aerospace

Article

Automated Design of CubeSats using Evolutionary Algorithm for Trade Space Selection

Himangshu Kalita and Jekan Thangavelautham *

Space and Terrestrial Robotic Exploration (SpaceTREx) Laboratory, Aerospace and Mechanical Engineering Department, University of Arizona, Tucson, AZ 85721, USA; hkalita@email.arizona.edu

* Correspondence: jekan@email.arizona.edu

Received: 3 September 2020; Accepted: 25 September 2020; Published: 28 September 2020

Abstract: The miniaturization of electronics, sensors, and actuators has enabled the growing use of nanosatellites for earth observation, astrophysics, and even interplanetary missions. This rise of nanosatellites has led to the development of an inventory of modular, interchangeable commercially-off-the-shelf (COTS) components by a multitude of commercial vendors. As a result, the capability of combining subsystems in a compact platform has considerably advanced in the last decade. However, to ascertain these spacecraft's maximum capabilities in terms of mass, volume, and power, there is an important need to optimize their design. Current spacecraft design methods need engineering experience and judgements made by of a team of experts, which can be labor intensive and might lead to a sub-optimal design. In this work we present a compelling alternative approach using machine learning to identify near-optimal solutions to extend the capabilities of a design team. The approach enables automated design of a spacecraft that requires developing a virtual warehouse of components and specifying quantitative goals to produce a candidate design. The near-optimal solutions found through this approach would be a credible starting point for the design team that will need further study to determine their implementation feasibility.

Keywords: CubeSat; design optimization; evolutionary algorithm

1. Introduction

The development of commercially-off-the-shelf (COTS) components has led to the capability of developing compact spacecraft platforms by combining subsystems from the readily available COTS inventory. For these COTS assembled spacecrafts, the availability of rideshare opportunities on small spacecraft launchers and the prospect of being launched as a secondary payload on larger launch vehicles has led to the rapid advancement of COTS components and expansion of the small spacecraft market. As such, the buses for these small spacecrafts are designed as modular platforms and the components designed to be as diverse and capable as possible, which can be assembled with a payload in a short amount of time, ready to fly [1]. One such platform is the CubeSat platform, which weighs only a few kilograms and is based on a form factor of a $10 \times 10 \times 10$ cm^3. CubeSats can be composed of a single cube (a "1U" CubeSat) or several cubes combined, forming, for instance, 3U or 6U units [2]. Although the COTS components are readily available, designing these spacecrafts is a long, expensive endeavor, where a team of experts work together to design the system for a specific mission using their engineering judgement. An alternative design approach is required that shortens and simplifies the spacecraft design process. In this paper, we propose a tool for automated design of a spacecraft using machine learning algorithms that utilize modular, interchangeable components from a COTS inventory to develop a spacecraft design. In our approach, the spacecraft design is presented in the form of a gene that selects subsystem components from an inventory, and a numerical goal function is specified along with constraints, based on which the spacecraft design is evaluated. An evolutionary algorithm (EA) is used to generate a population of candidate solutions, which undergo evolutionary operations such as selection, crossover, and mutation, and the fittest individuals are carried to the next generation, while the unfit individuals are culled off. This method of evolution is performed over hundreds of generations until the fittest individual in the population meets a desired performance metric [3]. This approach can be very useful to a design team to make informed system design decisions by finding a set of near-optimal solutions that represent the trade-offs among conflicting objectives, such as mass, cost, and performance.

Automated design of engineering systems is not new. When compared to human created designs, computationally derived evolutionary designs have shown competitive advantages in terms of robustness, performance, and creativity. Design optimization through evolutionary computation has been studied for decades. Work has been done to design several satellite subsystems. One such example is an X-band antenna for NASA's Space Technology 5 (ST5) spacecraft, which is in fact the first computer-evolved hardware in space [4]. Other work includes design optimization of a power subsystem [5], low-thrust orbit transfer [6], spacecraft structure [7], and a corner retroreflector for satellites [8], which has shown potential improvements over traditional methods. An evolutionary approach has also been applied to design satellite constellations for continuous regional coverage [9], nontraditional constellations such as heterogeneous and rideshare constellations using COTS components to increase resiliency and responsiveness [10], reconfigurable satellite constellations [11], regional navigation satellite system constellations [12], pattern forming tasks using robot teams [13] and even teams of excavating robots for lunar base preparation [14], which have produced human-competitive designs. This process has also been successfully applied to the design of mobile robots [15] and water desalination systems [16]. Automated design is in sharp contrast to current spacecraft design methods. Current design methods use engineering experience and judgement of a team of experts to identify candidate designs. This process requires significant expertise and experience and is long and expensive in terms of time and labor. The lack of a systematic approach to fully evaluate the whole design space might lead to a sub-optimal solution or worse an intractable solution. Such an approach can also miss innovative designs not thought of by the human design team. Evolutionary design techniques however benefit from the exponential rise in computational speed and can overcome human team limitations, including number of experts available, time available to perform a design task, and time available to meet together and work continuously. A machine-automated approach can augment a human design team by being able to review a larger candidate of designs than would otherwise be possible and thus find better designs.

Our approach to the spacecraft design problem is modelled after the knapsack problem (KP). Optimization of the knapsack problem is considered an NP (non-deterministic polynomial-time) hard problem. NP is the set of decision problems in computational complexity theory that are solvable by a non-deterministic Turing machine in polynomial time. NP hard are the problems that are at least as hard as the hardest problems in NP and they are not necessarily decision problems [17]. In the knapsack problem, the goal is filling a knapsack with as many items so that the value of the items is maximized without exceeding the capacity of the knapsack. For a spacecraft, thanks to high launch costs, mass and volume are still a premium. In our spacecraft design problem, the focus is to effectively package components from a COTS inventory within a specified mass and volume constraint satisfying the standard CubeSat form factor. In our approach, the design of the spacecraft is specified by a gene that determines what components are to be packaged inside the spacecraft. This method of describing the design with genes keeps the process simple and produces results fast using desktop computers. In the following sections, we provide a brief description of the knapsack problem and the multidimensional multiple-choice knapsack problem which is the basis for our CubeSat design problem through trade space selection. Next, we present the optimization problem to design a 3U nadir-pointing, earth-observing CubeSat in low Earth orbit (LEO) followed by the models used for simulation. This is followed by the results and conclusions.

2. Knapsack Problem

The knapsack problem is analogous to the packing of items into a knapsack, where items are selected from an available set such that the total value of the selected items is maximized while simultaneously not exceeding the knapsack's capacity [18]. It is a highly popular problem in the research field of constrained and combinatorial optimization. The most common problem being solved is the 0-1 knapsack problem, which restricts the number of copies, x_i, of each kind of object, o_i, to zero or one. Given a set of n objects $O = (o_1, o_2, \ldots, o_n)$, each with a weight, w_i, and a value, v_i, along with a maximum weight capacity, W, of the knapsack, the objective is to find a subset, $S \subseteq O$, in such a way that the weight sum over the objects in S does not exceed the knapsack capacity and yields a maximum value. The problem is mathematically expressed as Equation (1).

$$KP = \begin{cases} \max \sum_{i=1}^{n} x_i v_i \\ \text{subject to } \sum_{i=1}^{n} x_i w_i \leq W \\ x_i \in \{0,\ 1\}\ \forall i \in \{1, \ldots,\ n\} \end{cases} \tag{1}$$

Here, x_i represents the number of instances of item i to include in the knapsack. Informally, the problem is to maximize the sum of the values of the items in the knapsack so that the sum of the weights is less than or equal to the knapsack's capacity. The problem is a reduction of real-world resource allocation problems with constraints. The 0–1 KP is weakly NP-hard, which means it can be solved exactly by utilizing pseudo-polynomial time algorithms based on dynamic programming. There are also other algorithms that can solve it a very short amount of time such as greedy algorithms and core branch and bound algorithms. However, the difficulty of the problem increases as the number of variables and correlation between them is increased.

3. Knapsack Problem in System Design

The knapsack problem is also relevant for designing systems consisting of multiple subsystems such that its performance is maximized satisfying multiple resource constraints. Considering a system to be composed of n subsystems with each subsystem, i, $\forall i \in \{1, \ldots,\ n\}$, consisting of l_i options, as shown in Figure 1, each option of the subsystem has a particular value and it requires m resources. The objective is to pick exactly one option from each subsystem for the maximum total value of the collected items, subject to m resource constraints of the knapsack (system).

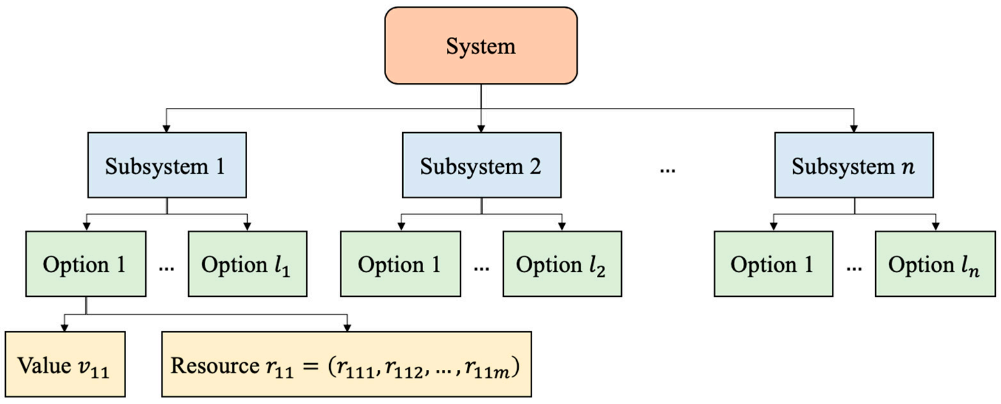


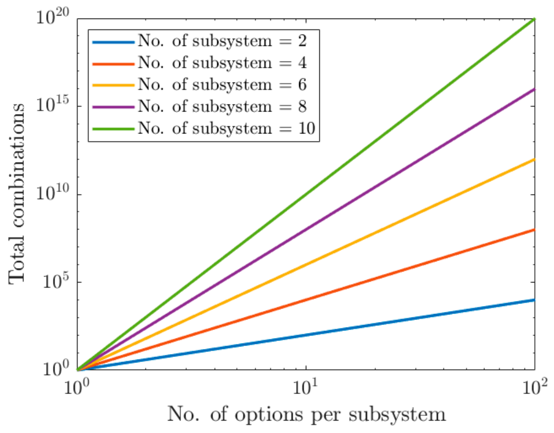
Figure 1. Description of a system composed of multiple subsystems and multiple options for each subsystem.

