\| \leq R$, where $\vec{x}_n$, $n \in N_C$ denotes the three-dimensional Euclidean coordinates of node n. Connectivity graph edges for satellite-to-ground links are defined analogously with a transmission radius of $R/2$. Two links (i_1, j_1) and (i_2, j_2) are connected with a conflict graph edge, if $\{i_1, j_1\} \cap \{i_2, j_2\} \neq \emptyset$ (primary interference) or $(j_1, i_2) \in L_C$ or $(j_2, i_1) \in L_C$ (secondary interference). This means that we assume secondary interference to occur if the receiving node of one link is within the transmission range of the other link's sender.

As described in Sec. III, we distinguish between four types of links: the outgoing links of s (added to model data generation process), the links between the copies of the same node (model buffering), the links between ground stations and d, and the rest. The capacities of the s-links, denoted Cap_{s*}, were considered equal and were varied between 1 kB/s and 5 kB/s. The resulting constant bit rate source traffic mirrors the data generation characteristics of missions with high and fixed measurement frequencies, as, for example, the magnetic field measurements generated by the SWARM mission [30]. The capacities of all other links were set to 5 kB/s. If not stated differently, the results presented in this section refer to $\mathrm{Cap}_{s*} = 1\,\mathrm{kB/s}$, unlimited buffer size, and $45°$ orbital inclination. All results presented below are calculated for one full orbit, using Gurobi Solver [31] v.7.0.2 on one core of a Intel-Xeon E7-4880 CPU using not more than 2.6 GiB of RAM. 95% confidence intervals are provided for all non-deterministic values.

The number of independent sets is one of the key factors that influence the size of the linear program. The total number of the independent sets for the scenario described above is equal to 1015650. The corresponding LP1 linear program is represented by a 7588×1023119 matrix that could be constructed and solved within $(348.6 \pm 4.6)\,\mathrm{s}$. The linear program LP2 that includes the additional constraints (2) is much larger and exceeds the 1000 GiB memory available in our evaluation setup.

In order to make LP2 feasible, we make use of LP1 to significantly reduce the number of independent sets to be considered. The number of independent sets whose λ variables are not equal to zero in the solution for LP1 is shown in Fig. 5 for different values of Cap_{s*} and orbital inclination. For $45°$ this amount ranges from 314 to 101 out of $\approx 10^6$ MISs in total, i. e., only a small fraction of all MISs is utilized in the optimal solution. To exploit this sparsity, we use LP1 as a preprocessing step of LP2 to compute which independent sets contribute to the solution: $\hat{\mathcal{I}} = (I_k \in \mathcal{I} \colon \lambda_k > 0)$. Then we construct the LP2 linear program using only $\mathcal{I}' = N \times \hat{\mathcal{I}}$ as candidate independent sets. The resulting LP2 matrix has the size of 44883×27055.

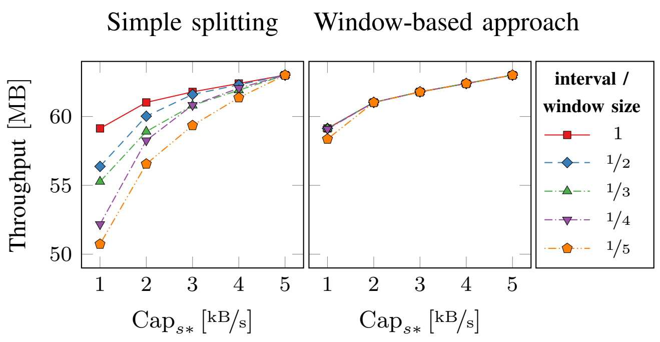


Fig. 7: Maximal throughput achieved by splitting the linear program into parts.

Another important influence factor is the number of copies of independent sets necessary to reach the optimal throughput when computing LP2. In our evaluations, no more than nine additional copies are required for our scenario to get within 0.01% of the optimal throughput.

