Applied Soft Computing 52 (2017) 446–456

Contents lists available at ScienceDirect

Applied Soft Computing

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

An evolutionary approach towards contact plan design for disruption-tolerant satellite networks

Juan A. Fraire [a,*], Pablo G. Madoery [a], Jorge M. Finochietto [a], Guillermo Leguizamón [b]

[a] Universidad Nacional de Córdoba – CONICET, Córdoba, Argentina
[b] Universidad Nacional de San Luis, San Luis, Argentina

ARTICLE INFO

Article history:
Received 5 April 2016
Received in revised form 15 October 2016
Accepted 17 October 2016
Available online 22 October 2016

Keywords:
Delay tolerant networks
Contact plan design
Evolutionary algorithms

ABSTRACT

Delay and disruption tolerant networks (DTNs) are becoming an appealing solution for satellite networks where nodes can temporarily store and carry in-transit data until a link with a suitable next-hop becomes available. Since satellite trajectories and orientation can be predicted, on-board routing schemes can base these forwarding decisions on a contact plan comprising all forthcoming communication opportunities. In general, contact plans are previously calculated on ground where their design can be optimized to consider not only available spacecraft resources but also the expected traffic which is largely foreseeable in space applications. Despite optimal contact plan design procedures exist, their computation complexity might result prohibitive even for medium-sized satellite networks. In this work, we propose an evolutionary algorithm to provide sub-optimal yet efficient and implementable contact plans in bounded time. In particular, we depict specific strategies such as encoding and repairing techniques to later evaluate the algorithm performance in a typical scenario demonstrating its usefulness for planning future DTN-based satellite networks.

1. Introduction

Today, optical and radar images are acquired continuously from orbit as they enable better understanding and improved management of the Earth and its environment. In particular, orbiting networks of distributed satellite sensors are emerging as a mean to extend Earth observation missions revisit time and ground coverage [1]. In order to optimize space-to-ground data delivery in satellite networks, nodes can cooperatively pass sensed data among them before establishing the final communication with the receiving ground station. To this end, satellites shall rely on efficient network protocols and algorithms properly designed to operate over inter-satellite links (ISLs) [2].

In the Internet, protocol operations are largely based on instant flow of information between sending and receiver nodes. However, in a space flight mission environment, the highly varying communication ranges on which mobile nodes have to operate, compels to face several disruptive situations [3]. For example, Fig. 1 illustrates a network of satellites with polar orbits where connectivity among them can only be guaranteed over the pole as ISL distances are minimal. Previous work has proposed to constrain ISL distances by applying strict flight-formations [4]; but later analysis showed that disruptions could be handled by considering delay and disruption tolerant networking (DTN) architecture [5].

Originally proposed in [6], DTN protocols assumes no continuous connectivity throughout the network. In particular, lapses in wireless links may be routine, lengthy, and recurring and should not be interpreted as errors or unwanted changes in topology. Indeed, the interval of time during which data may be passed from one node to another over a link is defined as a *contact*. However, the impermanent nature of contacts compels the nodes to have local storage for temporary retention of data which cannot be forwarded immediately. As a result, information can flow between nodes through contacts in a *Store-and-Forward* fashion until reaching its final destination. This forwarding scheme is illustrated in Fig. 2, where node 1 sends data to node 3. Since there is no direct communication between the source and destination nodes, data needs to go through intermediate node 2. However, a sporadic link exists between nodes 2 and 3, which requires to store in-transit data on the local storage of node 2 until the contact with node 3 is available.

In general, episodes of communication between satellites and ground stations on Earth are typically scheduled weeks or months before they occur. Specifically, the beginning and end of each contact can be accurately computed from known orbital elements on ground [7]. As a result, all forthcoming network communication opportunities, including ISLs, can be imprinted in a *contact plan* (CP) which can be conveniently provisioned to DTN nodes in advance [8]. Then, as traffic flows in the network, on-board routing schemes

* Corresponding author.
E-mail address: juan.fraire@unc.edu.ar (J.A. Fraire).

http://dx.doi.org/10.1016/j.asoc.2016.10.023
1568-4946/

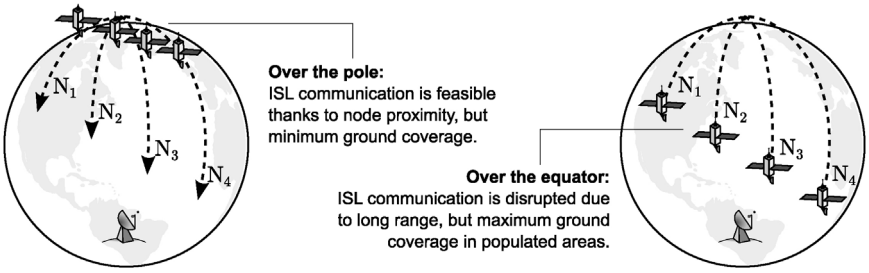

Fig. 1. Orbiting network of 4 polar orbit satellites.

can take advantage of this topological information to take efficient forwarding decisions.

Recent research has suggested to further adapt and optimize CPs in accordance with available on-board transceivers on each node [9]. This process became known as *contact plan design* (CPD) and has received increasing attention from the community as different criteria can significantly impact the final network performance [10–15]. Among existing schemes, the Traffic-Aware Contact Plan (TACP) [15] was proposed in 2016 as a suitable scheme for satellite networks as it considers all possible parameters including the expected amount of traffic and its generation time. Indeed, data acquisitions from on-board instruments are also centrally scheduled by a mission control center on ground [16].

Nonetheless, TACP operating performance relies on a mixed integer linear programming (MILP) formulation whose computation complexity becomes intractable even for medium-sized networks. The latter becomes a critical issue in DTN satellite networks where CPs have to be timely provisioned to orbiting nodes. In order to guarantee bounded duration in the CPD cycle, this article contributes with an heuristic alternative to TACP based on evolutionary algorithms (TACP-EA). Even though a similar algorithm based on simulated annealing have recently been proposed for the CPD problem [14], it assumes as optimization criteria tailored for navigation constellations with unpredictable traffic. To the best of authors knowledge, this is the first time an heuristic approach is proposed for the CPD of Earth observation satellite networks with scheduled traffic.

This work is an archival quality version of the article [17] with a more detailed explanation on the algorithm and an extended and improved performance analysis. The article is structured as follows. In Section 2 we provide a general overview of the CPD problem and the design constraints to later describe TACP and discuss its computational limitations. In order to overcome the latter, we describe TACP-EA as an alternative approach in Section 3. Next, in Section 4 we evaluate the performance of TACP-EA in a Low Earth Orbit (LEO) satellite network example to finally draw the conclusions in Section 5.

2. Contact plan design overview

Colloquially, a contact can be defined as a communication opportunity between DTN nodes. In practice, the information encoded in a contact include source and destination node, start and end time, and expected data rate. Indeed, these parameters can be calculated on ground by means of precise orbital mechanics comprising position, range, and attitude (orientation of the spacecraft in the inertial system) [7]. Also, the resulting values such as transmission power, modulation, bit-error-rate, etc, can be further adjusted based on wireless communication models. Finally, the set of all feasible contacts within a given time *interval* are imprinted in the CP and provisioned to the orbiting DTN nodes. Protocols for distributing CP information has been studied in [18], but they are beyond the scope of this article.

Once on-board, the CP is used by existing distributed routing schemes such as contact graph routing (CGR) [19] to determine the optimal next-hop for a given packet flow. In particular, CGR runs in each DTN node, and creates a *contact graph* based on the CP where a modified Dijkstra's search can be executed. Furthermore, CGR considers the data rate encoded in the contacts in order to avoid overbooking of future links (i.e., congestion) [20]. The interested reader can refer to [19] for an extended analysis on how distributed routing exploits the data encoded in CPs.

