# Nanosatellite constellation control framework using evolutionary contact plan design

1st Carlos E. Gonzalez
*Electrical Engineering Department*
*University of Chile*
Santiago, Chile
carlgonz@uchile.cl

2nd Alexandre Bergel
*Department of Computer Science*
*University of Chile*
Santiago, Chile
abergel@dcc.uchile.cl

3rd Marcos A. Diaz
*Electrical Engineering Department*
*University of Chile*
Santiago, Chile
mdiazq@ing.uchile.cl

***Abstract*—Space agencies, educational institutions, and private companies have adopted CubeSat nanosatellites to do scientific research, training, technology demonstration, and space-based industries in the New Space era. The next step in this changing space sector corresponds to the assembly and operation of large satellite constellations consisting of hundreds or thousands of small- or nano-satellites. This context adds new requirements and challenges to the production and operation lines of these space projects. This work focuses on the agile operation of a large nanosatellite constellation with inter-satellite communications. We propose using the constellation contact topology to design contact plans using evolutionary algorithms and use the contact plan information to control the constellation operations. The contact plan is then used to create a Global Flight Plan table that summarizes all the operations required to execute a proposed task. Thus, satellites and ground station nodes only need a flight software capable of queuing, executing, and transferring Flight Plan commands. The evolutionary contact plan design approach shows promising scalability results opening the possibility of controlling satellite mega constellation of hundreds or thousands of nanosatellites.**

***Index Terms*—CubeSat, constellation, contact plan design, evolutionary algorithm**

## I. Introduction

CubeSat nanosatellites [1] have demonstrated that a cost- and time-effective access to space is possible. Space agencies, educational institutions, and private companies have adopted this technology to do scientific research, training, technology demonstration, and space-based industries in the New Space context [2]. In general, satellites can operate independently or in formation flying, i.e., several vehicles are used to accomplish a mission cooperatively. The most common satellite formations flying are trailing, cluster, and constellation [3] using Walker-delta or Walker-star geometries [4]. However, it is unlikely that a large or mega CubeSat constellation can be deployed with a specific geometry. Instead, Ad-hoc configurations resulting from several secondary payload launching opportunities have been used more frequently [5], [6].

Today, the trend in this changing space sector is the assembly and operation of large satellite constellations consisting of hundreds or thousands of small or nanosatellites [7]–[9]. Such a number of satellites have no precedents and propose new challenges. Currently, satellite operations largely depend on human operators, statically assigned ground capabilities, or homogeneous satellite networks. Constellations with inter-satellite capabilities, except for a few companies, are not widely deployed yet. Small and nanosatellites add numerous challenges to this problem: heavy restrictions in space, power, and communications capabilities, plus different configurations, short life cycle, and rapid technological evolution. These challenges may stress satellites' production lines [10].

Radhakrishnan *et al.* (2016) [3] review the challenges of constellations missions with small satellites from Physical to Network layers of the OSI model. However, the assembly and operation of these space systems also present many challenges in the Application Layer, i.e., the flight software to control a large nanosatellite constellation. Table I summarizes the different efforts and challenges from the Physical to the Application layers.

From the application layer point of view, a high level of automation is required to optimize small satellite constellation capabilities. Translation from high-level requirements to the actual constellations capabilities is required to facilitate the operation of these systems. Assigning earth and space resources, distributing goals, or propagating changes in the constellation system are complex problems that require intelligent algorithms and heuristics to be solved [17], [18]. Also, the deployment, maintenance, data acquisition, routing, and optimization of constellation operations are complex scheduling problems [13]. Even the simplest versions of these scheduling problems are Mixed Integer Linear Programming (MILP) and hence NP-hard class [19]. Therefore exact solution algorithms can be impractical for large constellations of thousands of nodes. On the other hand, heuristics approaches such as Evolutionary Algorithms algorithms have been used with promising results.

From the Network and Transport layers perspective, traditional TCP/IP protocols are not well suited for a satellite constellation due to long-delay and low reliable communication links. CCSDS protocols are more widely used in space applications but with limited use cases in large constellations. The family of DTN routing protocols is better suited in LEO constellations with ISL. In contrast with TCP/IP stack protocol, Delay or Disruption Tolerant Network (DTN) protocols assume that contacts among nodes are sporadic, so nodes require a buffer to store and carry messages until a link is

TABLE I
SMALL- AND NANOSATELLITE CONSTELLATION CHALLENGES SUMMARY

| OSI model | Challenges |
|---|---|
| Application<br>Presentation<br>Session | Flight software architectures capable of scaling to assembly hundreds or thousands of small or nanosatellites in an agile fashion [11]. Support an autonomous operation of the constellation. Optimize the usage of the constellation resources (computational, energy, lifetime, etc.) [12], [13] |
| Transport<br>Network | Network protocols that support long delays and/or interrupted communication links [14]. Protocols aware of computational resources, energy, and link capacity limitations [15]. |
| Data-link<br>Physical | Inter-satellite communication links for small or nanosatellites with energy, pointing, and space limitations [3]. MAC protocols for large wireless network [16]. |

available. Despite the time-evolving nature of the connections in a Low Earth Orbit (LEO) satellite network, the contact opportunities can be predictable due to orbital mechanics [14]. Therefore, the contact information can be used to design contact plans with different goals [14], [15], [20].