Our results also show that the presented approach does not cause a significant computational overhead compared to LP1. While $(348.6 \pm 4.6)\,\mathrm{s}$ are required to compute LP1, the total computation time for LP2 is only $(82.9 \pm 0.3)\,\mathrm{s}$. Note that both computation time and the number of additional independent sets required for LP2 depend on the difference between Cap_{s*} and link capacities. Thus, in our case it is possible to achieve even better results if a higher value of Cap_{s*} is assumed.

The presented results show that our approach can be applied for real-world scenarios. A straightforward approach to further improve scalability would be to split the time-expanded graph into a number of intervals and to compute them separately. The main drawback here is a decrease in throughput. To mitigate this effect, the sliding window approach described in Sec. VI is used. The results of applying these two approaches can be seen in Fig. 7.

It becomes clear that splitting the problem into parts can lead to a considerable loss depending on the amount of data available for transmission and the size of the intervals. The graph also shows that these are not the only factors influencing the throughput. For example, splitting the program into five intervals sometimes leads to better results compared to splitting the program into four intervals. This leads us to the assumption that the choice of splitting points may constitute an additional influencing factor. For instance, if such a point occurs before a period with more connectivity, it may happen that the data available for transmission does not suffice to utilize the links.

Compared with splitting the program into parts, the loss in throughput which results from using the slide window approach is much lower. This improvement is, however, achieved at the expense of a considerably higher computation load. For large programs it would therefore be reasonable to combine both approaches.

VIII. Conclusion

In this paper we have presented a new linear programming formulation for throughput maximization in satellite networks. Unlike other solutions based on calculating independent link sets, our formulation allows to compute a complete transmission schedule directly. In particular, the optimal transmission order is determined as a part of the linear program solution instead of relying on heuristics.

Our first formulation, LP1, is the straightforward adaption of a linear programming formulation for computing throughput-optimal joint routing and scheduling in static wireless networks to the mobile use case of a LEO satellite constellation. Our second approach, LP2, extends LP1 to incorporate link scheduling and transmission slot length's, thereby providing a complete transmission schedule in a mobile setting. In our evaluation we have shown that using LP1 as preprocessing step, LP2 can be applied well to real-world scenarios. Moreover, the computation of scheduling order requires only very limited additional computation time.

To compensate for the inevitable increase in the size of the linear program, which results from node mobility, we developed a novel scaling window-based approach. This approach allows to find a good trade off between optimality and computation complexity. We showed that our solution provides significantly better results then a simpler method, where the time-expanded graph is split in a number of intervals which are considered separately.

In future work we plan to consider further real-world challenges like a more realistic transmission and interference model, sophisticated in-orbit dissemination of transmission schedules, as well as finite buffer sizes.

Acknowledgment

This work was supported by the German Research Foundation (DFG) under grant numbers SCHI 327/29-1 and SCHE 1649/7-1.

References

[1] S. Bandyopadhyay *et al.*, "A Review of Impending Small Satellite Formation Flying Missions," in *AIAA-ASM '15*, Jan. 2015.

[2] M. D'Errico, *Distributed Space Missions for Earth System Monitoring*. Springer Science & Business Media, 2012, vol. 31.

[3] K. Schilling, "Perspectives for Miniaturized, Distributed, Networked Cooperating Systems for Space Exploration," *Robotics and Autonomous Systems*, vol. 90, pp. 118–124, Apr. 2017.

[4] T. H. Zurbuchen *et al.*, "Performing High-Quality Science on CubeSats," *Space Research Today*, vol. 196, pp. 11–30, 2016.

[5] K. Schilling, "Networked Distributed Pico-Satellite Systems for Earth Observation and Telecommunication Applications," in *AGNFCS '09*, Jul. 2009.

[6] J. Alvarez and B. Walls, "Constellations, clusters, and communication technology: Expanding small satellite access to space," in *2016 IEEE Aerospace Conference*, pp. 1–11.

[7] A. Capone *et al.*, "Routing, scheduling and channel assignment in Wireless Mesh Networks: Optimization models and algorithms," *Ad Hoc Networks*, vol. 8, no. 6, pp. 545–563, 2010.

[8] J. A. Fraire, P. G. Madoery, and J. M. Finochietto, "Traffic-Aware Contact Plan Design for Disruption-Tolerant Space Sensor Networks," *Ad Hoc Networks*, vol. 47, no. Supplement C, pp. 41–52, Sep. 2016.