This is a variant of the classical 0-1 knapsack problem and is termed the multidimensional multiple-choice knapsack problem (MMKP) [19]. In mathematical notation, let v_{ij} and r_{ij} =

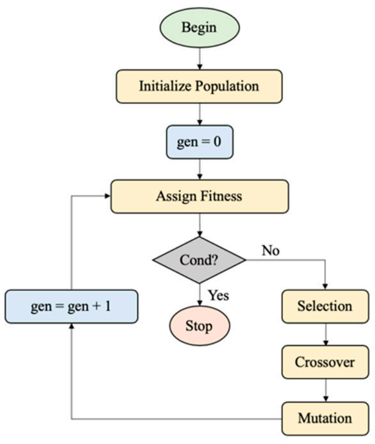
$\left(r_{ij1},\ r_{ij2}, \ldots,\ r_{ijm}\right)$ be the value and resource vector of the object o_{ij}, i.e., $j-$th option of the $j-$th subsystem. In addition, assume that $R = (R_1,\ R_2, \ldots,\ R_m)$ is the resource bound to the knapsack (system). Now, the problem is mathematically expressed as Equation (2).

$$MMKP = \begin{cases} \max \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{l_i} x_{ij} v_{ij} \\ \text{subject to } \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{l_i} x_{ij} r_{ijk} \le R_k\ \forall k \in \{1, \ldots,\ m\} \\ x_{ij} \in \{0,\ 1\} \\ \sum_{j=1}^{l_i} x_{ij} = 1\ \forall i \in \{1, \ldots,\ n\} \end{cases} \tag{2}$$

The number of possible designs, D, is the product of the number of options of all n subsystems; $D = \prod_{i=1}^{n} l_i$, which grows exponentially as the number of subsystems, n, and number of options, l_i, increases as shown in Figure 2a. One way to solve this problem is to use a brute force method by trying all D possible designs, which is highly inefficient. Other methods include greedy algorithms and dynamic programming [18]. Several heuristic algorithms have also been proposed for MMKP. One such algorithm is based on the Lagrange multiplier method [20]. Other methods include HEU [21], the modified guided local search [22], the modified reactive local search [23], the convex hull-based method [24], and the column generation method [25]. Several exact algorithms have also been proposed such as the branch-and-bound method with linear programming [21] and the EMKP algorithm [26]. Although these algorithms are able to obtain fairly good solutions for small search spaces, they become ineffective due to the *curse of dimensionality* when approaching practical scenarios, where the number of subsystems and the number of options for each subsystem grows significantly.



(**a**)



(**b**)

Figure 2. (**a**) Possible design combinations for varying number of components and options; (**b**) evolutionary algorithm structure [27].

However, EA provides a way to solve the knapsack problem in linear time. Evolutionary algorithms gain their inspiration from the natural process of evolution and emulate evolution by applying recombination (crossover), mutation, and natural selection on a population [27]. Highly fit individuals from the population are sampled, recombined, and resampled to form offspring of potentially higher fitness. As the evolutionary algorithm progresses through generations, fitter building blocks proliferate and unfit building blocks are culled from the population. In a way, by working with these particular schemata the complexity of the problem is reduced. Instead of building a high-performance individual by trying every conceivable combination, better and better individuals are constructed from the best partial solutions of past sampling.

Evolutionary algorithms (EAs) are a stochastic optimization method that provides an approach to learning by mimicking the metaphor of natural biological evolution or Darwinian evolution. By applying the principle of survival of the fittest, EAs progressively produce a better solution over generations by operating on a population of potential solutions. By mimicking the process of natural evolution, EAs perform evolutionary operations borrowed from natural genetics such as selection, crossover, and mutation. Theoretically, this method of evolving a population of individuals would progressively generate individuals that are better suited to their environment, just as in natural adaptation [28,29]. Figure 2b shows the structure of a simple evolutionary algorithm (EA). When successful, this approach will produce near-optimal solutions. These solutions when compared to solutions obtained through point design by a team of experts potentially presents major improvements. However, these solutions should be used as a starting point by the design team and would need further study to determine their implementation feasibility.

4. CubeSat Design

Current CubeSat design methods require a team of experts, who use their engineering expertise and decision to develop a spacecraft design from available COTS components. Such an approach may not result in an optimal design in terms of mass, volume, and power capabilities and can also miss innovative designs not thought of by the human design team. As such, the CubeSat design problem is modeled as a multidimensional multiple-choice knapsack problem. The goal is to effectively package subsystems from an inventory of COTS components within a specified mass and volume constraint such that the spacecraft can meet the defined mission goals.

As a test example, the objective is to design a 3U nadir-pointing, earth-observing CubeSat in LEO. The CubeSat is composed of many components that can be broken down into six major subsystems: (a) structure, (b) command and data handling (on-board computer), (c) communication (transceiver and antenna), (d) power (solar panels, battery pack, and power management board), (e) attitude determination and control unit (reaction wheels, magnetorquers, sun sensor, accelerometer, star tracker, magnetometer), and (f) payload (camera and lens) [1]. At first, an inventory of all the components is created from major CubeSat vendors (GomSpace, ISIS, Nanoracks, EnduroSat, DHV, NanoAvionics, Maryland Aerospace, Blue Canyon, Clydespace, Pumpkin, Space Micro, Rincon Research) which is used as a database for the Knapsack Problem. To solve the MMKP, an EA is used where a candidate population is evolved for hundreds of generations using evolutionary operators like selection, crossover, and mutation. Each individual in the population is defined by a gene, $\mathcal{G}$, which specifies what and how many components are to be packaged inside the spacecraft. This keeps the gene and design process simple and produces results fast using desktop computers. There are also other approaches to automated design, and these include the use of variable length generative coding schemes [28] that generate a constructive program to design the gene. Other bio-inspired approaches model morphogenesis [30].

An example gene structure is shown in Table 1. It consists of two parts: the first part is the subsystem specifier {sID, cID, aID, tID, bID, pID, adID, spID, caID} that selects components from the available database, and the second part selects the number of batteries $\{n_b\}$ and the number and location of solar panels: body panels {b1, b2, b3, b4}, side deployed panels {ds1, ds2, ds3, ds4}, and top deployed panels {dt1, dt2, dt3, dt4}. Figure 3 shows an example of a phenotype of the CubeSat rendered in MATLAB VRML. A script for automated assembly of all the components inside the structure is written such that each component is modeled as a cube with its maximum dimensions, and a spacing of 5 mm between components is provided for the wiring space requirement.

Table 1. Genotype of the CubeSat showing the structure of the gene: subsystem specifier; and specifier for number of batteries, and number and location of solar panels.

Structure	**OBC**	**Antenna**	**Transceiver**	**Battery**	**PMB**	**ADCS**	**Solar Panel**	**Camera**
sID	cID	aID	tID	bID	pID	adID	spID	caID
Integers $\begin{bmatrix} 1 & l_i \end{bmatrix}$: $l_i \rightarrow$ no. of options available								

No. of Batteries	**No. of Body Panels**				**No. of Deployed Panels (Side)**				**No. of Deployed Panels (Top)**			
n_b	b1	b2	b3	b4	ds1	ds2	ds3	ds4	dt1	dt2	dt3	dt4
Integers [1 10]	Integers [0 1]				Integers [0 3]							

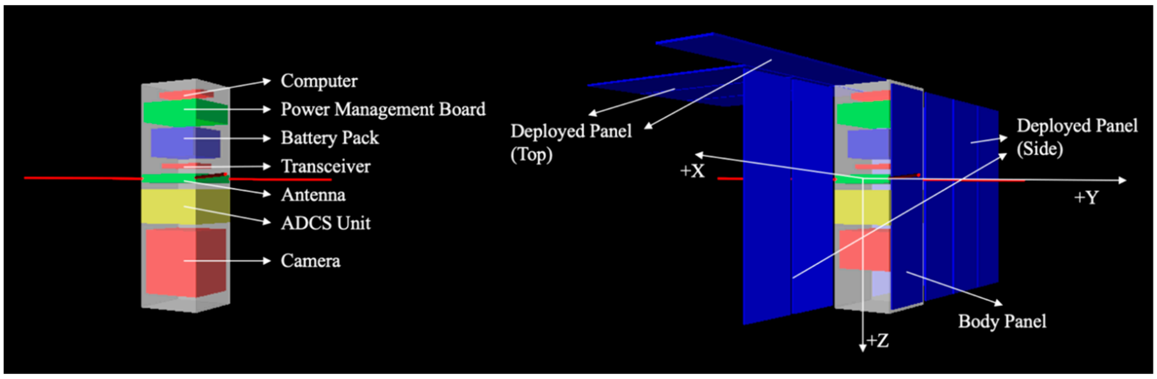


Figure 3. Example of a phenotype of the CubeSat genotype rendered in MATLAB VRML.