All the operations mentioned above must be supported in the satellite flight software as well as the ground control nodes [12], [13]. Previous works have remarked that modular, extensible, and reliable flight software architectures are required to deliver quality software in less time and with less effort [21], [22]. In previous work, we describe a nanosatellite flight software solution focused on the extensibility, modularity, reusability, and scalability to constellations called SUCHAI Flight software [23]. This software is capable of executing remote commands and also implements a Flight Plan to execute scheduled commands.

Despite the incipient deployment of small- and nanosatellite mega-constellations, the studies in task scheduling, and advances in flight software development, there is a gap in solutions that integrate those concepts to scale the production and operation of nanosatellites constellations from tens to hundreds or thousands of nodes. Thus, in this work, we present a nanosatellite constellation control framework (See section II) that uses the contact topology information to design a global flight plan. Satellites have to execute the flight plan to solve a particular task cooperatively. This global flight plan is created using a contact plan, just as in DTN routing protocols. An evolutionary algorithm is used to design this contact plan, considering scaling to a large number of nodes. Based on typical CubeSats' hardware and software capabilities, we decided to delegate the scheduling problem to the ground nodes and deliver a global flight plan to satellites. Initial validation of these ideas is shown in Section III through a case study.

## II. SATELLITE CONSTELLATION CONTROL FRAMEWORK

Figure 1 shows a scenario with three nanosatellites and two ground stations. Satellites are equipped with some earth observing payload and inter-satellite links (ISL). Let consider that the satellites and ground stations software supports the execution of commands scheduled in the flight plan table as described in the previous section. Also, let define a simple task as taking an instrument's data over a certain location and download that data in the designated ground station in a shorter period. Two key variables must be considered to solve this problem: commands and contact opportunities. Commands are satellites and ground stations' actions, including sending data to nodes and taking data from instruments. Contact opportunities may refer to two ideas: the instants where nodes can establish radio links or when targets are visible to nodes. These two variables can be expressed in a global flight plan that includes the time, node, and command to be executed to accomplish a task. This flight plan must be distributed to all nodes using the inter-satellite links. LEO nanosatellite constellations with ISL can be considered a DTN, so it is possible to use the contact information to schedule which nodes are used in a task. In a DTN, for a given Contact List (CL), there are many possible paths to visit the targets defined in a task. Finding a Contact Plan (CP) that satisfies a set of restrictions is called Contact Plan Design (CPD) [14].

Thus, the constellation control framework aims to generate a global flight plan that solves a specific task for a certain scenario. Generating a valid flight plan is a scheduling problem. Variables such as delivery time, starting time, or resource usages are optimized under contact feasibility and node capabilities restrictions. In this work, an evolutionary contact plan design approach is used to solve the scheduling problem and generate a valid flight plan. The framework consists of three main modules: a contact list generator, the contact plan design, and the flight plan generator. The contact list generator module uses the scenario definition to determine future satellite to satellite, satellite to ground stations, and satellite to targets contacts; satellite capabilities such as communication system and payload instruments parameters are used to define a feasible contact. According to the restrictions defined in the task, the contact plan design module searches for a valid solution in the space of all contact opportunities. Finally, the selected contact plan is translated into a flight plan containing the commands that nodes must execute to complete the proposed task.

Using the CL Finite State Machine (FSM) representation (see Fig 2), the CPD consists on selecting a sequence of nodes $S = [S_1, ..., S_L]$ at states $K = [k_1, ..., k_L]$ that visits the nodes defined in the task. For example, let defined the following task: execute the command `get_data data1` over the South