Unfortunately, due to limited resources in satellite platforms, e.g., limited available transceivers, not all calculated contacts in a CP can be implemented. For example, a node may have potential contacts with more than one node at a given time but only one of these opportunities can be utilized taking into account one transceiver on-board. As a consequence, only a subset of the feasible contacts can be activated simultaneously in the final CP [9]. The latter selection process is known as CPD, and should be carefully addressed in order to schedule those contacts that resolve possible resource conflicts as described below.

2.1. Design constraints and procedures

Among the resource limitations of traditional satellite architectures, authors have classified and analyzed those concerning wireless communications [9]. For instance, the simplest yet more frequent constraint is evidenced when a spacecraft antenna radiation pattern allows to reach two or more neighbors as shown in Fig. 3(a). However, multiple channel access schemes is generally discouraged in satellite networks due to the impact of the negotiation overhead over long distances [21]. As a result, even though two simultaneous contacts are feasible, only one can be included in the final CP. Next, Fig. 3(b) and (c) illustrates a similar situation where two antennas and two transceivers are available on-board, but only one of them can be enabled at a given time. Also, when considering electronically or mechanically steerable antenna techniques

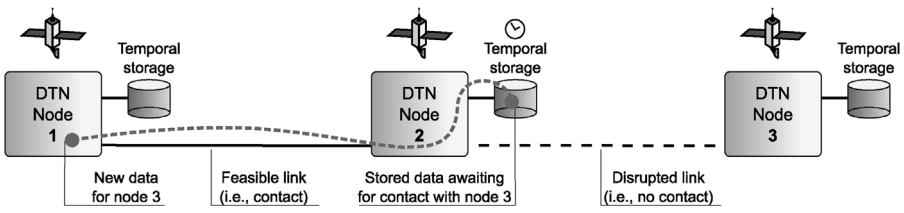

Fig. 2. Store-and-Forward scheme in DTN.

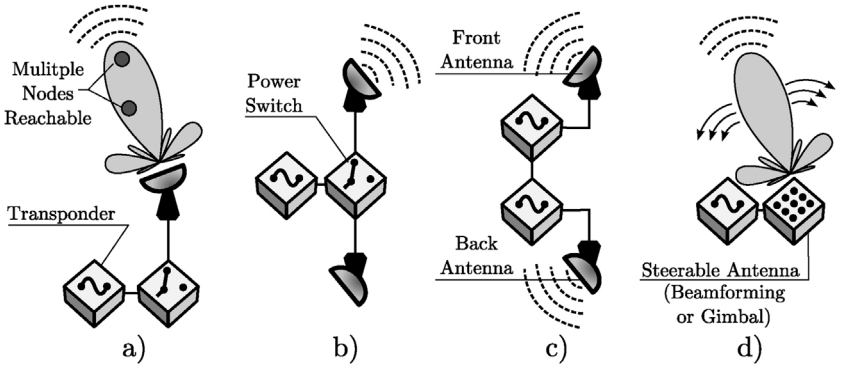

Fig. 3. Satellite architecture with (a) multiple target nodes, (b) a power switch, (c) two transponders, and (d) a steerable antenna.

(i.e., beam-forming or gimbals-based antennas as in Fig. 3(d)), one contact out of many might also need to be chosen in advance.

In general, the highlighted design constraints require of a contact selection and adjustment process such that the implemented links in node i do not exceed the resources available on each orbiting node (i.e., $links(node_i) \leq maxLinks_i$). Nonetheless, the selection on node i affects next-hop nodes (i.e., all direct neighbors), then second hops, an so forth potentially deriving in a non-trivial combinatorial problem. Indeed, the complexity of the latter exponentially rises as the interval, nodes, and feasible contacts increase.

Due to the impact of the CPD on the overall network performance, recent contributions has suggested automated procedures and models to aid in the planning of DTN-based satellite networks. Indeed, different optimization criteria has been studied to guide the selection of contacts. Among them, single hop fairness is proposed in [12], route delay minimization in [13,14], and traffic delivery time minimization in TACP [15]. Fig. 4 graphically illustrates the relative performance in terms of final traffic delivery time used by these existing CPD procedures. Among them, we study TACP as it is the only scheme that consider scheduled traffic as seen on Earth observation satellite networks.

2.2. TACP overview

The TACP scheme is specifically designed to maximize traffic amount and delivery time by exploiting the a priori knowledge of the satellite network traffic. In particular, since both telemetry (i.e., spacecraft status report) as well as in-orbit acquisitions can be fairly predicted [16], they are considered by this model in the CPD procedure. In other words, the resource selection is mandated by the traffic expected to flow in the network. Also, contacts that will not carry traffic are disabled to improve the resource usability and power consumption. These features of TACP are further described in [15] and captured by a theoretical MILP formulation where the CP is modeled with a finite state machine approach.

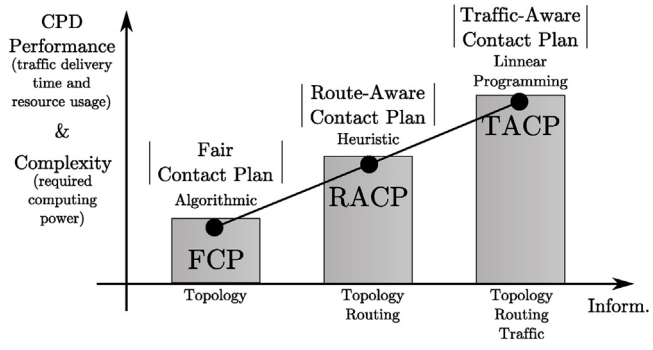

Fig. 4. Contact plan design methodologies information and performance.

Each state in the CP model comprises a graph whose vertices and edges symbolize DTN nodes and their feasible contacts respectively. Specifically, the topology is discretized by a set of k time intervals $[t_k, t_{k+1}]$ or division points. Each state has a graph representing the communication opportunities within its interval duration ($i_k = t_k - t_{k-1} : 1 \leq k \leq K$). As a result, the time evolution is encoded in two matrices $T = \{t_k\}$ and $I = \{i_k\}$ of size K, representing each k state starting time and interval duration respectively. For every start and end of a contact, there is a k_a to k_{a+1} state evolution in the finite state machine. Indeed, a single contact can span multiple consecutive k states. For example, Fig. 5 illustrates the CP for a half-orbit interval of the orbiting network in Fig. 1 and its corresponding finite state machine modeling. The time measurements shown in the figure was calculated by means of realistic satellite propagators based on [7]. Henceforth, an *edge* will represent the portion of a contact during a given k state.

In order to model capacity, each edge between node i and j at state k is parameterized by a capacity value (traffic volume) encoded in $c_{k,i,j}$. However, since DTN implements a store-carry-and-forward scheme, the system capacity is not only related to link transfer (as expressed in $C = \{c_{k,i,j}\}$) but also to the storage capability of each intermediate node. Therefore, in addition, TACP assumes each vertex i has an associated maximum buffer capacity of b_i encoded in a buffer matrix B. As a result, C and B will bound the traffic flow driven by a traffic matrix D (where $D = \{d_k^{i,j}\}$). Each $d_k^{i,j}$ traffic volumes represents data to be generated at the beginning of state k (at time t_{k-1}) at node i with j as final destination. Finally, resource constraints are modeled by $P = \{p_i\}$ whose p_i components encode the maximum quantity of communication links node i can simultaneously implement at a given time.

The resulting traffic flow is therefore affected by coefficients in C, B, D and P while its resulting value is captured by means of $X_{k,i,j}^{y,z}$ variables. Each $X_{k,i,j}^{y,z}$ represents the transmission with source y and destination z flowing through node i to j at state k. Indeed, the summation of these flows should never exceed the associated edge capacity $c_{k,i,j}$ or provoke a buffer (b_i) overflow in the receiving node i. Besides the traffic flow, TACP also captures the buffer allocation required to store the information in $B_{k,i}^{y,z}$, and the optimal link selection in binary variables $Y_{k,i,j}$. Obtaining the latter is indeed the main objective in the CPD procedure as it defines which contact will finally be included in the CP. Table 1 summarizes the parameters and values considered by TACP. Nonetheless, the interested reader should refer to [15] for an extended explanation of these input coefficients and output variables.