The design objective is to maximize the total downloadable ground area coverage and minimize the total mass and total cost of the satellite. For a selected camera from the inventory, the spatial resolution, r, is calculated as Equation (3).

$$r = \frac{ap}{f} \tag{3}$$

where a is the altitude of the satellite, f is the focal length of the lense, and p is the pixel pitch of the image sensor as shown in Figure 4. Considering the sensor having $p_H \times p_V$ number of pixels, the total ground area covered per image is given by $r^2 p_H p_V$ and the total size of each image is $bp_H p_V$, where b is the size of each pixel. If D is the total downloadable data, then the total downloadable ground area coverage, AC_D, is expressed as Equation (4).

$$AC_D = \frac{D}{bp_H p_V} r^2 p_H p_V = \frac{Dr^2}{b} \tag{4}$$

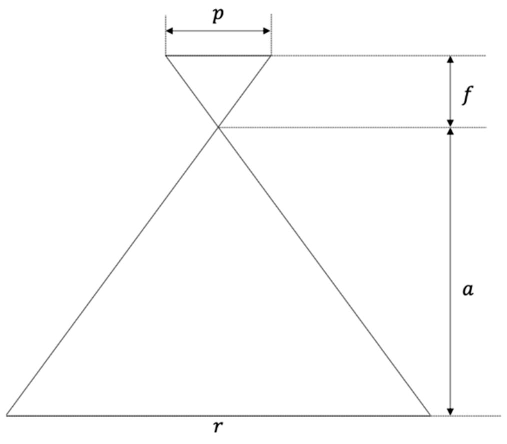


Figure 4. Schematic of spatial resolution of a camera for each pixel.

The mass and cost of each component is added to determine the total mass, m_T, and total cost, c_T, of the CubeSat, and thus, the objective function for the MMKP is expressed as Equation (5).

$$\min \frac{-AC_D}{m_T c_T} \tag{5}$$

To complete the knapsack problem, several constraints are added in terms of mass, volume, power and other properties such as the state of charge (SOC) of the battery subsystem, the pointing accuracy of the ADCS subsystem, the frequency and storage capacity of the on-board computer

(OBC), and the bandwidth of the antenna and transceiver. The first constraint is such that the total mass of the CubeSat is less than the maximum allowable mass of a 3U CubeSat (4 kg). For volume constraint, three constraints are added such that after all the components are stacked inside the structure the dimensions along x, y, and z direction (l_x, l_y, l_z) are within the 3U CubeSat standard ($10 \times 10 \times 34$ cm) [31]. For power, two constraints are added: the first is that the average solar power generated per orbit, P_{avg}, should be greater than the total power requirement of the CubeSat, P_T, (which is determined from each component selected), and the second is that the state of charge of the battery subsystem should not drop below a minimum value (user-defined). Other constraints are also added: ADCS—pointing accuracy is within a user-defined value, OBC—clock frequency, f_{OBC} and storage capacity are within some user-defined values, transceiver and antenna—bandwidth of antenna selected matches that of transceiver. The complete MMKP CubeSat design problem is then mathematically expressed as Equation (6).

$$\min_{\mathcal{G}} f(\mathcal{G}) = \frac{-AC_D}{m_T c_T}$$
$$\text{s.t.}\begin{cases} g_1(\mathcal{G}) \equiv m_T - 4 \leq 0 \text{ kg} \\ g_2(\mathcal{G}) \equiv l_x - 105 \leq 0 \text{ mm} \\ g_3(\mathcal{G}) \equiv l_y - 105 \leq 0 \text{ mm} \\ g_4(\mathcal{G}) \equiv l_z - 340 \leq 0 \text{ mm} \\ g_5(\mathcal{G}) \equiv P_{avg} \geq P_T \equiv P_T - P_{avg} \leq 0\, W \\ g_6(\mathcal{G}) \equiv SOC \geq 0.2 \equiv 0.2 - SOC \leq 0 \\ g_7(\mathcal{G}) \equiv Pointing\ Accuracy - 1^\circ \leq 0 \\ g_8(\mathcal{G}) \equiv f_{OBC} \geq 730 \equiv 730 - f_{OBC} \leq 0 \text{ MHz} \\ g_9(\mathcal{G}) \equiv Storage\ Capacity \geq 1.25 \equiv 1.25 - Storage\ Capacity \leq 0 \text{ Gb} \\ g_{10}(\mathcal{G}) \equiv f_{trans}^{(L)} + \frac{BW}{2} \leq f_{ant} \leq f_{trans}^{(U)} - \frac{BW}{2} \end{cases} \tag{6}$$

Now, to calculate the total downloadable data, D, average solar power generated, P_{avg}, state of charge of the batteries, SOC, and pointing accuracy of the ADCS system, a simulation environment is created that models orbit dynamics, attitude dynamics, solar power generation, battery state of charge, and communication disciplines as shown in Figure 5. Every individual of the population is an input to the COTS database that produces the CubeSat design. The design is then an input to the simulation environment, and then the outputs of the simulation environment along with mass, volume, power, and other properties of the design are used to determine the fitness (combination of objective function and the constraints). Details of the modeled disciplines are provided below.

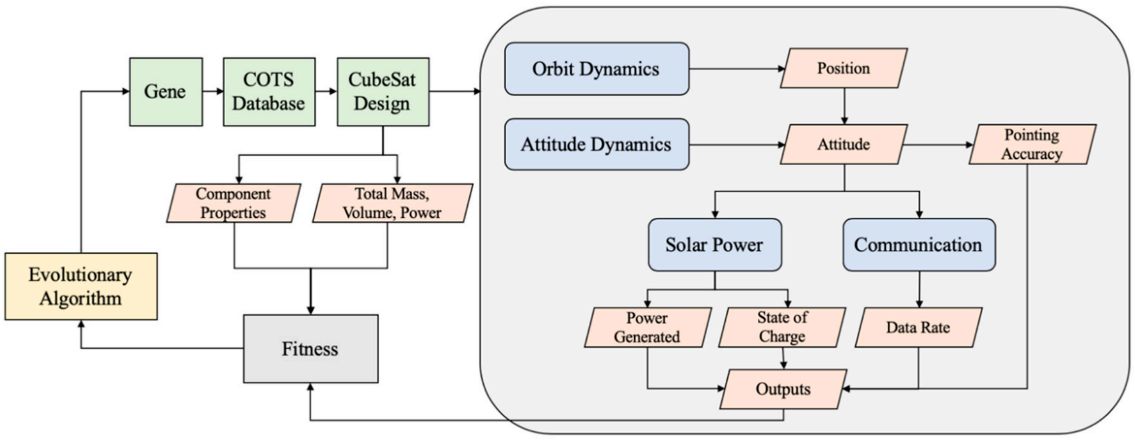


Figure 5. Simulation environment for CubeSat design optimization.

4.1. Orbit Dynamics

The orbit-dynamics discipline computes the Earth-to-body position vector of the satellite in the Earth-centered-inertial (ECI) frame. In the orbit equation, the J_2, J_3, and J_4 coefficients capture the fact that the Earth is not a perfectly spherical and homogeneous mass and is expressed by Equation (7).

$$\begin{gathered}\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} - \frac{3\mu J_2 R_e^2}{2r^5}\left[\left(1-\frac{5r_z^2}{r^2}\right)\vec{r} + 2r_z\hat{z}\right] - \frac{5\mu J_3 R_e^3}{2r^7}\left[\left(3r_z-\frac{7r_z^3}{r^2}\right)\vec{r} + \left(3r_z-\frac{3r^2}{5r_z}\right)r_z\hat{r}\right] + \\ \frac{15\mu J_4 R_e^4}{8r^7}\left[\left(1-\frac{14r_z^2}{r^2}+\frac{21r_z^4}{r^4}\right)\vec{r} + \left(4-\frac{28r_z^2}{3r^2}\right)r_z\hat{z}\right]\end{gathered} \tag{7}$$

where $\vec{r}$ is the position vector of the satellite in the ECI frame, μ is Earth's gravitational parameter, R_e is Earth's radius and J_2, J_3, and J_4 are orbit perturbation coefficients. The J_2, J_3, and J_4 terms must be considered because their effect is to rotate the orbit plane, which affects the power generation and communication discipline.

4.2. Attitude Dynamics

Since the CubeSat is an earth observing satellite, it must always maintain a nadir-pointing orientation. The attitude dynamics is simulated based on the reaction wheels/magnetorquers available in the selected ADCS system from the COTS inventory according to Equation (8).

$$J_s\dot{\vec{\omega}}_s = -\vec{\omega}_s \times \left(J_s\vec{\omega}_s + J_{RW}\vec{\omega}_{RW}\right) + \tau_{RW} + \tau_m + \tau_d \tag{8}$$

where J_s is the moment of inertia matrix of the satellite, $\vec{\omega}_s$ is the angular velocity vector of the satellite, J_{RW} is the moment of inertia matrix of the reaction wheel system, $\vec{\omega}_{RW}$ is the angular velocity vector of the reaction wheel system, τ_{RW} is the torque applied by the reaction wheels system, τ_m is the torque applied by the magnetorquers, and τ_d is the external disturbance torques, which is a sum of the gravity gradient, aerodynamic, and solar radiation pressure disturbances. The simulation of the attitude dynamics based on the selected ADCS system provides the pointing accuracy of the satellite, which is used for the optimization problem.