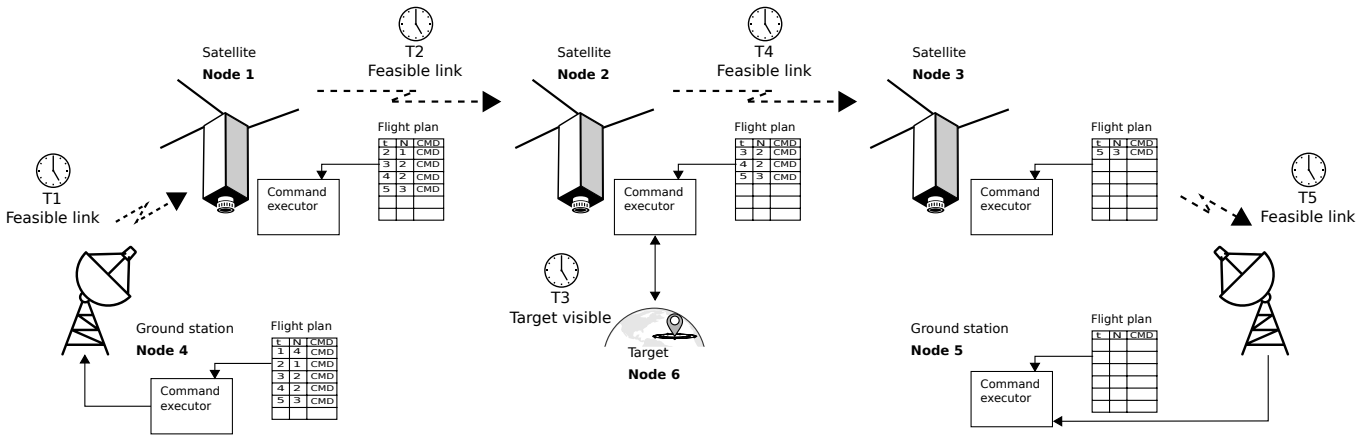



Fig. 1. Constellation working scheme

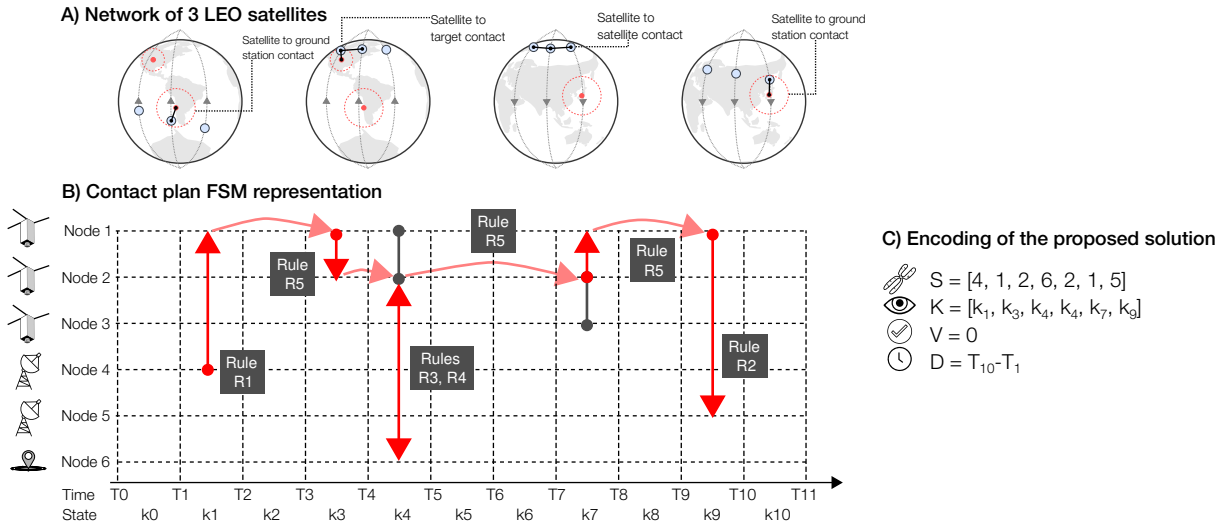



Fig. 2. A) A LEO constellation with 3 satellites with ISL and two targets. B) Contact List an Contact Plan FSM representation; also, CPD Design rules. C) Encoding of an example Contact Plan solution

Atlantic Anomaly (SAA) (*node 6*) and download this data on Tokyo ground station (*node 5*) starting from Santiago ground station (*node 4*). The Fig. 2 show a valid CP that selects the contacts of the sequence of nodes $S_1 = [4, 1, 2, 6, 2, 1, 5]$ at states $K_{S1} = [k_1, k_3, k_4, k_7, k_9]$. Note that this is not the only valid sequence, and that in real scenarios the solution space grows significantly.

Using this example CP, it is possible to define the **delivery time** of the task as $D_{S1} = T_10 - T_1$, the sequence length $L = 7$ and the **validity** of the solution $V = 0$. These variables are described below:

1) **Validity (V):** Because this work includes the targets and ground stations in the CL, not all sequences of contacts are valid and some rules are added to the CPD to determine if a given CP is valid or not:

R1. The CP must start in the ground station defined by the task.

R2. The CP must end in the ground station defined by the task.

R3. The CP must visit all targets defined in the task.

R4. Targets are not data relays. That is, if a contact from satellite A to target T occurs, the next contact must start from the same satellite A

R5. Satellites are data relays. That is, if contact between satellite A to satellite B occurs, the next contact must start from the satellite B.

If one or more of these rules are not satisfied in a CP, we define it as invalid. Thus, validity is defined as the sum of contact rules not satisfied per state $k$ of the CP: $V_k = 0$ if R1. to R5. are meet, else $V_k = 1$. So, the sum of contacts validity $V = \sum_{k=1}^{n} V_k$ must be zero to define a sequence as valid.

2) **Delivery time (D):** Delivery time is defined as the total time the task takes to execute. If the sequence start at

state $k$ and ends at state $l$, then it the time between the state $k$ start time ($T_k^{start}$) and the state $l$ end time ($T_l^{end}$): $D = T_l^{end} - T_k^{start}$.

3) **Number of contacts (L):** The length of the solution or the number of contacts used in the contact plan $L = length(S)$.

Thus, the CPD is an optimization problem, primarily over the *delivery time* variable, subject to the *validity* of the solution:

$$minimize: D = T_l^f - T_k^i \tag{1}$$

$$subject\ to: V = \sum_{k=1}^{L} V_k = 0 \tag{2}$$

$$0 < D \leq T_n^f - T_0^i \tag{3}$$

$$0 < L \leq n \tag{4}$$

$$\tag{5}$$

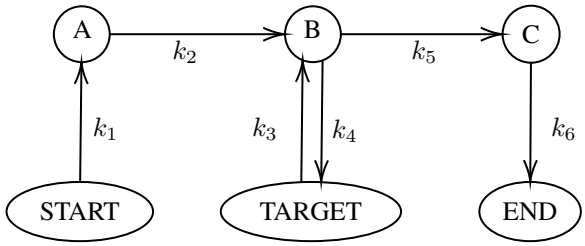
Different approaches can be used to solve the CPD problem; however, as the number of nodes increases, classical optimization techniques are no practical, and evolutionary algorithms have been proposed in the literature [24]. In this work, a genetic algorithm is used to find a CP.

*1) Genetic algorithm:* The proposed GA for the Constellation Control Framework CPD is designed to generate multiple CP candidates and evaluate its *validity* and optimize the *delivery time* function. First, the algorithm is designed to find valid contact paths evolutionarily, a problem similar to find the escape route in a labyrinth. Then, the algorithm focuses on minimizing the cost of this path.