[9] D. Zhou *et al.*, "Toward High Throughput Contact Plan Design in Resource-Limited Small Satellite Networks," in *PIMRC '16*, Sep. 2016, pp. 1–6.

[10] K. Jain *et al.*, "Impact of Interference on Multi-hop Wireless Network Performance," *Wireless Networks*, vol. 11, no. 4, pp. 471–487, Jul. 2005.

[11] P. Björklund, P. Värbrand, and D. Yuan, "A column generation method for spatial TDMA scheduling in ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 4, pp. 405–418, Oct. 2004.

[12] B. Kotnyek, "An annotated overview of dynamic network flows," *INRIA*, no. RR-4936, Sep. 2003.

[13] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," in *SIGCOMM '04*, Sep. 2004, pp. 145–158.

[14] A. Ferreira, A. Goldman, and J. Monteiro, "Performance evaluation of routing protocols for MANETs with known connectivity patterns using evolving graphs," *Wireless Networks*, vol. 16, no. 3, pp. 627–640, Apr. 2010.

[15] G. Araniti *et al.*, "Contact graph routing in DTN space networks: overview, enhancements and performance," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38–46, Mar. 2015.

[16] M. Kodialam and T. Nandagopal, "Characterizing Achievable Rates in Multi-hop Wireless Networks: The Joint Routing and Scheduling Problem," in *MobiCom '03*, Sep. 2003, pp. 42–54.

[17] J. Zhang *et al.*, "Joint Routing and Scheduling in Multi-Radio Multi-Channel Multi-Hop Wireless Networks," in *BROADNETS '05*, Oct. 2005, pp. 631–640 Vol. 1.

[18] Y. Wang *et al.*, "Interference-Aware Joint Routing and TDMA Link Scheduling for Static Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 12, pp. 1709–1726, Dec. 2008.

[19] M. Johansson and L. Xiao, "Cross-layer optimization of wireless networks using nonlinear column generation," *IEEE Transactions on Wireless Communications*, vol. 5, no. 2, pp. 435–445, Feb. 2006.

[20] G. Konidaris, S. Toumpis, and S. Gitzenis, "Primal decomposition and online algorithms for flow optimization in wireless DTNs," in *GLOBECOM '13*, Dec. 2013, pp. 84–90.

[21] F. Dürr and N. G. Nayak, "No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN)," in *RTNS '16*, Oct. 2016, pp. 203–212.

[22] E. Schweissguth *et al.*, "ILP-based Joint Routing and Scheduling for Time-triggered Networks," in *RTNS '17*, Oct. 2017, pp. 8–17.

[23] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental Flow Scheduling Routing in Time-sensitive Software-defined Networks," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2017, IEEE Early Access.

[24] G. Bonin *et al.*, "CanX–4 and CanX–5 Precision Formation Flight," Aug. 2015.

[25] J. A. Fraire *et al.*, "Assessing contact graph routing performance and reliability in distributed satellite constellations," *Journal Comp. Netw. and Communic.*, vol. 2017, pp. 2 830 542:1–2 830 542:18, 2017.

[26] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[27] C. Bonillo *et al.*, "Mission Management for Constellations and Highly Manoeuvrable Satellites," in *Spacecraft Guidance, Navigation and Control Systems*, ser. ESA Special Publication, B. Schürmann, Ed., vol. 425, Feb. 2000, p. 219.

[28] Á. Mozo-García *et al.*, "Galileo constellation design," *GPS Solutions*, vol. 4, no. 4, Apr 2001.

[29] M. G. Matossian, "Improved candidate generation and coverage analysis methods for design optimization of symmetric multisatellite constellations," *Acta Astronautica*, vol. 40, no. 2, pp. 561 – 571, 1997.

[30] E. Friis-Christensen *et al.*, "Swarm – an earth observation mission investigating geospace," *Advances in Space Research*, vol. 41, no. 1, pp. 210 – 216, 2008.

[31] Gurobi Optimization, Inc. (2016) Gurobi optimizer reference manual. [Online]. Available: http://www.gurobi.com