The TACP problem can then be formally expressed by Eqs. (1) to (9). In particular, the objective function (1) aims at minimizing the sum of the products of data units ($X_{k,i,j}^{y,z}$) with the time associated to the k state (t_k) modified by a weighting function $w(t_k)$. Among the constraints, Eq. (2) maps the flow imbalance in each node i to its buffer variation for all states and each (y, z) flow. Eqs.

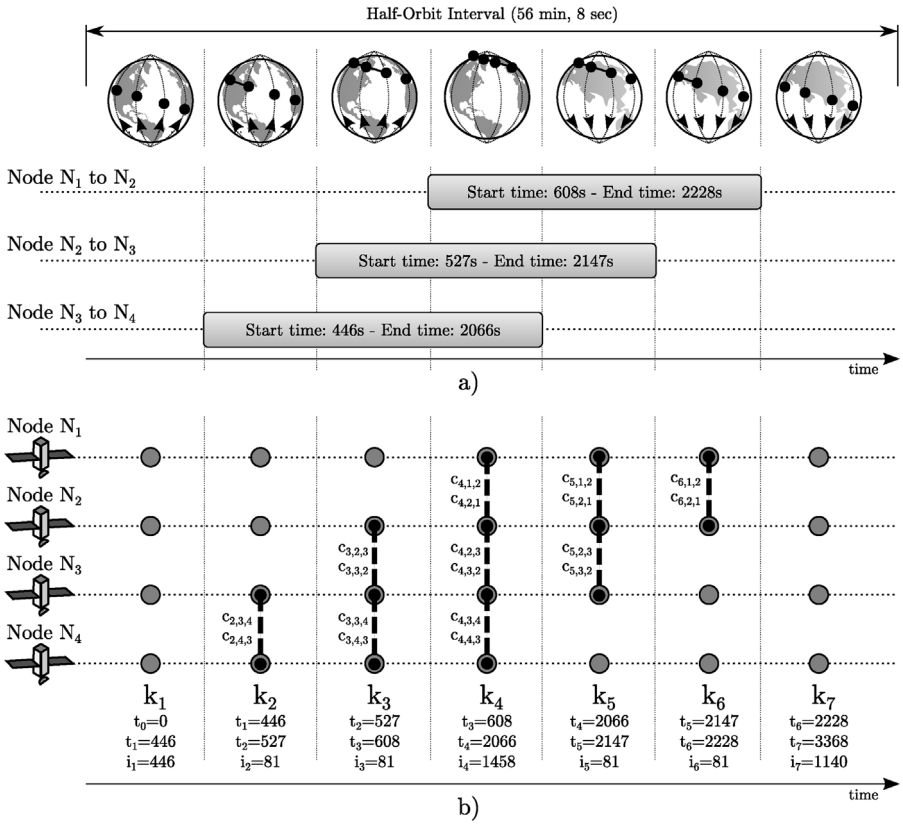

Fig. 5. A contact plan in (a) and its finite state machine modeling in (b).

(3) and (4) impose an upper bound b_i and an initial empty condition $B^{y,z}_{0,i} = 0$ for each node i and y, z flow respectively. Next, the maximum capacity for each edge is enforced in (5) and each source and sink outflow and inflow imbalance is generated by (6) and (7) respectively. Furthermore, Eqs. (8) and (9) models constraints by limiting the maximum quantity of simultaneous links to use in a given node i. One more time, the interested reader can refer to [15] for a complete and detailed description of the TACP formulation.

Once the MILP model is solved, resulting binary variables $Y_{k,i,j}$ dictates whether an edge between node i and j should be enabled at state k in the resulting CP. Furthermore, if all flow variables $X^{y,z}_{k,i,j}$ over a given i–j edge are 0, this contact can be disabled as it is not expected to carry data. Indeed, this allows for an efficient power management of the system from the CPD procedure perspective.

Table 1
TACP model parameters.

Input coefficients	
N	Nodes quantity
K	Topology states quantity
$T = \{t_k\}$	State k start time ($1 \le k \le K$)
$I = \{i_k\}$	State k interval duration ($i_k = t_k - t_{k-1} : 1 \le k \le K$)
$C = \{c_{k,i,j}\}$	Capacity of i to j contact at state k ($1 \le k \le K$ and $1 \le i, j \le N$)
$B = \{b_i\}$	Node i buffer capacity ($1 \le i \le N$)
$D = \{d^{i,j}_k\}$	Traffic from i to j originated at the beginning of k ($1 \le k \le K$ and $1 \le i, j \le N$)
$P = \{p_i\}$	Number of simultaneous links in node i ($1 \le i \le N$)
M	Big "M" coefficient for link equations
Output variables	
$\{X^{y,z}_{k,i,j}\}$	Traffic from y to z at state k in i to j edge ($1 \le i, j, y, z \le N$)
$\{B^{y,z}_{k,i}\}$	Node i buffer occupancy at the end of k by traffic from y to z ($1 \le i, y, z \le N$)
$\{Y_{k,i,j}\}$	Interface selection from i to j at state k ($1 \le k \le K$ and $1 \le i, j \le N$)

After provisioned to the DTN nodes, the resulting CP can be used by on-board routing schemes to take forwarding decisions based on this optimized view of the forthcoming network topology.

$$\text{minimize}: \quad \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{y=1}^{N}\sum_{z=1}^{N} w(t_k) * X^{y,z}_{k,i,j} \tag{1}$$

Subject to:

$$\sum_{j=1}^{N} X^{y,z}_{k,j,i} - \sum_{j=1}^{N} X^{y,z}_{k,i,j} = B^{y,z}_{k,i} - (B^{y,z}_{k-1,i} + d^{i,z}_k) \quad \forall k, i, y, z \tag{2}$$

$$B^{y,z}_{k,i} \le b_i \quad \forall k, i, y, z \tag{3}$$

$$B^{y,z}_{0,i} = 0 \quad \forall i, y, z \tag{4}$$

$$\sum_{y=1}^{N}\sum_{z=1}^{N} X^{y,z}_{k,i,j} \le c_{k,i,j} \quad \forall k, i, j \tag{5}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{N} X^{y,z}_{k,i,j} = \sum_{k=1}^{K} d^{i,j}_k \quad \forall i = y, z \tag{6}$$

$$\sum_{k=1}^{K}\sum_{i=1}^{N} X^{y,z}_{k,i,j} = \sum_{k=1}^{K} d^{i,j}_k \quad \forall y, j = z \tag{7}$$

$$\sum_{j=1}^{N} Y_{k,i,j} \le p_i \quad \forall i, k \tag{8}$$

$$\sum_{j=1}^{N}\sum_{y=1}^{N}\sum_{z=1}^{N} X^{y,z}_{k,i,j} \le M * Y_{k,i,j} \quad \forall i, k \tag{9}$$

Despite the optimal results provided by TACP, its MILP formulation is based on a time evolving extension of the multi-commodity flow problem which is known to be strongly NP-complete [22]. This implies that the time required to solve the problem using any currently existing MILP solver increases very quickly as the size of the problem grows. Indeed, if the time required to design an arbitrarily complex CP cannot be bounded, a satellite network control center will not be able to guarantee a timely provision of this critical configuration element. Indeed, in Section 4, we show that TACP applicability for a case study of satellite network results questionable.

As a result, in Section 3 we provide an heuristic approach specifically tuned to solve this problem in bounded time-frames while still providing valuable and efficient CPs. To the best of authors knowledge, this is the first heuristic proposal to solve the traffic-aware CPD problem for satellite networks. Previous contributions such as [14] proposed simulated annealing alternatives to CPD for navigation systems. However, since the design criteria in the latter does not consider traffic, the resulting CP differs from the expected in traffic-aware schemes. Therefore, it is not possible to perform a comparison of their computing performance.