The algorithm requires the scenario and task definitions as inputs because this information will be used to generate the initial population according to the encoding and validity rules (See Sections II-2 and II-4. The CL is used to evaluate the individual's fitness function because a proposed CP sequence has to be contrasted with actual contact opportunities; thus, infeasible or low-quality solutions will be discarded in the evolution (See section II-3). Other parameter are: $L$, the sequence length; $P_{size}$, the population size; $Iter$, the maximum number of iterations; $p_{mut}$, the mutation probability; and $p_{cr}$, the crossover probability. Individuals are generated as random sequences, but they are fixed using the task information, so individuals always contain the start, target, and end node as described in Section II-4. Then, a population is evaluated obtained the $Validity$ and $Fitness$ values. As described in Section II-3, *Validity* represents how feasible is the solution and *Fitness* evaluates the *delivery time*. During the tournament, individuals are sorted first by $Validity$ and then by $Fitness$; thus, the algorithm first finds feasible solutions and then optimizes the *delivery time*. In each iteration, variability is generated by replacing the old population with the best individuals, new individuals product of the crossover operation, or a mutation according to the probabilities and operator described in Section II-6 and II-5. Finally, after $Iter$ iterations or if the stop condition described in Section II-4 the algorithm returns the best individual.

*2) Encoding:* In this work, individuals are encoded using a list of integers. Thus, the sequence $S = [s_1, ..., s_L]$ represents a list of nodes $s_i$ to visit to execute the CP or FP associated with a task definition. This work's approach is to use the information of the task and the validity rules to encode valid individuals (but not necessarily feasible) from the beginning. Consider the situation described in Figure 2, where the task is taking a data sample from *node 6 (Target)*, starting from *node 4 (Start)*, and finishing in *node 5 (End)*. The following diagram describes the situation.


Of course, at this point, we do not know how to travel to these nodes. Let say we can move through $Start$, $Target$ and $End$ nodes, using the nodes (satellites) $\{A, B, C\}$ which encodes to the sequence $S = [Start, A, B, Target, B, C, End]$ ($L = 7$). This genotype $S$ is expressed as a phenotype $K = [k_1, ..., k_6]$, which is the sequence of states $k_i$ where the contacts are feasible according to the CL. This situation is described in the following diagram.




In the sequence $S = [Start, A, B, Target, B, C, End]$, the $Start$, $Target$, and $End$ nodes are known (fixed in the task definition). Thus, the algorithm has to find the values $A, B,$ and $C$. These values are generated randomly, and the genotype $K$ is obtained during the fitness function evaluation by searching sequentially in the CL for states that makes the sequence $S$ feasible. If there are less than $L-1$ states for the sequence $S$, it is an unfeasible or invalid solution. Since $L$ is a parameter, it can be set arbitrarily large because redundant contacts are allowed. Figure 2 shows the encoding of a possible solution to the example scenario and task definition.

*3) Constraints and fitness function:* The fitness function is the objective function in Eq. 2, *i.e.,*, the delivery time (also mentioned as sequence duration, or simply duration). However, in this problem, the validity restrictions (See rules R1. to R5.) are absolutely relevant to evaluate a solution. Therefore, validity is also considered in the fitness function, creating a multi-objective optimization problem. Thus, the following fitness function is used.

$$fitness: \quad F_i = (V_i, D_i) \tag{6}$$
$$0 \leq V_i \leq L \tag{7}$$
$$D_i > 0 \tag{8}$$
$$\tag{9}$$

Where $V_i$ is the validity of the sequence $S_i$ and is calculated as the sum of invalid contacts. Let define $V_k$ as 1 if the contact at state $k$ breaks any rule R1. to R5. or 0 if not. If a sequence of $S_i = \{s_1, ..., s_L\}$ of length $L$ contains only valid contacts then $V_i = 0$:

$$V_i = \sum_{k=1}^{L} V_k \tag{10}$$

Delivery time $D_i$ is calculated as the time difference between first and last contact in sequence $S_i$:

$$D_i = (T^{end}_{k_{L-1}} - T^{start}_{k_0}) \tag{11}$$

*4) Initialization and stopping criteria:* Individuals are created from three parameters: the task, the maximum number of nodes allowed ($L$), and the target's position in the sequence ($I$). The *start*, *target* and *end* node numbers are obtained from the task definition. The maximum number of nodes is a parameter defined by the user, which limits the sequence length. This number can be arbitrarily large because redundant contacts are allowed, and the final sequence can be simplified to a short version without repeated contacts. The position of the target nodes is an index randomly chosen, so $I \in [2, L-2]$ and an individual tracks this value to maintain its validity during the genetic operations. A couple of examples of individuals are shown in Figure 3. This idea can be extended to arbitrary large sequences and an arbitrary number of targets.

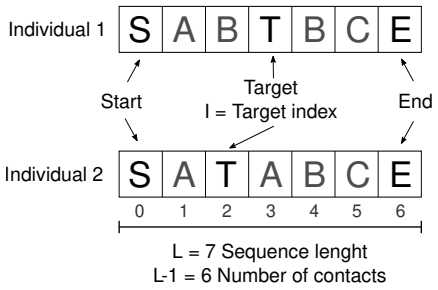



Fig. 3. Individuals initialization. S, T, and E are the node number for the Start, Target, and End nodes, respectively. These values are known from the task definition. The index $I$ of node T is particular to an individual. A, B, and C are unknown and so generated randomly for each individual

Thus, a population of $N$ individuals of length $L$ is created. The target index $I$ and the nodes (other than the *start*, *target* and *end*) are chosen randomly. After the evaluation, each individual keeps track of the sequence of valid contacts $K$ reached by its sequence.