4.3. Solar Power Generation

For solar power, first the sun-to-satellite line-of-sight, LOS_s, is calculated which is essentially a multiplier for the solar panel exposed area to calculate the solar power generated. The LOS_s is 0 if the satellite is in the eclipse and 1 otherwise, however, to implement the umbra and penumbra effects, the discontinuous function is smoothed by using a cubic function [32], as shown in Equation (9).

$$LOS_s = \begin{cases} 1 & , \vec{r}_{b/e}\cdot\hat{r}_{s/e} \geq 0 \\ \left\{\begin{array}{ccc} 1 & , & d_s > R_e \\ 3\eta^2 - 2\eta^3 & , & \alpha R_e < d_s < R_e \\ 0 & , & d_s < \alpha R_e \end{array}\right\} & , \vec{r}_{b/e}\cdot\hat{r}_{s/e} < 0 \end{cases} \tag{9}$$

where d_s and η are given by Equation (10).

$$d_s = \|\vec{r}_{b/e} \times \hat{r}_{s/e}\|_2 \text{ and } \eta = \frac{d_s - \alpha R_e}{R_e - \alpha R_e} \tag{10}$$

The vectors $\vec{r}_{b/e}$ and $\hat{r}_{s/e}$ are shown in Figure 6. The value of α represents how far this smoothing effect extends into the Earth's shadow, typically $\alpha = 0.9$. Next, with the position of the sun known with respect to the satellite, the exposed solar panel area is required to calculate the total solar power generated. The exposed solar panel area is done in two steps. First for each panel having an area A_p and unit normal vector $\hat{n}_p$ and the body-to-sun unit vector $\hat{r}_{s/b}$, the projected area of each panel on a plane perpendicular to $\hat{r}_{s/b}$ is given by Equation (11) and shown in Figure 7a.

$$A_{proj} = \begin{cases} \hat{r}_{s/b} \cdot \hat{n}_p A_p & , \hat{r}_{s/b} \cdot \hat{n}_p > 0 \\ 0 & , \hat{r}_{s/b} \cdot \hat{n}_p \leq 0 \end{cases} \tag{11}$$

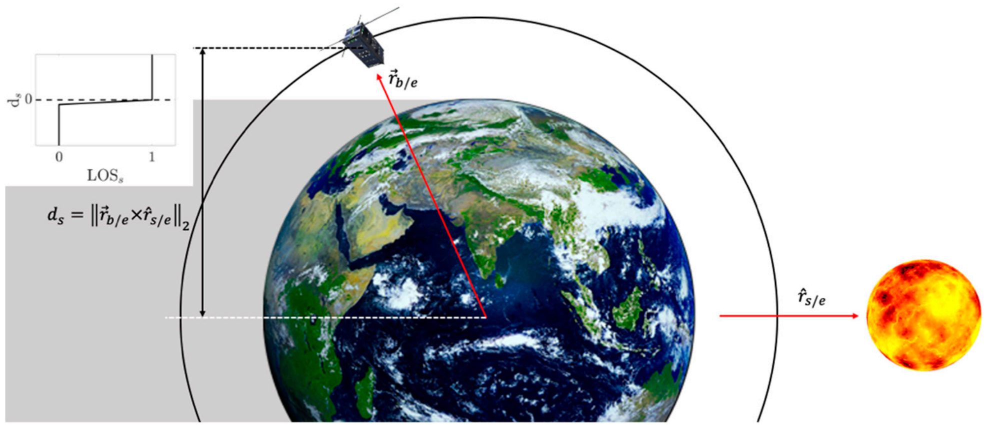


Figure 6. Illustration of Sun line-of-sight.

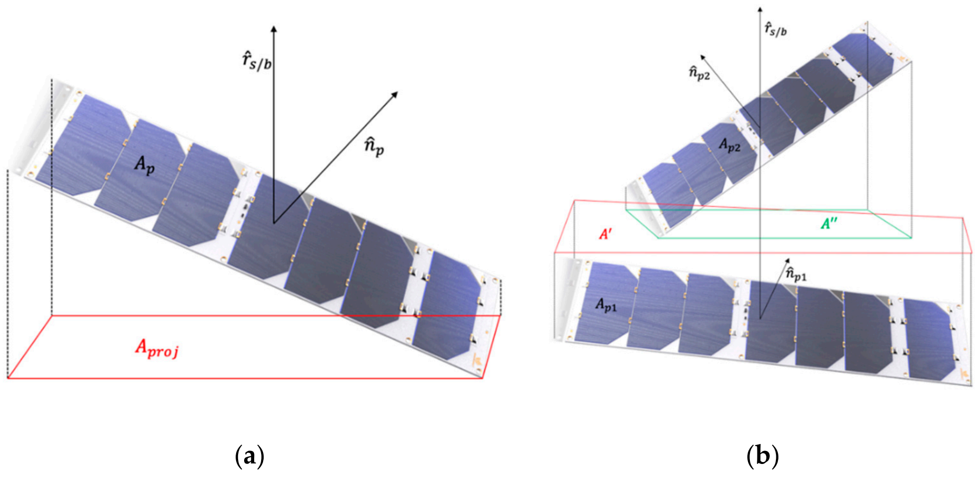


Figure 7. Solar panel projected area: (**a**) projected area of a single solar panel; (**b**) projected area of one panel onto another for shadow calculation.

Next, if the satellite consists of deployable solar panels, the effects of shadows projected by one panel onto another must be considered. For two panels {panel 1, panel 2} with areas $\{A_{p1}, A_{p2}\}$ and unit normal vectors $\{\hat{n}_{p1}, \hat{n}_{p2}\}$, and body-to-sun unit vector $\hat{r}_{s/b}$, the shadow cast by panel 2 on panel 1 is calculated by two successive projections and an intersection operator as shown in Figure 7b. The first projection is of panel 1 on a plane perpendicular to $\hat{r}_{s/b}$ given by $A' = \hat{r}_{s/b} \cdot \hat{n}_{p2} A_{p1}$, and the second projection is of panel 2 on a plane perpendicular to $\hat{r}_{s/b}$ given by $A'' = \hat{r}_{s/b} \cdot \hat{n}_{p1} A_{p2}$. Finally, the projected shadow cast by panel 2 on panel 1 on a plane perpendicular to $\hat{r}_{s/b}$ is the intersection of areas A' and A'', as shown by Equation (12).

$$A_{shad} = A_{p1} \cap A'' \tag{12}$$

Thus, the total exposed solar panel area is expressed as Equation (13).

$$A_{exp} = \sum_{i=1}^{n} \left(A_{(i)proj} - \sum_{j=1}^{k} A_{shad(j)} \right) \tag{13}$$

where n is the total number of solar panels, and $j = \{1, \ldots, k\}$ are the neighboring panels for panel i. Finally, the solar power generated is calculated as Equation (14).

$$P_{solar} = S_0 \varepsilon_{cell} A_{exp} \tag{14}$$

where S_0 is the solar constant, and ε_{cell} is the efficiency of the photovoltaic solar cells on each solar panel. Figure 8 shows the simulation results of the solar panel exposed area model for a CubeSat with two panels: one body panel and one deployed panel, as shown in Figure 8a. The simulation is performed in MATLAB, and the rendering of the CubeSat model shown in Figure 8a is done in MATLAB VRML. The body-to-sun unit vector $\hat{r}_{s/b}$ is represented in a spherical coordinate system with respect to the body frame, and as such, the results are presented against azimuth and elevation angles of the sun. Figure 8b shows the exposed area of panel 1 ignoring shadows cast by panel 2, while Figure 8c shows the projected shadow cast by panel 2 on panel 1, and finally, Figure 8d shows the total exposed area of panel 1. Our method is able to calculate the total exposed solar panel area for the above CubeSat with two panels, which is used for more complicated designs evolved through the optimization process.

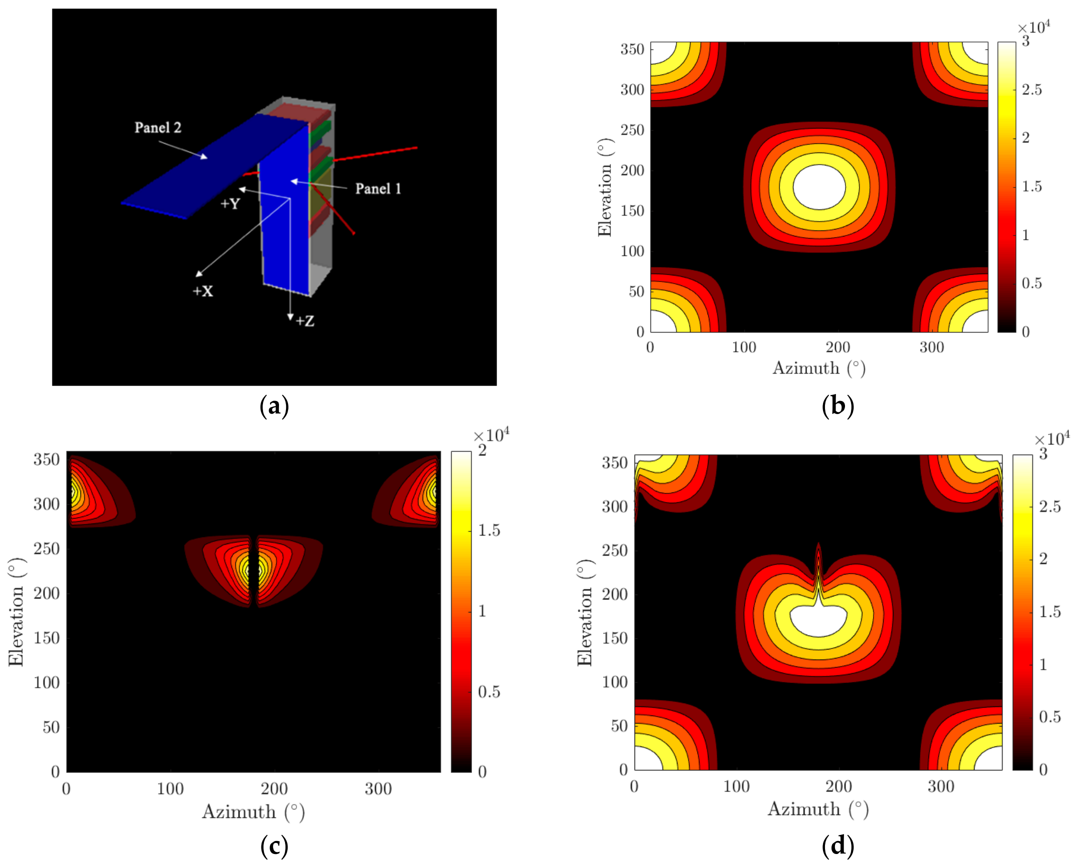


Figure 8. Solar panel exposed area (**a**) CubeSat model with two panels for simulation; (**b**) exposed area of panel 1 without panel 2; (**c**) projected shadow cast by panel 2 on panel 1; (**d**) total exposed area of panel 1. All areas are in mm^2.