3. Genetic algorithm optimization

When using the MILP theoretical model of TACP to solve small DTN systems, an optimal solution might be obtained in reasonable time frame with traditional *cutting-plane* mechanisms available in free solvers such as GNU Linear Programming Kit (GLPK) [23]. Nevertheless, when considering a sufficiently large instance of the problem, these methods fail to deliver an output in reasonable time as the required computing power increases exponentially with the size of the decision variables (satellites, antennas, etc.). For these cases, approximate methodologies such as Population-based metaheuristics has previously been studied in the literature [24]. Among the latter, evolutionary algorithms (EAs) are one of the most studied for complex problems. In this section we consider a EA in order to tackle the CPD problem with TACP criteria for large-sized DTN systems.

In general, EAs are based on the notion of competition, mimicking the evolution of species [25]. Our EA proposal for the traffic-aware CPD (TACP-EA) formulation is illustrated in Fig. 6, specified in Algorithm 1 and described throughout this section. In contrast with the MILP formulation of TACP which directly delivers the optimal CP, TACP-EA is designed to intelligently generate multiple CPs to evaluate which of them provides the best traffic metric. This process is iterated several times with aim of improving the quality of best CP in the group. The main advantage of TACP-EA against TACP is that the CPD process can be time-bounded by restricting the quantity of iterations. As previously discussed, this is a desired feature in satellite networks where CPs must be timely delivered to orbiting nodes.

To initialize TACP-EA, the initial *population* (P) of CPs is randomly generated and then repaired (lines 1–5, Algorithm 1) resulting in a set of *individuals* (S), each one encoding a feasible solution of the problem (a CP). The length of the encoding used for each CP is previously calculated and stored in E_{size}. This is further analyzed and described in Section 3.1. For each S an *objective function* defines its *fitness* (evaluated in lines 9–11, Algorithm 1) which in turn determines its survivability chances ($probSel[S]$) in the system environment (lines 12-14, Algorithm 1). Among them, and at each iteration (*iter*) step, a set of *parents* ($P_{parents}$) are selected and subjected to *crossover* or *mutation* modifications (lines 15–19, Algorithm 1). This is detailed in Section 3.2. Finally, as explained in Sections 3.3 and 3.4, the resulting *offspring* of this operations are eventually *repaired* and included in the population by displacing previous generations guided by a specific *replacement strategy* (lines 20-22, Algorithm 1). This procedure is repeated through several iterations improving the solution quality in the population to finally return the best S (line 23, Algorithm 1) standing for the most efficient CP found. In the following sections a detailed description of each stage of the algorithm is provided.

Algorithm 1. TACP evolutionary algorithm (TACP-EA)

input : CP_{input}, $maxLinks_i \quad \forall i$, P_{size}, $iter$, pMt, pCr
output: CP_{output}
1 **for** $\forall i, k \in CP_{input} \mid links(node_i) \geq maxLinks_i$ **do**
2 $E_{size} \leftarrow E_{size} + maxLinks_i - links(node_i)$;
3 **for** $\forall S \in P$ **do**
4 $S \leftarrow$ `random(0,`$2^{E_{size}}$`)` ;
5 $P[S] \leftarrow$ `repair(`$P[S]$`)` ;
6 **for** $iter$ **do**
7 $P_{nextGen}[P_{size}] \leftarrow 0$;
8 $fitness_{total} \leftarrow 0$;
9 **for** $\forall S \in P$ **do**
10 $fitness[S] \leftarrow 1/$`objectiveFunct(`S`)` ;
11 $fitness_{total} \leftarrow fitness_{total} + fitness[S]$;
12 **for** $\forall S \in P$ **do**
13 $probSel[S] \leftarrow fitness[S]/fitness_{total}$;
14 $P_{parents} \leftarrow$ `rouletteSelection(`P,$probSel[S]$`)` ;
15 **for** $\forall S \in P_{parents}$ **do**
16 **if** `random(`$0, 1$`)` $\leq pMt$ **then**
17 $P_{nextGen}[S] \leftarrow$ `mutate(`S`)`;
18 **if** `random(`$0, 1$`)` $\leq pCr$ **then**
19 $P_{nextGen}[S] \leftarrow$ `crossover(`$P_{parents}$`)` ;
20 $P_{nextGen}[S] \leftarrow$ `repair(`$P_{nextGen}[S]$`)` ;
21 **for** $\forall S \in P \mid fitness[S] \rightarrow 0$ **do**
22 $S \leftarrow P_{nextGen}[S]$;
23 `return` $CP_{output} = S \mid fitness[S] \rightarrow max$;

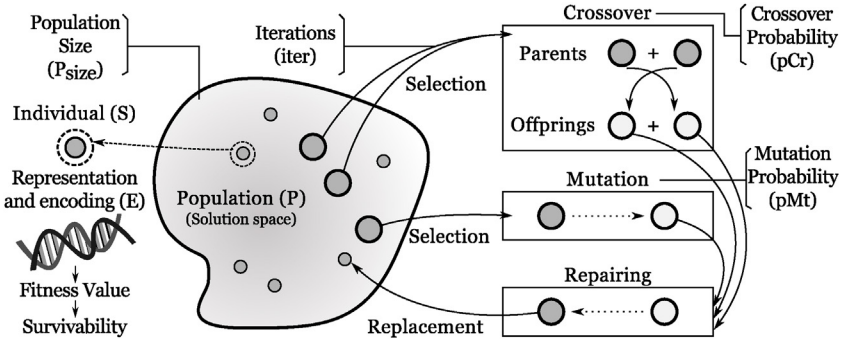

Fig. 6. Evolutionary algorithm procedure.

3.1. Encoding

Defining a logical representation (E) of the solution (individuals) is a critical step towards designing an efficient iterative meta-heuristic. In EA, the *genotype* represents the encoding while the *phenotype* represents the solution. In general, the genotype plays a major role in the efficiency and effectiveness of any meta-heuristic. It must facilitate the determination of the objective function of the solution (a CP), be capable of representing all feasible solutions, and also be easy to manipulate. In TACP-EA, solutions represent implementable CPs that satisfies the resource constraints. As previously stated in Section 2.2, TACP decision variables are the set $\{Y_{k,i,j}\} \mid Y_{k,i,j} = \{0, 1\}$ representing the contact selection from node i to j at state k. In other words, $Y_{k,i,j}$ are binary values that specifies if the contact from i to j is to be enabled ($Y_{k,i,j} = 1$) or not ($Y_{k,i,j} = 0$) at state k. As a result, a set of $E_S = \{Y_{1,1,2}, Y_{1,1,3}, Y_{1,2,1}, \ldots, Y_{k,i,j}\}$ is sufficient to encode an individual S phenotype and therefore, a CP solution (not necessarily feasible at this point) of the stated problem.

In particular, this set of $E_S = \{Y_{k,i,j}\}$ of size E_{size} results a convenient structure that can be encoded in a straightforward fashion as a genotype in a binary encoding scheme. Binary encoding is a classical and popular technique for decision-based evolutionary algorithms (i.e., yes/no problems) where a string of 1s and 0s defines a set of binary variables of a solution [24]. Fig. 7 illustrates the finite state machine modeling with the corresponding encoding in the same topology of Fig. 5. Within the binary encoding, each bit conveniently represents an edge decision in the represented CP and its corresponding genotype and phenotype. In order to reduce the solution space, we are only including $Y_{k,i,j}$ variables that are involved in a contact selection process (filtered in line 1 and 2, Algorithm 1). For instance, in the example of Fig. 7, there is no need to encode variable $Y_{2,3,4}$ since it is not involved in a resource constraint decision making. In other words it can be considered enabled by default without violating any resource restriction. Indeed, these variables can later be disabled if traffic is not expected to flow through them for energy saving purposes. In general, and throughout the TACP-EA iterations, we will decode all chromosomes as they need to be subjected to a routing scheme in order to determine their objective function value (line 10, Algorithm 1).