This sequence is a valid contact plan if an individual of length $L$ reaches $L-1$ valid contacts, *i.e.,* $V = 0$. The delivery time or sequence duration $D$ is also evaluated using the information of the CL.

Suppose individuals reached the validity condition ($V = 0$), and there is no noticeable improvement of the delivery time $D$ of the best individual during five consecutive generations. In that case, GA is assumed to have reached convergence, and the stop condition is satisfied.

*5) Mutation operation:* The mutation operation starts selecting an index $i$ of the sequence to mutate. Depending on the value of $i$ there are several cases:

- $i \in \{0, I, L-1\}$: If the mutation index $i$ points to the start, end or target position do nothing. These values cannot be changed.
- $i \in \{I-1, I+1\}$: In any case, replace both $I-1$ and $I+1$ with a new random value. This operation respects the validity rule R4..
- Otherwise: replace the node at index $i$ with a new random value.

Figure 4 graphically describes the mutation operation. Note that this operation always generates a valid sequence.

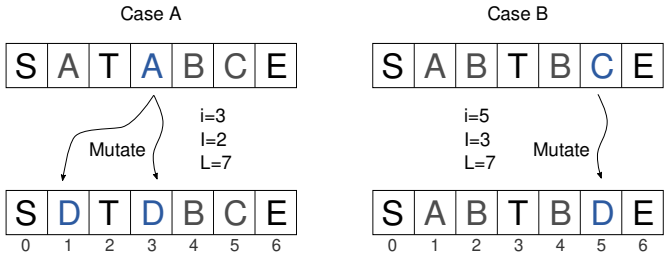



Fig. 4. Mutation operation. Left, case A: $i \in \{I-1, I+1\}$, so both indexes 1 and 3 are mutated. Right, case B: $i \notin \{0, I-1, I, , I+1, L-1\}$ so mutate node at index 5.

*6) Cross-over operation:* To do the crossover between two individuals, a cut point $j$ is selected. The cut point is always the index next to the target index to maintain the sequences valid, i.e., $j = I + 1$. However, the sequence to cut is chosen randomly. Thus, as described in Figure **??** the crossover operation consists of mixing section $S_A[0 : j]$ of the first individual with section $S_B[j : L-1]$ of the second individual, or vice versa.

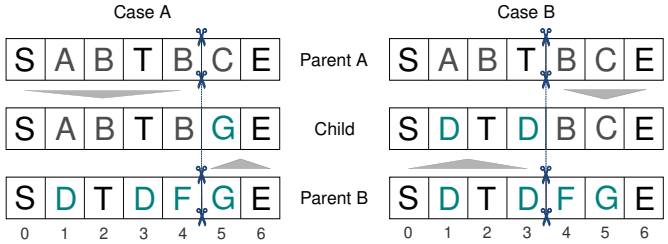



Fig. 5. Cross-over operation. Left, case A: the cut point is chosen from parent A so $j = I_A + 1 = 4$ and the first section comes from parent A while the second section comes from parent B. Right, case B: in this case the cut point is chosen from parent B so so $j = I_B + 1 = 3$ and the first section also comes from parent B

### A. Flight plan design

This module has to find the Flight Plan (FP) that is a valid solution to execute a task using the constellation's capabilities. Figure 6 describes how the Contact Plan (CP) is related to the FP to solve the problem. The CP defines the instants and operations required to visit the nodes involved in the solution, and the FP defines the commands to execute in each node. Commands depend on the contact type and the task definition. Four different commands should be implemented in the nodes' software:

- `set_fp <time> <command>`: This command will be used to set a FP entry. It can be used to queue commands that will be executed in future contact.
- `send_fp <node>`: This command can be used to transfer FP entries to another node during a contact.
- *`get_data <data_id>`*: Satellites must implement specific commands to operate its payloads and get data from targets.
- `send_data <node> <data_id>`: This command can be used to transfer payload data back to the ground stations or between satellites. In combination with the `set_fp` command it is possible to automate data transfers.

The FP contains all the information required to execute the task. This information is sent to the first node, a ground station, to start the task's automated execution. The execution of this global FP can be simulated to validate the solution's feasibility or detect any problem. The remaining question is how to build a constellation capable of executing this kind of FP, *i.e.,*, a homogeneous FS solution (capable of running commands in the satellites and ground station nodes) is required.

## III. Case study

We used a case study to evaluate the performance of the genetic algorithm generating contact plans. This case study is also used to tune the algorithm's hyper-parameters: mutation rate and population size. The scenario consists of a constellation with 10 satellites in an Ad-hoc configuration, two ground stations, and one target. In the Ad-hoc configuration, the orbital parameters are chosen randomly with an altitude between 500 km. and 600 km. We simulate the constellation operation up to 14400 seconds, with a resolution of 300 seconds to calculate the contact opportunities, resulting in 346 contacts. The complete details of this configuration are shown in Table II.

*1) Task 1:* The first task to test was a storage and forward mission. The goal is to execute the command `take_data data1` over the SAA, starting in Santiago ground station to download the result `data1` in Tokyo ground station. The scenario and task definition were loaded into the constellation controller framework to obtain the case's contact plan and flight plan. The hyper-parameters evaluation included 4 population sizes (50, 100, 150, and 200), 4 mutation rates (0.2, 0.4, 0.6, and 0.8). Because of GA's stochastic nature, 100 independent runs of the algorithm for each combination were executed, each run with a different random seed. A summary of the results is shown in Figure 7. Results indicate a convergence of the algorithm when the population size is larger than 150 individuals or the mutation rate is larger than 0.4. The algorithm converges to the fitness value (duration or delivery time) of 5100 seconds. The complete contact list and contact plan for this solution is detailed in Table III.