4.4. Battery State of Charge

The state of charge of the battery system is determined by the Ordinary Differential Equation (ODE) (Equation (15)) [32].

$$S\dot{O}C = \frac{P_{bat}}{V_{bat}Q} \tag{15}$$

where $Q = n_b q$ is the nominal discharge capacity of the battery system with n_b being the number of batteries and q being the discharge capacity of each battery. The dependence of the battery voltage is on the SOC and also on the temperature, which is exponential, as shown by Equation (16).

$$V_{bat}(SOC) = \left(3 + \frac{e^{SOC} - 1}{e - 1}\right)\left(2 - e^{\lambda\frac{T-T_0}{T_0}}\right) \tag{16}$$

where T_0 is the reference temperature, and λ is the temperature decay coefficient. At any given time, the battery power is the sum of the loads, given by Equation (17).

$$P_{bat} = P_{sol} - P_{adcs} - P_{comm} - P_0 \tag{17}$$

where P_{sol} is the power generated by the solar panels, P_{adcs} is the power consumed by the ADCS system, P_{comm} is the communication power, and P_0 is the power consumed during the nominal operation of the CubeSat and the payload (camera).

4.5. Communication

For communication, first the ground station-to-satellite line-of-sight, LOS_c, is computed based on the dot product between the Earth-to-ground station unit vector and the ground station-to-body vector in the inertial frame, as shown in Figure 9. The discontinuous function is smoothed using a cubic function. Next, the resulting data download rate, B_r, is calculated using Equation (18) [32,33].

$$B_r = \frac{c^2 G_r \eta_p P_{comm} L_l G_t}{16\pi^2 f^2 k T_s (SNR) S^2} LOS_c \tag{18}$$

where c is the speed of light, k is the Boltzmann constant, f is the transmission frequency, T_s is the system noise temperature, G_r is the receiver (ground station) gain, SNR is the maximum acceptable signal to noise ratio, η_p is the communication efficiency, L_l is the line loss factor, P_{comm} is the communication power, G_t is the transmitter (satellite) gain, and S is the distance from the satellite to the ground station.

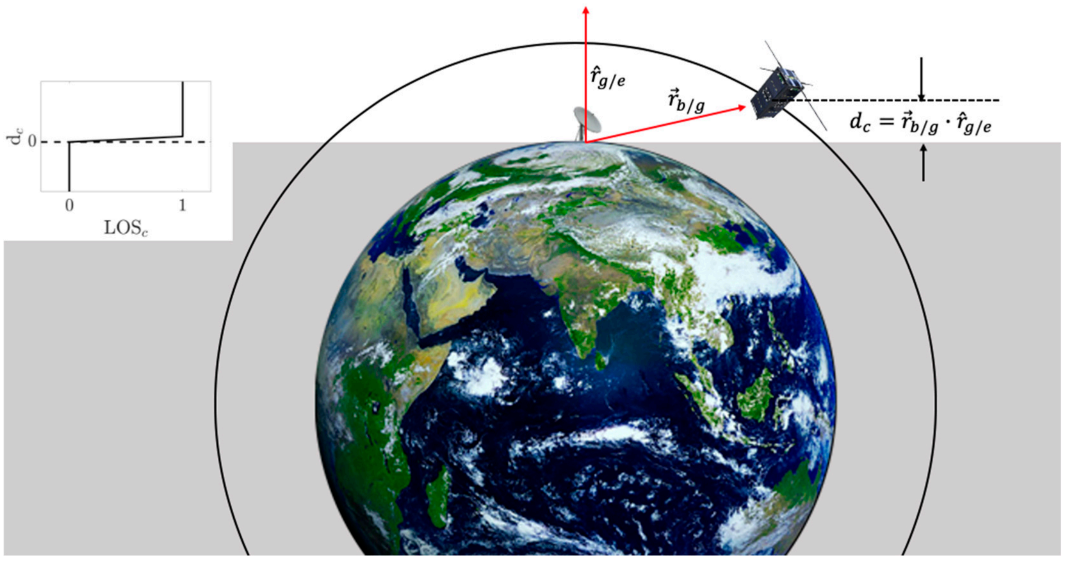


Figure 9. Illustration of communication line-of-sight.

4.6. Optimization

Now to solve the described CubeSat design problem implemented as a multidimensional multiple-choice knapsack problem, an evolutionary algorithm is used. The constraints described in Equation (6) divide the search space into two regions—feasible and infeasible regions. The constraints are handled using the penalty function approach. For each solution $\mathcal{G}^{(i)}$, i being the index of the gene, the constraint violation for the inequality constraints $\mathbb{G}_j\left(\mathcal{G}^{(i)}\right) \leq 0$ are calculated as in Equation (19).

$$\omega_j\left(\mathcal{G}^{(i)}\right) = \begin{cases} \left|\mathbb{G}_j\left(\mathcal{G}^{(i)}\right)\right|, \; if \; \mathbb{G}_j\left(\mathcal{G}^{(i)}\right) > 0 \\ 0 \; otherwise \end{cases} \tag{19}$$

Thereafter, all constraint violations are added together to get the overall constraint violation as in Equation (20). There are 10 constraints in the optimization problem and constraint 10 is decomposed into 2 inequality constraints.

$$\Omega\left(\mathcal{G}^{(i)}\right) = \sum_{j=1}^{11} \omega_j\left(\mathcal{G}^{(i)}\right) \tag{20}$$

This constraint violation is then multiplied with a penalty parameter, $\mathcal{P}$, and then the product is added to each of the objective function as in Equation (21).

$$\mathcal{F}\left(\mathcal{G}^{(i)}\right) = f\left(\mathcal{G}^{(i)}\right) + \mathcal{P}\Omega\left(\mathcal{G}^{(i)}\right) \tag{21}$$

The fitness function, $\mathcal{F}$, takes into account the constraint violations. For a feasible solution the corresponding Ω term is zero and $\mathcal{F}$ becomes equal to the original objective function f. However, for an infeasible solution, $\mathcal{F} > f$, thereby adding a penalty corresponding to the total constraint violation. Once the fitness function is derived, the steps for an EA are used to optimize the problem. Initially, a random parent population, P_0, is created of size N_P. Each individual of the parent population comprises of the design variables represented by the gene, $\mathcal{G}$, chosen by a discrete uniform distribution $\mathcal{G}_j = \mathcal{U}\left(\mathcal{G}_j^{(U)}, \mathcal{G}_j^{(L)}\right)$, where $\mathcal{G}_j$ is the j-th design variable and $\mathcal{G}_j^{(U)}$ and $\mathcal{G}_j^{(L)}$ are the upper and lower bounds of the design variables. For each individual, the value of the fitness function $\mathcal{F}(\mathcal{G})$ is evaluated using the simulation environment as shown in Figure 6. Next, the usual tournament selection, crossover, and mutation operators are used to create an offspring population, Q_0, of size N_Q [34]. For crossover, a blend crossover (BLX-α) operator is used [35]. During mutation, we use a non-uniform mutation operator. The mutation operator is constructed such that the probability of creating a solution closer to the parent is more than the probability of creating one away from it [36]. By comparing the current population with previously found best solutions, elitism is introduced into the algorithm. For the t^{th} generation, first a combined population $R_t = P_t + Q_t$ is formed. The population R_t is of size $N + N_Q$. Since all previous and current population members are included in R_t, elitism is automatically ensured. Next, the fitness of each individual in R_t is calculated according to the fitness function $\mathcal{F}(\mathcal{G})$. Based on the value of the fitness function, the population is sorted in ascending order and the best N individuals are selected for the next generation. The new population, P_{t+1}, of size N is again used for selection, crossover, and mutation to create a new population, Q_{t+1}, of size N_Q. The process is repeated until the desired results are obtained, or the max number of generations are achieved.

5. Results

For the test case the orbit of the satellite is simulated for an altitude of 400 km, inclination of 90°, eccentricity of 0 (circular orbit), RAAN of 90°, argument of perigee of 0°, and mean anomaly of 0°. The values of the constants used in the simulation are shown in Table 2. The optimization problem is simulated for 10 orbits in MATLAB, and the results are presented in Figures 10 and 11.

Table 2. Constants used in simulation.

Constant	Symbol	Value
Earth's gravitational parameter	μ	398,600.44 km^3·s^{-2}
Earth's radius	R_e	6378.137 km
Orbit perturbation coefficients	J_2	1.08264×10^{-3}
	J_3	-2.51×10^{-6}
	J_4	-1.60×10^{-6}
Solar constant	S_0	1367 W/m^2
Reference temperature	T_0	293 K
Temperature decay coefficient	λ	$\ln(1/(1.1)^5)$
Speed of light	c	2.99792458×10^8 m/s
Boltzmann constant	k	1.380648×10^{-23} m^{2}kg/(s^{2}K)
System noise temperature	T_s	500 K
Receiver gain	G_r	12 dB
Signal-to-noise ratio	SNR	5.0 dB
Communication efficiency	η_p	0.2
Line loss factor	L_l	−2.0 dB
Penalty parameter	$\mathcal{P}$	100
PSO tuning parameters	ω	1.1
	β	1.49
	γ	1.49
Annealing parameter	κ	5