3.2. Objective function and fitness

In order to evaluate the survivability of the individuals as they evolve, a *fitness* parameter needs to be defined. Indeed, the fitness will depend on the *objective function* value of the solution encoded in the individual [24]. In the TACP-EA formulation, the fitness is expressed as an inverse function of the objective value based on the best delivery time (BDT) of all the expected traffic in matrix D. Indeed, this is analogous to the objective function of the TACP MILP formulation formalized in Eq. (1). Specifically, minimizing all $X^{y,z}_{k,i,j}$ flows weighted by the time associated to the k state (t_k) implies minimizing the traffic BDT [13]. Therefore, in TACP-EA, each CP fitness is computed by $fitness[S] = 1/(\text{BDT})$. As a result, the higher the traffic evacuated and the lower its BDT, the higher the chances the individual S has to survive among those in the population.

In order to determine the BDT metric, the encoded CP needs to be evaluated by a routing and forwarding mechanism so as to determine how the expected traffic would flow if finally provisioned to the satellite network. This step is represented by the `objectiveFunct(S)` call in line 10 of Algorithm 1. At this point, any routing scheme such as CGR that can consider edges capacity $C = \{c_{k,i,j}\}$, buffer capacities $B = \{b_{k,i,j}\}$, and traffic $D = \{d^{i,j}_k\}$ can be

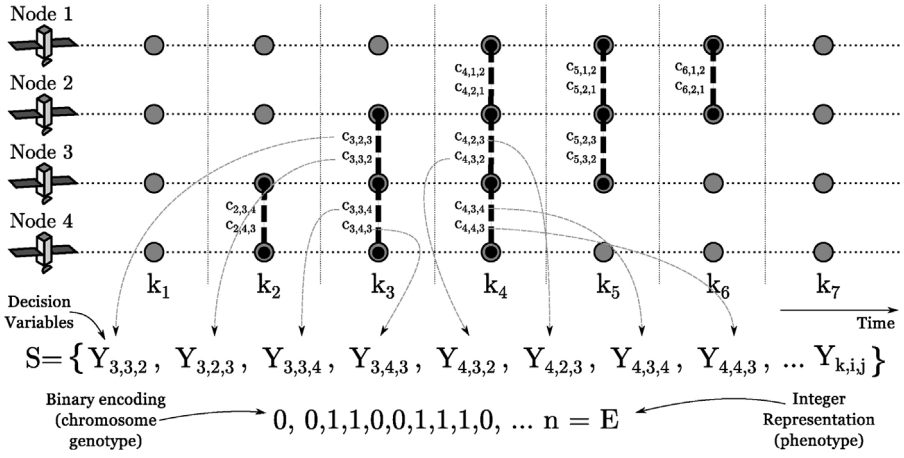

Fig. 7. Contact plan example representation and encoding.

considered. Furthermore, on-line simulation can be considered for a more realistic traffic analysis. Indeed, this flexibility can be highlighted as another advantage of TACP-EA against TACP where the routing calculations are based on a multi-commodity flow problem formulation. In order to be able to compare the results of TACP-EA and TACP, we choose to use the same routing criteria. In particular, this is based on a continuous linear programming model derived from [26] which is quickly solvable (it has no integer variables as TACP) by existing solvers implementing efficient *simplex* methodologies. The BDT metric can be directly derived from $X^{y,z}_{k,i,j}$ variables. Since routing schemes is out of scope of this article, the interested reader can refer to [13,26] for further details.

It is worth clarifying that the stated fitness function includes no penalty to those CPs whose resource restrictions are unmet. Indeed, CPs whose resources are not implementable in the final satellite network, must be handled and filtered in a different procedure as discussed below.

3.3. Constraint handling

Since encoded solutions do not necessarily represent a CP that satisfies every link restriction, both the initial population and the offspring from the crossover and mutation operators needs to be analyzed and eventually modified. This is noted as `repair(P)` in lines 5 and 20 of Algorithm 1. In general, this techniques are known as *constraints handling* among which *repairing strategies* are a popular choice for decision-based problems with binary encoding [27]. In particular, repairing strategies consist in transforming an infeasible solution into a feasible one. As per TACP-EA encoding, the repairing procedure needs to enforce and eventually fix bidirectional mismatching within a chromosome (i.e., if $Y_{k,i,j} = 1$, then $Y_{k,j,i} = 1$ as well and reversal). Also, it should detect possible links restriction noncompliance (i.e., $\sum Y_{k,i,j} \leq maxLinks_i \ \forall i$) of all individual in P. For instance, a solution S with $Y_{3,3,2} = 1$ and $Y_{3,4,3} = 1$ represents two simultaneous contacts for node 3 with $maxLinks_3 = 1$ in the network illustrated Fig. 7 and must be therefore repaired.

Algorithm 2. CPD-EA repair procedure

```
   input  : P_input[S]
   output: P_output[S]
1  for ∀S ∈ P_input do
2      for ∀Y_{k,i,j} ∈ E(S) do
3          if Y_{k,i,j} ≠ Y_{k,j,i} then
4              Y_{k,i,j} ← Y_{k,j,i};
5      for ∀k, i ∈ E(S) do
6          links ← 0 ;
7          for ∀j ∈ Y_{k,i,j} do
8              if Y_{k,i,j} = 1 then
9                  links ← links + 1 ;
10         if links(node_i) > maxLinks_i then
11             linksToDisable ← links(node_i) − maxLinks_i;
12             for ∀linksToDisable do
13                 disableRandomLink(E(S));
```

The proposed repairing strategy is detailed in Algorithm 2 which basically goes through all the population P in order to fix infeasible CP chromosomes. In lines 1–4 of the Algorithm 2, all bits of the encoding E are evaluated. If any bi-directional contact is part of the decision variables it might be disabled or enabled in order to match its corresponding edge pair ($Y_{k,i,j} = Y_{k,j,i}$). Next, from line 5–13, the Algorithm 2 disables the contact in case a link constraint is unmet. In particular, for each chromosome, and for all edge of each node i, the enabled links are accumulated in *links* between lines 5 and 10. Then, if the link count is higher than the allowed ($maxLinks_i$), the repair procedure is triggered through lines 10–13 where a total of *linksToDisable* links are randomly disabled in the encoding $E(S)$ (`disableRandomLink(E(S))`). When this happens, bi-directionality constraint is also maintained.

As a result, this repairing technique guarantees that every CP in the solution space (P) meet the available satellite resources. This repairing process is executed every time that new CP are included among the population. In particular, this can be reflected in lines 4 and 20 of Algorithm 1.

3.4. Initialization, replacement, and stopping criteria

On lines 1–4 of the Algorithm 1, the initial population P is *created*. As an overall consideration, we create a population of $P_{size} = 20$ individuals which is a general recommendation for this kind of EA formulations [24]. Despite different strategies can be considered, we initially propose a *random initialization* where all decision variables in set E are uniformly distributed on per bit basis (line 4). Indeed, in line 5, a repairing technique needs to be applied in order to maintain bi-directionality and links restriction as previously explained in Section 3.3.

After repairing, in lines 9–11 of Algorithm 1, the initial population fitness is evaluated to later derive the probability of *selection* of each individual S ($probSel(S) = fitness(S)/\sum fitness(S') \quad \forall S' \in P$). The probability of selection is then subjected to a *roulette wheel selection* which is the most common selection strategy for evolutionary algorithms [24]. Once the parent assignation is filled up, the crossover is performed among them guided by the general parameter of *probability of crossover* (pCr). Indeed, within lines 16–20, the pCr and pMt determines if a given pair of previously chosen individual S are to be considered for the crossover and mutation operator respectively.

Next, a *replacement* strategy need to be considered to change the individuals in previous generation by the new offspring. In the literature, these strategies encompasses egalitarian mechanisms where sub-optimal individuals still have a considerable probability of subsisting in order to promote a wider exploration of the solution space. Nevertheless, in this particular EA, the repairing technique provides enough exploration capability as it drastically modifies the offspring in a random fashion. As a result, in Section 4.1 we show that an *elitist* replacement strategy provides a better overall performance than a random approach and a parent replacement scheme in particular case study. The adopted replacement strategy can be observed in lines 21 and 22 of Algorithm 1.

Table 2
Case study time lapses and orbital parameters.