TABLE II
Scenario description

| **Simulation time** | | | | |
|---|---|---|---|---|
| Start time: 2020-09-30T00:00:00 UTC (1601424000 Unix time) | | | | |
| Simulation time: 14400 seconds (∼2.67 orbits) | | | | |
| Simulation resolution: 30 seconds | | | | |
| Contact list resolution: 300 seconds | | | | |
| Number of contacts: 346 | | | | |
| **Satellites** | | | | |
| Node | Period (min) | Incl | Mean anom. | R. ascension |
| 0 | 95.58 | 99.0° | 24.0° | 136.0° |
| 1 | 95.36 | 89.0° | 146.0° | 78.0° |
| 2 | 95.53 | 96.0° | 66.0° | 161.0° |
| 3 | 95.45 | 97.0° | 80.0° | 178.0° |
| 4 | 96.21 | 98.0° | 126.0° | 177.0° |
| 5 | 94.97 | 83.0° | 52.0° | 59.0° |
| 6 | 94.71 | 91.0° | 150.0° | 73.0° |
| 7 | 95.37 | 92.0° | 171.0° | 84.0° |
| 8 | 96.18 | 95.0° | 61.0° | 20.0° |
| 9 | 95.81 | 82.0° | 121.0° | 14.0° |
| **Ground stations and targets** | | | | |
| Node | Lat. | Lon. | Alt. | Reference |
| 14 | -33.3833° | -70.7833° | 476 m | Santiago, Chile |
| 16 | 35.6830° | 139.7670° | 5 m | Tokyo, Japan |
| 18 | -15.0° | -15.0° | 500 km | S. Atlantic Anomaly |

TABLE III
Task 1 example contact plan and flight plan solution.

| **Contact plan** | | | |
|---|---|---|---|
| from | to | start | end |
| 2 | 14 | 1601431200 | 1601431500 |
| 2 | 8 | 1601432400 | 1601432700 |
| 8 | 18 | 1601433600 | 1601433900 |
| 8 | 16 | 1601436000 | 1601436300 |
| **Flight plan** | | | |
| Time | Node | Command | |
| 1601431200 | 14 | fp_send 2 | |
| 1601432400 | 2 | fp_send 8 | |
| 1601433600 | 8 | get_data data1 | |
| 1601436000 | 8 | send_data 16 data1 | |

*2) Task 2:* A second task was analyzed, consisting of a store and forward operation over two targets. The idea is to execute the command `take_data data1` over STGO, and the command `take_data data2` over SAA starting in Tokyo ground station to download both `data1` and `data2` in the same ground station. Like the previous section, task 2 was solved with the framework, and the GA's performance was evaluated by hyper-parameters analysis. The results for this scenario and task combination are shown in Figure 8 and exhibits convergence for a population size of 100 or

Contact plan

| From | To | Start | End |
|---|---|---|---|
| 4 | 1 | T1 | T2 |
| 1 | 2 | T2 | T3 |
| 2 | 6 | T3 | T4 |
| 2 | 3 | T4 | T5 |
| 3 | 5 | T5 | T6 |

Ground station to satellite or satellite to satellite contact: transfer flight plan

Satellite to target contact: get data from target

Ground station to satellite or satellite to satellite contact: return data and transfer flight plan

Flight plan

| Time | Node | Command |
|---|---|---|
| T1 | 4 | `fp_send 1` |
| T2 | 1 | `fp_send 2` |
| T3 | 2 | `get_data 55a1` |
| T4 | 2 | `send_data 3 55a1` |
| T4 | 2 | `fp_send 3` |
| T5 | 3 | `send_data 5 55a1` |

Fig. 6. Relation between Contact Plan (CP) and Flight Plan (FP)

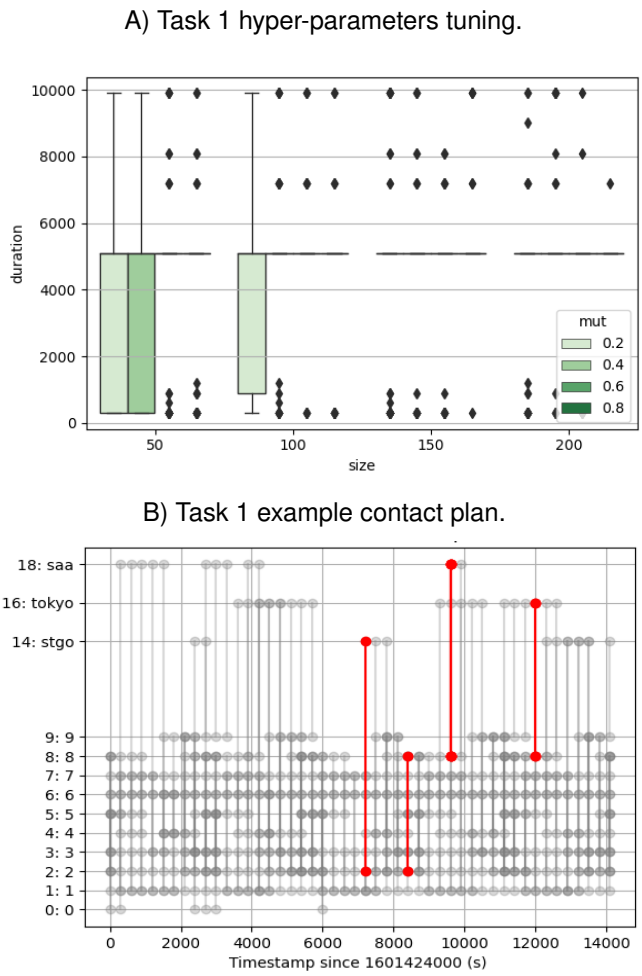



Fig. 7. A: Task 1 hyper-parameters tuning with maximum 8 hops the box plot shows the fitness value. B: Example contact plan FSM representation.