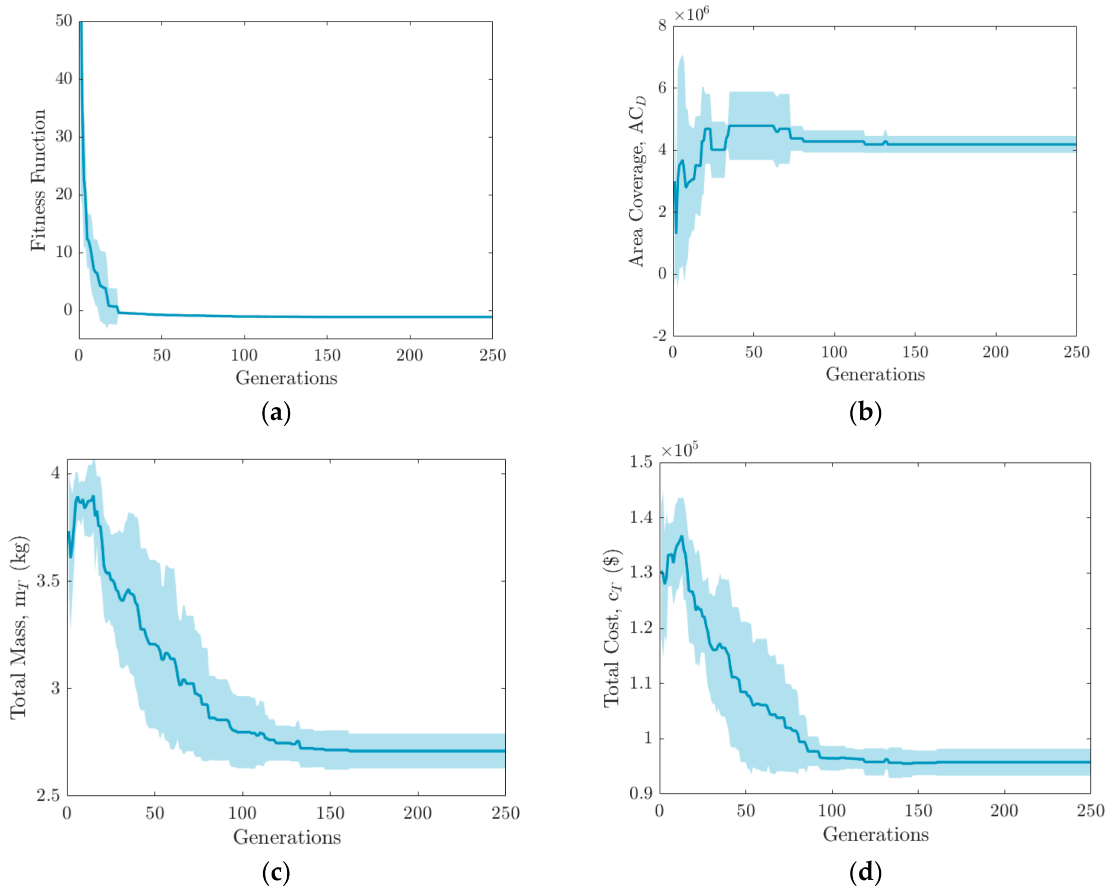


Figure 10. Performance of evolutionary algorithm on the CubeSat design multidimensional multiple-choice knapsack problem (MMKP): evolution of the (**a**) fitness function; (**b**) total downloadable ground area coverage; (**c**) total mass of the CubeSat; (**d**) total cost of the CubeSat, over generations along with the 95% confidence interval over 20 runs.

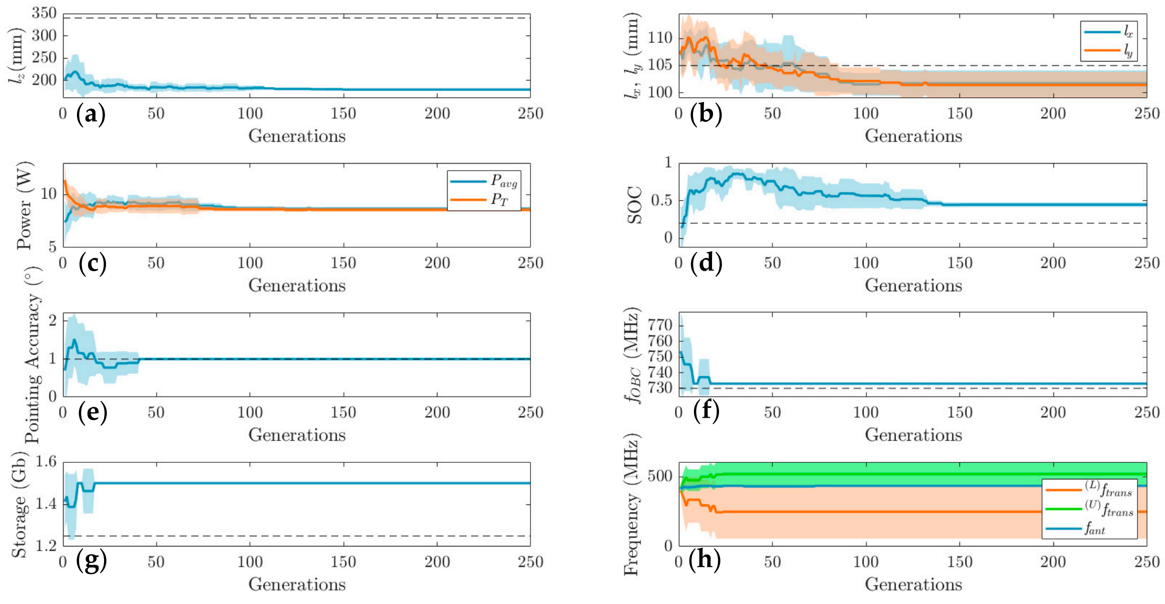


Figure 11. Evolution of the constraints g_2 to g_{10} for the CubeSat design MMKP.

Figure 10a shows the evolution of the fitness function $\mathcal{F}(\mathcal{G})$ over 250 generations with 100 populations in each generation. Figure 10b shows the evolution of the total downloadable ground area coverage, AC_D, Figure 10c shows that of the total mass, m_T, and Figure 10d shows that of the total cost, c_T, of the CubeSat over 250 generations. The shaded region of each of the plots show the 95% confidence interval over 20 runs. The results of the plots show that the fitness function decreased with generations converging to a constant fitness, while the total downloadable ground area coverage increased, and the total mass and total cost decreased with generations as implemented in the optimization problem.

Figure 11 shows the evolution of the constraints g_2 to g_{10}. The constraint g_1 is already presented in Figure 10c, which shows that the total mass is below the 4 kg allotted mass of a 3U CubeSat. Figure 11a,b shows the evolution of the dimension of the assembled CubeSat components along z, x, and y directions, showing it is within the 340 mm and 105 mm limit (shown by the dashed lines). Figure 11c shows the evolution of the average solar power generated per orbit, P_{avg}, and the total power requirement of the CubeSat, P_T, satisfying the requirement $P_{avg} \geq P_T$. Figure 11d shows the evolution of the state of charge, SOC, of the battery system with the dashed line showing the minimum requirement of 0.2. Figure 11e shows the evolution of the pointing accuracy of the ADCS system, which satisfies the constraint $Pointing\ Accuracy \leq 1°$. Figure 11f,g shows the evolution of the clock frequency and storage capacity of the on-board computer, which are greater than the specified values of 730 MHz and 1.25 Gb, respectively (shown by the dashed lines). Figure 11h shows the evolution of the antenna frequency (f_{ant}) along with the upper ($f^{(U)}_{trans}$) and lower ($f^{(L)}_{trans}$) limit of the frequency of the transceiver, showing the antenna frequency is within the limit of the bandwidth of the transceiver. It is clearly evident that all the constraints have been satisfied when the optimization process was terminated, and as such, the final solutions found are in the feasible region of the design space.

Figure 12 shows the rendering of the evolution of the best design over generations 1, 12, 15, 20, 30, 80, 100, and 150. It can be seen that the number of solar panels and their positions changed over generations, and at the same time, the internal components vertically stacked inside the structure also changed. Figures 13 and 14 show the variation of the design variables (represented by the gene in Figure 3) of the best individual over generations. The color of the design variables represents the component selected from the inventory in Figure 13 and the number of batteries and solar panels in Figure 14. It can be seen that the combination of the subsystems and the number of batteries and solar panels changed until generation 150, and then an optimal combination was found.

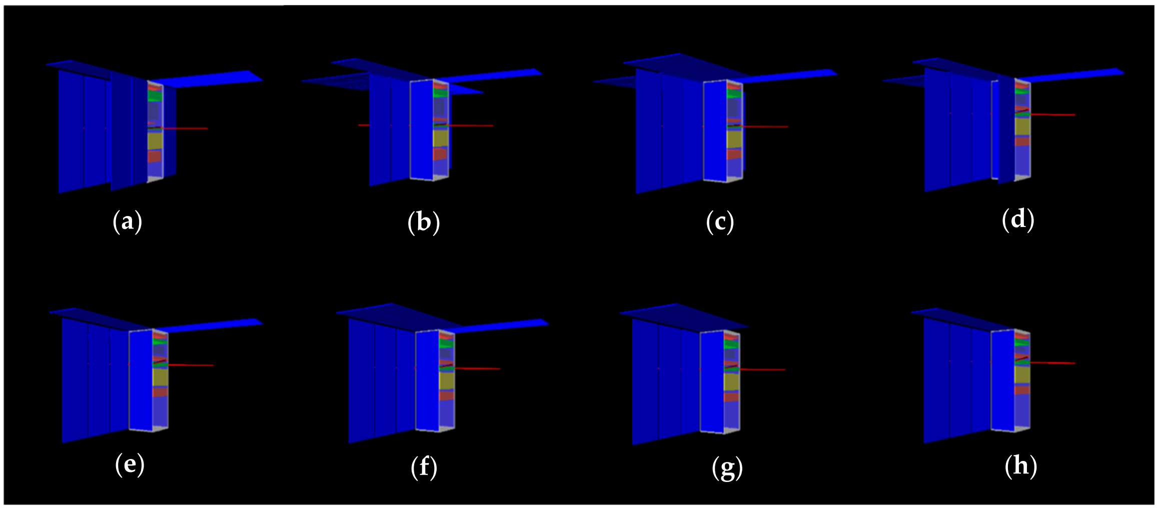


Figure 12. Rendering of the evolution of the design of the CubeSat done in MATLAB VRML. Each figure shows the best design among the population in each generation: (**a**) generation 1; (**b**) generation 12; (**c**) generation 15; (**d**) generation 20; (**e**) generation 30; (**f**) generation 80; (**g**) generation 100; (**h**) generation 150.