Interval start	January-1st, 2016, 0 h 0 min 0 s
Interval end	January-1st, 2016, 3 h 22 min 36 s
Bstar coefficient (/ER)	0
Inclination (°)	98
RAAN (°)	0, 5, 10, and 15
Eccentricity	0
Argument of perigee (°)	180, 185, 190, 195
Mean anomaly (°)	0
Mean motion (rev/day)	14.92
Height (km)	600 km

Finally, we have configured a *static* stopping criteria where the iterations quantity of the algorithm is parametrically controlled (*iter* parameter). The manual intervention in the duration of TACP-EA is necessary to maintain time-bounded CPD cycle. We leave the research of possible dynamic stopping criteria and their impact in satellite networks as future work. Once the iterations are completed, the best individual is taken from the population and returned as the best TACP-EA CP in line 23.

4. Performance analysis

In this section we describe a particular and realistic case study of a sample satellite network in order to analyze and compare the performance of the proposed TACP-EA with the theoretical model of TACP. As previously stated, to the best of author knowledge, TACP-EA is the first heuristic approach towards traffic-aware CPD, there is no other heuristic method that could be considered in this comparison.

The case study is comprised of a 4 polar-orbit DTN Low Earth Orbit (LEO) satellite network with the orbital parameters shown in Table 2. The latter are deliberately configured to have one node ahead of the other on the trajectory vector with a slight variation of the right ascension of the ascending node (RAAN) angle. Other parameters in the table are provided as reference to reproduce the satellite orbits. This scenario is illustrated in Fig. 1 and is of particular interest for Earth observation missions as sensing instrumentation accounts for maximum relative distance (coverage) on populated areas while approaching each other in the poles. In these areas, contacts become feasible between adjacent spacecraft producing a train-like formation with an ISL distance of 500 km where two directive point-to-point antennas (placed in front and back of each satellite) conforms a network like the one shown in Fig. 5.

The analysis is presented in two stages: an initial simple study where the CP is designed for an interval of 3 h, 22 min and 36 s and an extended scenario where the total propagation time is 12 h. In particular, the former allows to easily obtain and present an optimal solution and to perform a basic parameter tuning for TACP-EA. Next, the 12 h scenario is presented as a realistic case study as the suggested interval is also the average time lapse between each satellite to ground access. This implies that a CP need to be designed in less than this time in order to be timely provisioned. To derive the contacts start and end time during both intervals, we have used accurate satellite propagators described in [7].

4.1. Simple case study

In the simple case study, the contact arrangement of Fig. 5 is repeated for 4 half-orbits to conform the topology proposed in this analysis. In this scenario, traffic is configured to be equitably generated in all N_2, N_3, and N_4 nodes at the first state of the topology (k_1) to then flow towards N_1 which is expected to later deliver the data by means of a space-to-earth high speed downlink transponder. This behavior resembles a simultaneous acquisition in the equator area and its consequent accumulation in a single node for later download. In particular, communications systems are constrained to up to one ISL per node configured with a 1 Mbps full-duplex throughput within a maximum range of 700 km. For sake of simplicity, transmitted packet sizes are set to 1 Mbit or 125 KByte which at 1 Mbps should occupy the channel for 1 s. Finally, we set the traffic load to, 67.5 MBytes or 540 packets per node.

Since the case study is simple enough, the optimal solution of the problem can be calculated. In particular, the resulting CP can be determined by encoding the theoretical model into GLPK [23] solver. The resulting decision variables are illustrated in Fig. 8 evidencing an optimal BDT of 4488 seconds since the beginning of the topology in state k_{10b} (2nd half-orbit period). It is interesting to note that state k_{10} has been fractionated in three states k_{10a}, k_{10b} and k_{10c} for better CPD granularity. The interested reader can refer to [13] for further detail on state fractionation.

With the theoretical TACP formulation, the overall system energy can also be optimized as edges in the final CP are only enabled if used to carry traffic. All remaining edges are disabled for the remaining of the topology interval. This allows to TACP MILP formulation to be the most effective scheme in terms of system resource usage among other existing procedures [12,13]. However, this comes at the expense of computational complexity further discussed below.

In order to evaluate TACP-EA on this case study, its input parameters (*pCr*, *pMt*, and *iter*) should be determined in the context of the proposed scenario. To this end, literature recommends to perform an empirical exploration by using a statistical method such as an *off-line* parameter tuning [24]. In particular, we perform a statistical analysis of 100 executions of the algorithms under four different crossover (*pCr*) and mutation (*pMt*) probability parameter set. The resulting box-plots of the BDT derived from the objective function are illustrated in Fig. 9.

From the previous analysis, the set ($pCr = 0.6$, $pMt = 0.1$) delivers the best media metric with a value of 6926.64 s of BDT which can be considered an acceptable value in comparison with the optimal of 4488 s. These particular parameters evidences a slightly better TACP-EA performance since the second and third quartile are closer to the media than the other values of *pCr* and *pMt*. In general, this result tends to outbalance the fact that a few outliers (black dots in the box-plots) CPs appeared along the TACP-EA executions. Nonetheless, this results proves that TACP-EA is not very sensible to variations in the mutation and crossover rates. Furthermore, in the same Fig. 9, three replacement criteria were evaluated for $pCr = 0.6$ and $pMt = 0.1$ showing that the elitist replacement proposed in Section 3.4 provides the best algorithm behavior.

Regarding the optimality of TACP-EA, It can be observed from the topology of Fig. 8, that the BDT for the TACP-EA solution is after the state k_{13}. This imply that the traffic over the TACP-EA CP would finally arrive to its destination (N_1) in the third half-orbit instead of the second one. Furthermore, this suggest that more edges are required to carry traffic requiring of more power consumption in DTN nodes than TACP. Even though the system performance is degraded, the processing time required by TACP-EA is drastically minor than TACP.

Both TACP theoretical model and TACP-EA algorithm were executed on the same benchmarking hardware. The host was configured with an Intel Core i3 (2nd Gen) 2310M at 2.1 GHz processor with 8 GB of DDR3 SDRAM memory at 1333 MHz and a Linux-based (Ubuntu 14.04) OS. In this platform, the processing time required for the MILP resolution in GLPK MIP solver was exactly 18,141 s (slightly more than 5 h). On the other hand, as expected, the TACP-EA execution time depended on the iterations configured in the algorithm. Fig. 10 illustrates the objective function value

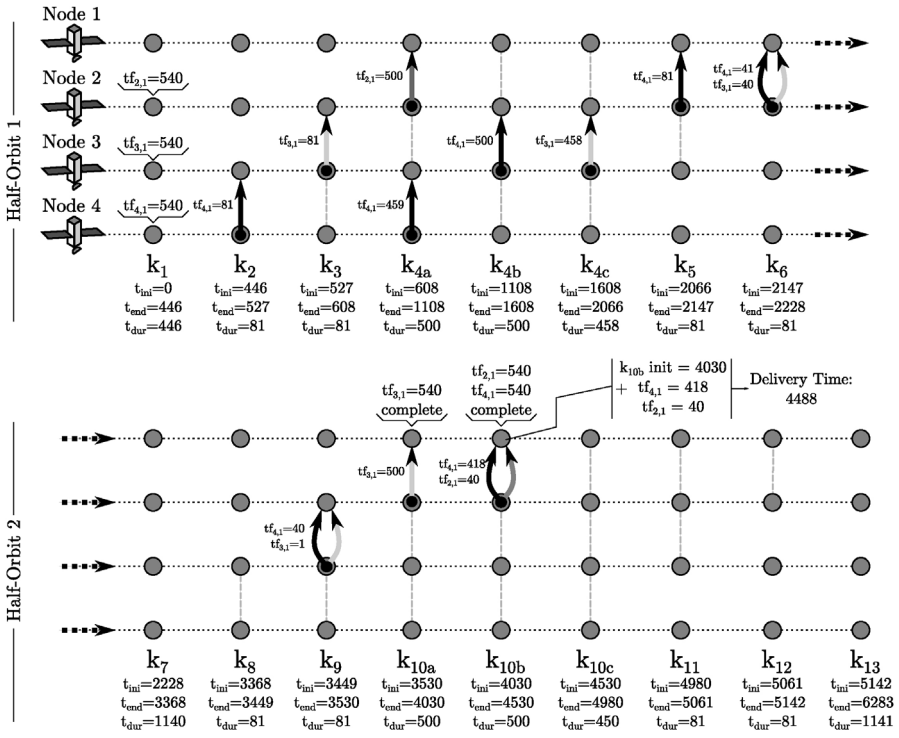

Fig. 8. TACP CP for the proposed network.

for different iterations (*iter*) of the genetic algorithm and its corresponding processing time. From the latter it can be concluded that, besides a few exceptional outliers, the TACP-EA is rather insensitive to the increase of *iter* at least up to a value of 100. Also, it clearly shows that EA execution outperforms non-commercial MILP solvers execution time. Indeed, the proposed TACP-EA can deliver implementable CPs in a constrained time-frame of 30 iterations for the proposed case study scenario.