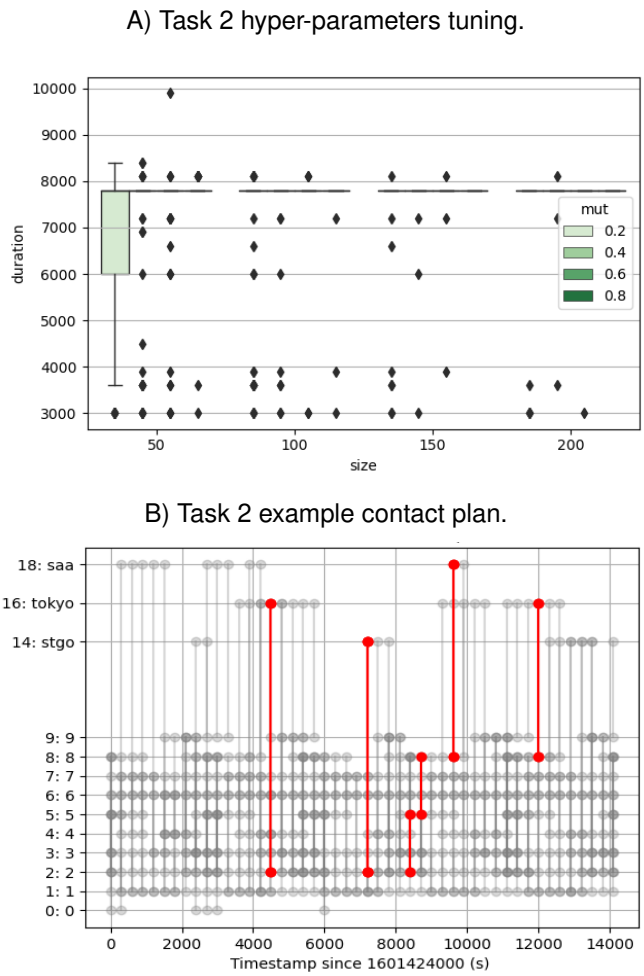



Fig. 8. A: Task 2 hyper-parameters tuning with maximum 8 hops the box plot shows the fitness value. B: Example contact plan FSM representation.

greater. The algorithm converges to the value of 7800 seconds (delivery time), solving the problem in 153 seconds. A graphic representation of an example Contact Plan (CP) is also shown in Figure 8 and the details of the contact plan and flight plan corresponding to this solution are found in Table IV

## IV. Conclusions

LEO constellations of CubeSat nanosatellites present restrictions in the logistics associated with the deployment of a large constellation. It is impossible to assume that CubeSat constellations will be deployed in particular geometries such as Walker Star or Walker Delta, but in an Ad hoc configuration due to several secondary payload launches. Also, the constellation control problem's complexity suggests using metaheuristic algorithms and ground-based planning scale better when the number of nodes in the constellation increases.

Therefore an evolutionary algorithm was used to optimize the CPD under the particular restrictions of the study case.

TABLE IV
SCENARIO B, TASK 2 EXAMPLE CONTACT PLAN AND FLIGHT PLAN SOLUTION.

| **Contact plan** | | | |
|---|---|---|---|
| from | to | start | end |
| 16 | 2 | 1601428500 | 1601428800 |
| 2 | 14 | 1601431200 | 1601431500 |
| 2 | 5 | 1601432400 | 1601432700 |
| 5 | 8 | 1601432700 | 1601433000 |
| 8 | 18 | 1601433600 | 1601433900 |
| 8 | 16 | 1601436000 | 1601436300 |
| **Flight plan** | | | |
| Time | Node | Command | |
| 1601428500 | 16 | fp_send 2 | |
| 1601431200 | 2 | get_data data1 | |
| 1601432400 | 2 | fp_send 5 | |
| 1601432401 | 2 | send_data 5 data1 | |
| 1601432700 | 5 | fp_send 8 | |
| 1601432701 | 5 | send_data 5 data1 | |
| 1601433600 | 8 | get_data data2 | |
| 1601436000 | 8 | send_data 16 data1 | |
| 1601436001 | 8 | send_data 16 data2 | |
| 1601436002 | 8 | fp_send 16 | |

The resultant Contact Plan is then used to generate a global Flight Plan table that describes the operations that must be performed by the satellites in the constellation. The case study based on an Ad Hoc constellation of 10 satellites showed that it is possible to control the satellites using the flight plan table derived from the contact plan design.

The results of this work open several opportunities to explore. Scalability to larger constellations of hundreds to thousands of satellites is an important concern. It is relevant to know if the algorithm can provide feasible solutions in bounded time, even for a large number of nodes. Also, a real-life demonstration is important to validate ideas, so we expect the system to be available for the upcoming launches of the SUCHAI 2, 3, and PlantSat nanosatellites being developed at the University of Chile.