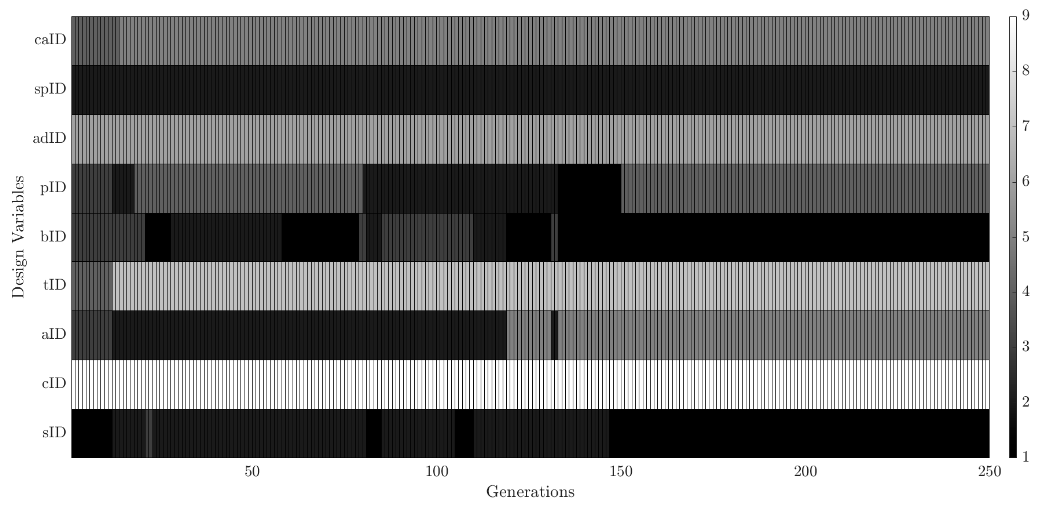


Figure 13. Variation of the design variables in the subsystem specifier over generations where the color denotes the component selected from the inventory.

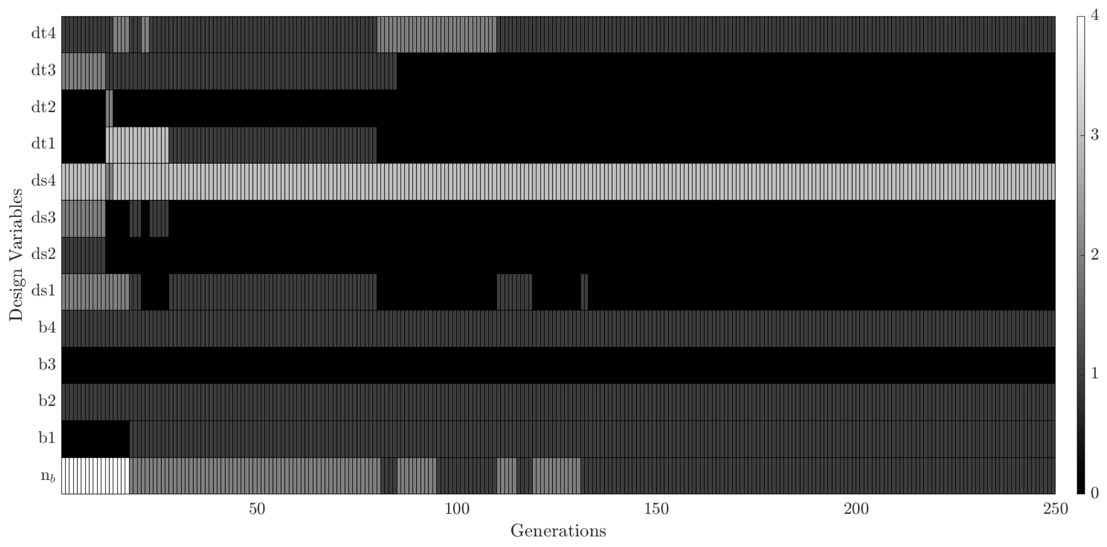


Figure 14. Variation of number of batteries and solar panels over generations where the color denotes the number.

Finally, we present a comparative analysis between three stochastic algorithms, evolutionary algorithm (EA), particle swarm optimization (PSO) [37], and simulated annealing (SA) [38], in terms of performance on the CubeSat design MMKP problem. The theory behind the application of the EA to our design problem has already been discussed in the preceding sections. We now provide a brief explanation on how PSO and SA are implemented.

5.1. Particle Swarm Optimization

Particle swarm optimization is a stochastic optimization method that is inspired from the social behavior of fish schooling or birds flocking. Just like EA, PSO is also initialized with a population (swarm) of individuals (particles). However, unlike EA, PSO does not use evolutionary operations to search for optima; instead, each particle is guided through the search space in every iteration by updating their positions and velocities using better positions found by other particles. In our CubeSat design problem, the fitness is defined by Equation (21) as $\mathcal{F}\left(\mathcal{G}^{(i)}\right)$, where $\mathcal{G}^{(i)}$ is the position of each particle, and the velocity of each particle is denoted by $\mathcal{V}^{(i)}$. The algorithm is initialized by generating N particles with their positions and velocities initialized randomly. Next, at every iteration, t, the velocity and position of each particle are updated by Equations (22) and (23), respectively.

$$\mathcal{V}_{t+1}^{(i)} = \omega \mathcal{V}_{t+1}^{(i)} + \beta r_1 \left(p_t^{(i)} - \mathcal{G}_t^{(i)} \right) + \gamma r_2 \left(p_t^{(g)} - \mathcal{G}_t^{(i)} \right) \tag{22}$$

$$\mathcal{G}_{t+1}^{(i)} = \mathcal{G}_t^{(i)} + \mathcal{V}_{t+1}^{(i)} \tag{23}$$

where ω, β, and γ are the tuning parameters, r_1 and r_2 are random numbers generated between 0 and 1, $p_t^{(i)}$ is the best remembered individual particle position, and $p_t^{(g)}$ is the best remembered swarm position. This process of updating the position and velocity of each particle continues until the desired fitness is achieved or maximum number of iterations reached.

5.2. Simulated Annealing

Simulated annealing is another stochastic optimization method that is inspired from annealing in metallurgy. The algorithm starts with a random trial solution, $\mathcal{G}$, whose fitness is determined by $\mathcal{F}(\mathcal{G})$, defined by Equation (21), and a large initial temperature, T_0. At each iteration, t, the algorithm randomly generates a new solution, $\mathcal{G}_{t+1}$, and its corresponding fitness, $\mathcal{F}(\mathcal{G}_{t+1})$, is calculated. If the new solution is better than the current solution, it is automatically selected as the next solution. However, if the new solution is worse, the algorithm accepts it as the next solution based on a probability of acceptance function defined by Equation (24).

$$P(\mathcal{G}_{t+1}) = \frac{1}{1 + \exp\left(\frac{\Delta}{\max(T)} \right)} \tag{24}$$

where $\Delta = \mathcal{F}(\mathcal{G}_{t+1}) - \mathcal{F}(\mathcal{G}_t)$ and T is the current temperature, which is systematically lowered by the algorithm at each iteration according to Equation (25).

$$T = \frac{T_0}{\ln(\kappa)} \tag{25}$$

where κ is the annealing parameter. Since the values of Δ and T are positive, larger Δ and a lower temperature leads to a smaller acceptance probability. The algorithm stores the best solution found so far and proceeds until the desired fitness is achieved or maximum number of iterations reached.

Figure 15a shows the evolution of the mean and standard deviation of fitness function over generations for all the 3 algorithms simulated and averaged over 20 times. The constant parameters used for PSO and SA are presented in Table 2. It can be clearly seen that the optimal solution found using simulated annealing lies in the infeasible region as the fitness function is positive. Both the evolutionary algorithm and particle swarm optimization were able to find a feasible solution, but the performance of the EA is better than PSO, as shown in Figure 15b, making it the better choice for this problem. Our observations show PSO stagnates prematurely, while EAs are more effective in continuing to innovate over generations.

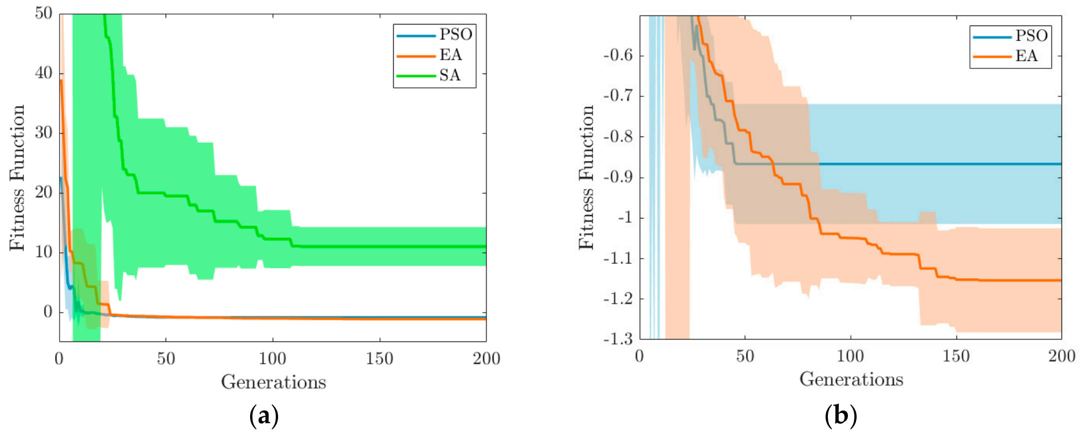


Figure 15. (**a**) Comparison of performance of evolutionary algorithm (EA) against particle swarm optimization (PSO) and simulated annealing (SA) in terms of average fitness and standard deviation over 20 runs. (**b**) Close up view of the comparison between EA against PSO.