In this particular and simple topology with 3 h and 22 min of orbiting time we were able to compare and analyze the performance of the EA against an optimal result derived by a MILP solver. However, the MILP solver processing time exponentially increases with the quantity of satellites (nodes) or topology states *k*. To further stress the problem size, in Section 4.2 we provide an extended case study where GLPK directly fails to deliver an optimum CP in the specified hardware.

4.2. Extended case study

In order to evaluate TACP-EA with a more complex topology, it is not necessary to include more satellites but to extend the topology interval. To this end, we propose to extend the scenario summarized in Table 2 from 3 h and 22 min up to 12 h of topology-interval. Indeed, this interval is of greater realism given the revisit rate of LEO satellites are typically around two times a day depending on the orbit and the location of the ground station. This implies that during this time-period, a CP must be designed at least one time. Nonetheless, it is generally advisable to design CP with intervals longer than 12 h so that in case of contingencies in ground stations, the nodes can still have a topological database.

Since the topology-interval is increased, the overall traffic flow on the system can also be raised. In particular, nodes N_2, N_3, and N_4 are set to generate a traffic load of 405 MBytes or 3240 packets

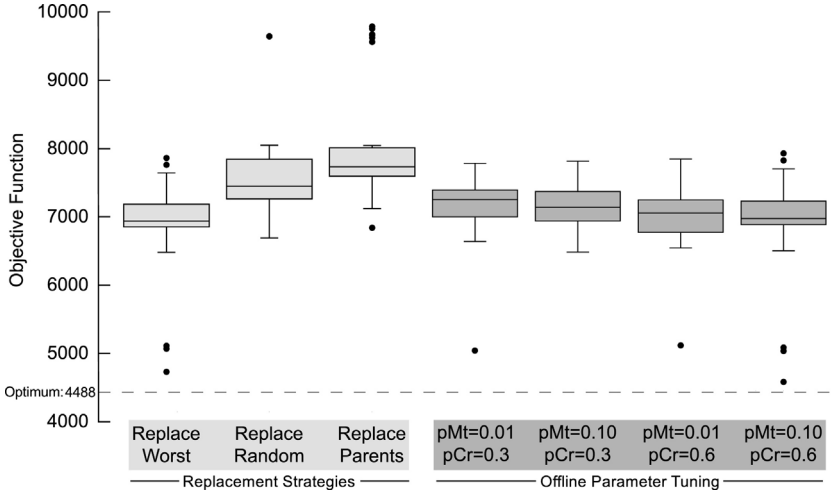

Fig. 9. Box-plots for different combinations of replacement strategies and *pMt* and *pCr* parameters.

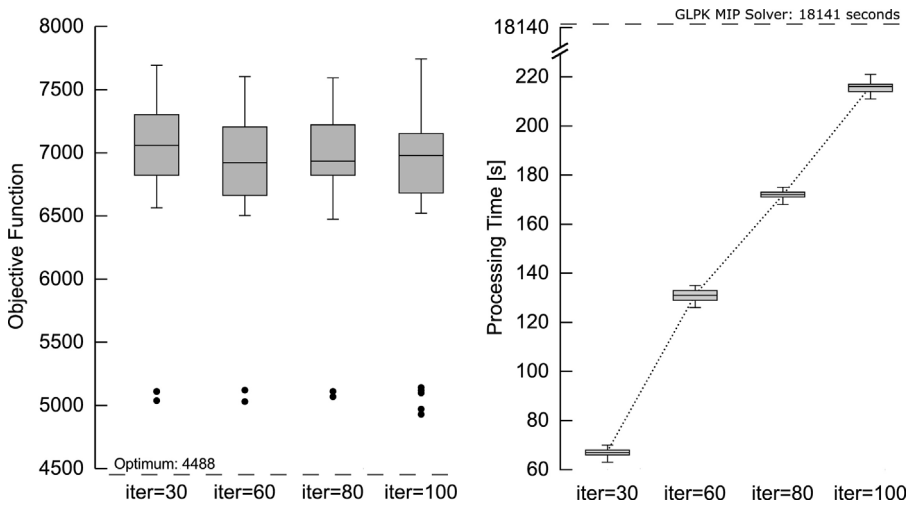

Fig. 10. Objective function value for different iterations of the genetic algorithm and it corresponding processing time.

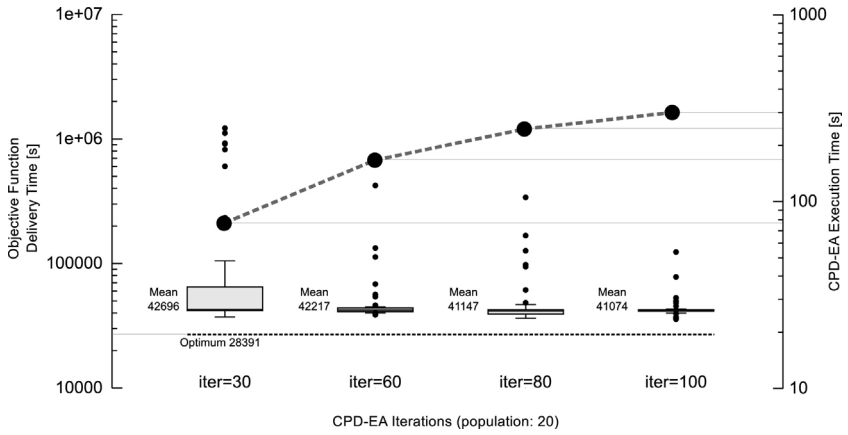

Fig. 11. Objective function value for different iterations of TACP-EA in the extended scenario and the corresponding processing time.

each towards node N_1. This network load can be delivered within the 15 over-the-pole communication opportunities listed in Table 3 which are scheduled to happen with the 12 h duration of the new topology-interval. In particular, each of the 15 half orbits, evidences a topology similar to the one in Fig. 7 resulting in a CP represented by 149 states. The required binary encoding requires of 240 bits to encode a solution of this problem whose decimal value can vary from 0 up to $1.76e^{+72}$.

Table 3
Extended analysis case study communication opportunities in h:min:s.

Half orbit	Relative start time	Relative end time
1	00:07:26	00:37:08
2	00:56:08	01:25:42
3	01:44:43	02:14:24
4	02:33:25	03:02:59
5	03:21:59	03:51:33
6	04:10:33	04:40:07
7	04:59:07	05:28:41
8	05:47:41	06:17:15
9	06:36:15	07:05:49
10	07:24:49	07:56:03
11	08:13:23	08:42:57
12	09:01:58	09:31:39
13	09:50:40	10:20:14
14	10:39:14	11:08:48
15	11:27:48	11:57:22

In this complex scenario, the GLPK solver was not able to deliver a feasible solution after 16 h of processing in the hardware specified in Section 4.1 due to memory exhaustion. Indeed, a High Performance Computing cluster with 64 GB of memory (8 times larger than the benchmark hardware) was required to solve this scenario which resulted in a CP with a BDT of 28,391 s (07 h, 53 min, 11 s). As per Table 3, the optimal BDT of the specified traffic to N_1 could be achieved in the 10*th* half orbit of the topology.