## REFERENCES


[1] Simon; Lee, Amy; Hutputanasin, Armen; Toorian, Wenschel; Lan, Riki; Munakata, Justin; Carnahan, David; Pignatelli, and Arash Mehrparvar. Cubesat design specification rev. 13. Technical Report 2, The CubeSat Program, Cal Poly San Luis Obispo, US, 2014.

[2] Deganit Paikowsky. What is new space? the changing ecosystem of global space activity. *New Space*, 5(2):84–88, 2017.

[3] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodriguez-Osorio, F. Pinto, and S. C. Burleigh. Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view. *IEEE Communications Surveys Tutorials*, 18(4):2442–2473, Fourthquarter 2016.

[4] Y. Su, Y. Liu, Y. Zhou, J. Yuan, H. Cao, and J. Shi. Broadband leo satellite communications: Architectures and key technologies. *IEEE Wireless Communications*, 26(2):55–61, 2019.

[5] Saptarshi Bandyopadhyay, Rebecca Foust, Giri P Subramanian, Soon-Jo Chung, and Fred Y Hadaegh. Review of formation flying and constellation missions using nanosatellites. *Journal of Spacecraft and Rockets*, (0):567–578, 2016.

[6] A. Marinan, A. Nicholas, and K. Cahoy. Ad hoc cubesat constellations: Secondary launch coverage and distribution. In *2013 IEEE Aerospace Conference*, pages 1–15, March 2013.

[7] Christopher Boshuizen, James Mason, Pete Klupar, and Shannon Spanhake. Results from the Planet Labs Flock Constellation. In *AIAA/USU Conference on Small Satellites*, aug 2014.

[8] Inigo [del Portillo], Bruce G. Cameron, and Edward F. Crawley. A technical comparison of three low earth orbit satellite constellation systems to provide global broadband. *Acta Astronautica*, 159:123 – 135, 2019.

[9] I. F. Akyildiz and A. Kak. The internet of space things/cubesats. *IEEE Network*, 33(5):212–218, 2019.

[10] J. Alvarez and B. Walls. Constellations, clusters, and communication technology: Expanding small satellite access to space. In *2016 IEEE Aerospace Conference*, pages 1–11, March 2016.

[11] Danilo JosÃ© Franzim Miranda, MaurÃcio Ferreira, Fabricio Kucinskis, and David McComas. A Comparative Survey on Flight Software Frameworks for TNew Space Nanosatellite Missions. *Journal of Aerospace Technology and Management*, 11, 00 2019.

[12] Andrew K. Kennedy and Kerri L. Cahoy. Performance analysis of algorithms for coordination of earth observation by cubesat constellations. *Journal of Aerospace Information Systems*, 14(8):451–471, 2017.

[13] Sreeja Nag, Alan S. Li, and James H. Merrick. Scheduling algorithms for rapid imaging using agile cubesat constellations. *Advances in Space Research*, 61(3):891 – 913, 2018.

[14] J. A. Fraire and J. M. Finochietto. Design challenges in contact plans for disruption-tolerant satellite networks. *IEEE Communications Magazine*, 53(5):163–169, May 2015.

[15] J. A. Fraire, G. Nies, C. Gerstacker, H. Hermanns, K. Bay, and M. Bisgaard. Battery-aware contact plan design for leo satellite constellations:the ulloriaq case study. *IEEE Transactions on Green Communications and Networking*, pages 1–1, 2019.

[16] Tomás Ferrer, Sandra Céspedes, and Alex Becerra. Review and evaluation of mac protocols for satellite iot systems using nanosatellites. *Sensors*, 19(8):1947, 2019.

[17] Zixuan Zheng, Jian Guo, and Eberhard Gill. "swarm satellite mission scheduling & planning using hybrid dynamic mutation genetic algorithm". *Acta Astronautica*, 137:243 – 253, 2017.

[18] Zixuan Zheng, Jian Guo, and Eberhard Gill. Onboard autonomous mission re-planning for multi-satellite system. *Acta Astronautica*, 145:28 – 43, 2018.

[19] Michel Lemaitre, Gérard Verfaillie, Frank Jouhaud, Jean-Michel Lachiver, and Nicolas Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6(5):367 – 381, 2002.

[20] J. A. Fraire, P. G. Madoery, and J. M. Finochietto. Traffic-aware contact plan design for disruption-tolerant space sensor networks. *Ad Hoc Networks*, 47:41–52, 9 2016.

[21] Carles Araguz, Marc Marí, Elisenda Bou-Balust, Eduard Alarcon, and Daniel Selva. Design Guidelines for General-Purpose Payload-Oriented Nanosatellite Software Architectures. *Journal of Aerospace Information Systems*, 15(3):107–119, mar 2018.

[22] Massimo Tipaldi, Cedric Legendre, Olliver Koopmann, Massimo Ferraguto, Ralf Wenker, and Gianni D'Angelo. Development strategies for the satellite flight software on-board Meteosat Third Generation. *Acta Astronautica*, 145:482–491, apr 2018.

[23] C. E. Gonzalez, C. J. Rojas, A. Bergel, and M. A. Diaz. An architecture-tracking approach to evaluate a modular and extensible flight software for cubesat nanosatellites. *IEEE Access*, pages 1–1, 2019.

[24] J. A. Fraire, Pablo G. Madoery, Jorge M. Finochietto, and Guillermo Leguizamón. An evolutionary approach towards contact plan design for disruption-tolerant satellite networks. *Applied Soft Computing Journal*, 52:446–456, 3 2017.