6. Conclusions

This paper provides an analysis of the use of evolutionary algorithms to design complex systems that consist of multiple subsystems. It utilizes an inventory of components that are made available for each subsystem. The mathematical formulation for designing such a system is presented as a multidimensional multiple-choice knapsack problem (MMKP). The number of possible designs for a MMKP problem increases exponentially as the number of subsystems and number of options available for each subsystem increases. As such, the use of stochastic methods such as evolutionary algorithms (EAs) that mimic the metaphor of natural biological evolution provides a way to solve the problem in linear time complexity. A CubeSat design problem is presented as an example and an evolutionary algorithm is successfully applied in finding near-optimal designs. The approach has the potential for producing designs that maximize a mission-specific cost function while minimizing cost and mass. Traditionally, mission concepts are created by a team of engineers by conducting trade studies for mass, cost, volume, performance, and risk. Our approach of automated design using evolutionary algorithms for trade space selection speeds up the design process. Moreover, it provides us with a better basis for detailed system architecture, and hence, better design decisions can be made with confidence. This method generates a set of near-optimal designs, which provides multiple designs to start with and potentially presents vast improvements over a solution obtained through engineering judgement and point design alone. Our EA approach also shows improvement over other stochastic optimization methods including particle swarm optimization (PSO) and simulated annealing (SA). Although our approach shows a credible pathway to identify and evaluate many more candidate designs, these optimized designs will need further study to determine their implementation feasibility.

Author Contributions: The contributions to this article are the following: conceptualization, J.T. and H.K.; methodology, H.K. and J.T.; software, H.K.; validation, H.K.; formal analysis, H.K.; writing—original draft preparation, H.K.; writing—review, H.K. and J.T. and editing, J.T.; visualization, H.K.; supervision, J.T.; project administration, J.T.; funding acquisition, J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by National Aeronautics and Space Administration, grant number 80NSSC19M0197.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. State of the Art Spacecraft Technology. *Small Spacecraft Systems Virtual Institute*; NASA/TP-2018-220027; Ames Research Center: Mountain View, CA, USA, 2018.

2. Nugent, R.; Munakata, R.; Chin, A.; Coelho, R.; Puig-Suari, J. The CubeSat: The Picosatellite Standard for Research and Education. In Proceedings of the AIAA SPACE 2008 Conference & Exposition, San Diego, CA, USA, 9–11 September 2008.
3. Kalita, H.; Thangavelautham, J. Automated Design of CubeSats and Small Spacecrafts. In Proceedings of the 67th International Astronautical Congress, Guadalajara, Mexico, 26–30 September 2016.
4. Hornby, G.S.; Lohn, J.D.; Linden, D.S. Computer-Automated Evolution of an X-Band Antenna for NASA's Space Technology 5 Mission. *Evol. Comput.* **2011**, *19*, 1–23. [CrossRef] [PubMed]
5. Kordon, M.; Klimeck, G.; Hanks, D.; Hua, H. Evolutionary Computing for Spacecraft Power Subsystem Design Search and Optimization. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 6–13 March 2004.
6. Lee, S.; Allmen, P.; Fink, W.; Petropoulos, A.E.; Terrile, R.J. Design and Optimization of Low-thrust Orbit Transfers. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 5–12 March 2005.
7. Boudjemai, A.; Bouanane, M.H.; Merad, L.; Mohammed, A.M. Small Satellite Structural Optimization Using Genetic Algorithm Approach. In Proceedings of the 3rd International Conference on Recent Advances in Space Technologies, Istanbul, Turkey, 14–16 June 2007.
8. Minato, A.; Sugimoto, N. Design of a Four-element, Hollow-cube Corner Retroreflector for Satellites by Use of Genetic Algorithm. *Appl. Opt.* **1998**, *37*, 438–442. [CrossRef] [PubMed]
9. Meziane-Tani, I.; Metris, G.; Lion, G.; Deschamps, A.; Bendimerad, F.T.; Bekhti, M. Optimization of Small Satellite Constellation Design for Continuous Mutual Regional Coverage with Multi-objective Genetic Algorithm. *Int. J. Comput. Intell. Syst.* **2016**, *9*, 627–637. [CrossRef]
10. Wagner, K.M.; Black, J.T. Genetic-Algorithm-Based Design for Rideshare and Heterogeneous Constellations. *J. Spacecrcraft and Rocket.* **2020**, *57*, 1021–1032. [CrossRef]
11. Paek, S.W.; Kim, S.; Weck, O. Optimization of Reconfigurable Satellite Constellations Using Simulated Annealing and Genetic Algorithm. *Sensors* **2019**, *19*, 765. [CrossRef] [PubMed]
12. Ozdemir, H.I.; Raquet, J.F.; Lamont, G.B. Design of a Regional Navigation Satellite System Constellation Using Genetic Algorithms. In Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation, Savannah, GA, USA, 16–19 September 2008.
13. Thangavelautham, J.; D'Eleuterio, G.M.T. A Neuroevolutionary Approach to Emergent Task Decomposition. In Proceedings of the Parallel Problem Solving from Nature, Birmingham, UK, 18–22 September 2004.
14. Thangavelautham, J.; Abu El Samid, N.; Smith, A.D.S.; D'Eleuterio, G.M.T. Autonomous Multirobot Excavation for Lunar Applications. *Robotica* **2017**, *35*, 2330–2362. [CrossRef]
15. Farritor, S.; Dubowsky, S. On Modular Design of Field Robotic Systems. *Autonomous. Robot.* **2001**, *10*, 57–65. [CrossRef]
16. Bilton, A.M.; Dubowsky, S. A Computer Architecture for the Automatic Design of Modular Systems with Application to Photovoltaic Reverse Osmosis. *J. Mech. Des.* **2014**, *136*, 101401. [CrossRef]
17. Ladner, R. On the Structure of Polynomial Time Reducibility. *J. Assoc. Comput. Mach.* **1975**, *22*, 155–171. [CrossRef]
18. Kellerer, H.; Pferschy, U.; Pisinger, D. *Knapsack Problems*; Springer: New York, NY, USA, 2004.
19. Han, B.; Leblet, J.; Simon, G. Hard Multidimensional Multiple Choice Knapsack Problems, an Empirical Study. *Comput. Oper. Res.* **2010**, *37*, 172–182. [CrossRef]
20. Moser, M.; Jokanovic, D.P.; Shiratori, N. An algorithm for the multidimensional multiple-choice Knapsack Problem. *Ieice Trans. Fundam. Electron. Commun. Comput. Sci.* **1997**, *80*, 582–589.
21. Khan, S.; Li, K.F.; Manning, E.G.; Akbar, M.M. Solving the Kanpsack Problem for Adaptive Multimedia Systems. *Studia Inform. Univers.* **2002**, *2*, 157–178.
22. Hifi, M.; Michrafy, M.; Sbihi, A. Heuristic Algorithms for the Multiple-choice Multidimensional Knapsack Problem. *J. Oper. Res. Soc.* **2004**, *55*, 1323–1332. [CrossRef]
23. Hifi, M.; Michrafy, M.; Sbihi, A. A Reactive Local Search-based Algorithm for the Multiple-choice Multi-dimensional Knapsack Problem. *Computational. Optimization. Appl.* **2006**, *33*, 271–285. [CrossRef]
24. Akbar, M.M.; Rahman, M.S.; Kaykobad, M.; Manning, E.G.; Shoja, G.C. Solving the Multidimensional Multiple-choice Knapsack Problem by Constructing Convex Hulls. *Comput. Oper. Res.* **2006**, *33*, 1259–1273. [CrossRef]
25. Cherfi, N.; Hifi, M. A Column Generation Method for the Multiple-choice Multi-dimensional Knapsack Problem. *Comput. Optim. Appl.* **2010**, *46*, 51–73. [CrossRef]

26. Sbihi, A. A Best First Search Exact Algorithm for the Multiple-choice Multidimensional Knapsack Problem. *J. Comb. Optim.* **2007**, *13*, 337–351. [CrossRef]
27. Mitchell, T. *Machine Learning*; The McGraw-Hill Companies, Inc.: Boston, MA, USA, 1997.
28. Sims, K. Evolving 3D Morphology and Behavior by Competition. *Artif. Life* **1995**, *1*, 353–372.
29. Lipson, H.; Pollack, J.B. Automatic Design and Manufacture of Robotic Lifeforms. *Nature* **2000**, *406*, 974–978. [CrossRef]
30. Roggen, D.; Federici, D. Multi-cellular Development: Is there Scalability and Robustness to Gain? In Proceedings of the Parallel Problem Solving from Nature, Birmingham, UK, 18–22 September 2004.
31. *CubeSat Design Specification Rev. 13. The CubeSat Program*; California Polytechnic State University: San Luis Obispo, CA, USA, 2014.
32. Hwang, J.T.; Lee, D.Y.; Cutler, J.W.; Martins, J.R.R.A. Large-Scale Multidisciplinary Optimization of a Small Satellite's Design and Operation. *J. Spacecraft Rocket.* **2014**, *51*, 1648–1663. [CrossRef]
33. Larson, W.; Wertz, J. *Space Mission Analysis and Design*; Kluwer Academic: Norwell, MA, USA, 1991.
34. Goldberg, D.E.; Deb, K. A Comparison of Selection Schemes Used in Genetic Algorithms. *Found. Genet. Algorithms* **1991**, *1*, 69–93.
35. Eshelman, L.J.; Schaffer, J.D. Real-Coded Genetic Algorithms and Interval-Schemata. *Found. Genet. Algorithms* **1993**, *2*, 187–202.
36. Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed.; Springer: Berlin, Germany, 1996.
37. Mezura-Montes, E.; Coello, C.A.C. Constrained-handling in Nature-inspired Numerical Optimization: Past, Present and Future. *Swarm Evol. Comput.* **2011**, *1*, 173–194. [CrossRef]
38. Corana, A.; Marchesi, M.; Martini, C.; Ridella, S. Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm. *ACM Trans. Math. Softw.* **1987**, *13*, 262–280. [CrossRef]