On the other hand, Fig. 11 illustrates the overall performance of the TACP-EA algorithm for this scenario for 30, 60, 80 and 100 iterations with a population of 20 individuals. The parameters of the algorithm remains the same as reported in Section 4.1. In general, the TACP-EA algorithm delivers feasible and reasonable solutions with execution times in the order of a few minutes. In particular, for the 100 iterations executions, the data shows a mean BDT of 41,074 s (11 h, 24 min, 34 s) meaning that most of these sub-optimal CPs will allow the orbiting network to deliver the traffic on the 14*th* or 15*th* half orbit. It is worth clarifying that objective function with values beyond 43200 seconds (12 h) corresponds to a CP that is not able to deliver all input traffic.

With this final analysis, TACP-EA proved to deliver efficient and implementable CPs in bounded time frames when available MILP solvers fail to deliver an optimal solution. Indeed, a computational bottleneck was found for TACP even for a very simple network of 4 satellites and an interval of 12 h. This feature becomes of significant importance to operate and configure complex satellite networks

where CPs need to be timely provisioned in order to keep the system operative.

We leave as further work the analysis of TACP-EA in networks of longer intervals and larger quantity of DTN nodes. Also, the proposal of heuristic alternatives can lead the way to consider the execution of CPD in-orbit instead of ground. This would indeed become a required feature for DTN satellite networks without permanent access to Earth such as those orbiting Mars or other planet in the solar system.

5. Conclusion

In this work we addressed the problem of the CPD for networked satellite systems based on DTN. This type of networks can be particularly optimized by exploiting their predictable nature in order to take efficient planning decisions in advance. Among this, the design of the forthcoming communication opportunities imprinted in a CP was considered for nodes with constrained resources. To this end, we recalled TACP: a theoretical formulation based on a MILP statement which quickly becomes intractable even for medium-sized constellations.

As a result we investigated a meta-heuristic approach as an alternative to support satellite DTN missions operation. In particular we developed TACP-EA: an evolutionary algorithm mimicking the genetic evolution of the species which delivers a sub-optimal yet efficient CP in bounded time frame. Indeed, the latter is not necessarily deterministic in the original version of TACP which and might render it unsuitable for networks that require periodic CP updates.

Throughout this paper, we have described specific strategies implemented in TACP-EA such as repairing to guarantee that each solution satisfies the resource constraints and encoding to efficiently explore the solution space. Finally, we demonstrated that TACP-EA outperforms non-commercial MILP solvers in terms of processing time (even for small-sized satellite networks) while still delivering efficient and implementable CP solutions. We leave as future work an extended analysis with commercial MILP solvers as well as larger DTN networks.

References

[1] H. Rashvand, A. Abedi, J. Alcaraz Calero, P. Mitchell, S. Mukhopadhyay, Wireless sensor systems for space and extreme environments: a review, Sens. J. IEEE 14 (11) (2014) 3955–3970.
[2] J. Chu, J. Guo, E. Gill, Fractionated space infrastructure for long-term earth observation missions, in: 2013 IEEE Aerospace Conference, 2013, pp. 1–9.
[3] J. Wertz, W. Larson, Space Mission Analysis and Design, vol. 8 of Space Technology Library, 3rd ed., Springer, Netherlands and Torrance, CA, USA, 1999.
[4] A. Krishnamurthy, R. Preis, Satellite formation, a mobile sensor network in space, in: Proceedings. 19th IEEE International Parallel and Distributed Processing Symposium, 2005, 2005, p. 7.
[5] C. Caini, H. Cruickshank, S. Farrell, M. Marchese, Delay and disruption tolerant networking (DTN): an alternative solution for future satellite networking applications, Proc. IEEE 99 (2011) 1980–1997.
[6] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM'03, New York, NY, USA, ACM, 2003, pp. 27–34.
[7] D.A. Vallado, Fundamentals of Astrodynamics and Applications, 4th ed., Microcosm, Hawthorne, CA, 2007.
[8] C. Caini, R. Firrincieli, Application of contact graph routing to LEO satellite DTN communications, in: 2012 IEEE International Conference on Communications (ICC), 2012, pp. 3301–3305.
[9] J.A. Fraire, J.M. Finochietto, Design challenges in contact plans for disruption-tolerant satellite networks, IEEE Commun. Mag. 53 (5) (2015) 163–169.
[10] M. Huang, S. Chen, Y. Zhu, Y. Wang, Cost-efficient topology design problem in time-evolving delay-tolerant networks, in: 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), 2010, pp. 1–5.
[11] M. Huang, S. Chen, F. Li, Y. Wang, Topology design in time-evolving delay-tolerant networks with unreliable links, in: 2012 IEEE Global Communications Conference (GLOBECOM), 2012, pp. 5296–5301.
[12] J. Fraire, P. Madoery, J. Finochietto, On the design and analysis of fair contact plans in predictable delay-tolerant networks, Sens. J. IEEE 14 (11) (2014) 3874–3882.
[13] J. Fraire, J. Finochietto, Routing-aware fair contact plan design for predictable delay tolerant networks, Ad-Hoc Netw. 25 (Part B) (2015) 303–313.
[14] H. Yan, Q. Zhang, Y. Sun, J. Guo, Contact plan design for navigation satellite network based on simulated annealing, in: 2015 IEEE International Conference on Communication Software and Networks (ICCSN), 2015, pp. 12–16.
[15] J.A. Fraire, P.G. Madoery, J.M. Finochietto, Traffic-aware contact plan design for disruption-tolerant space sensor networks, Ad Hoc Netw. 47 (2016) 41–52.
[16] D.J. Lary, An objectively optimized earth observing system, in: 2007 IEEE Aerospace Conference, 2007, pp. 1–3.
[17] J.A. Fraire, P.G. Madoery, J.M. Finochietto, G. Leguizamón, Preliminary results of an evolutionary approach towards contact plan design for satellite DTNs, in: 2015 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), 2015, pp. 1–7.
[18] N. Bezirgiannidis, F. Tsapeli, S. Diamantopoulos, V. Tsaoussidis, Towards flexibility and accuracy in space DTN communications, in: Proceedings of the 8th ACM MobiCom Workshop on Challenged Networks, CHANTS'13, New York, NY, USA, ACM, 2013, pp. 43–48.
[19] E. Birrane, S. Burleigh, N. Kasch, Analysis of the contact graph routing algorithm: Bounding interplanetary paths, Acta Astronaut. 75 (2012) 108–119.
[20] J.A. Fraire, P. Madoery, J.M. Finochietto, E.J. Birrane, Congestion modeling and management techniques for predictable disruption tolerant networks, in: 2015 IEEE 40th Conference on Local Computer Networks (LCN), 2015, pp. 544–551.
[21] J.A. Fraire, P.G. Madoery, J.M. Finochietto, P. Ferreryra, R. Velazco, Internetworking approaches towards along-track segmented satellite architectures, in: 2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), 2016 (in press).
[22] S. Even, A. Itai, A. Shamir, On the complexity of time table and multi-commodity flow problems, in: 16th Annual Symposium on Foundations of Computer Science, 1975, 1975, pp. 184–193.
[23] A. Makhorin, GLPK (GNU Linear Programming Kit), Department for Applied Informatics, Moscow Aviation Institute, Moscow, Russia, 2015 http://www.gnu.org/software/glpk/.
[24] E.-G. Talbi, Metaheuristics, From Design to Implementation, Wiley, 2009, ISBN: 978-0-470-27858-1.
[25] J. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1992.
[26] J. Alonso, K. Fall, A Linear Programming Formulation of Flows Over Time with Piecewise Constant Capacity and Transit Times, Technical Report IRB-TR-03-007, Intel, 2003.
[27] E. Mezura-Montes, Constraint-Handling in Evolutionary Optimization, 1st ed., Springer Publishing Company, Incorporated, 2009.