applied sciences

Article

Two-Level Approach for Simultaneous Component Assignment and Layout Optimization with Applications to Spacecraft Optimal Layout

Juliette Gamot [1,2], Mathieu Balesdent [1,*], Romain Wuilbercq [1], Arnault Tremolet [1] and Nouredine Melab [2]

[1] ONERA, DTIS, Université Paris-Saclay, 91120 Palaiseau, France
[2] CNRS/CRIStAL, Centre Inria de l'Université de Lille, Université de Lille, 59491 Villeneuve d'Ascq, France

Abstract: Optimal layout problems consist in positioning a given number of components in order to minimize an objective function while satisfying geometrical or functional constraints. Such kinds of problems appear in the design process of aerospace systems such as satellite or spacecraft design. These problems are NP-hard, highly constrained and dimensional. This paper describes a two-stage algorithm combining a genetic algorithm and a quasi-physical approach based on a virtual-force system in order to solve multi-container optimal layout problems such as satellite modules. In the proposed approach, a genetic algorithm assigns the components to the containers while a quasi-physical algorithm based on a virtual-force system is developed for positioning the components in the assigned containers. The proposed algorithm is experimented and validated on the satellite module layout problem benchmark. Its global performance is compared with previous algorithms from the literature.

Keywords: optimal layout; assignment; satellite module layout problem; hybridization; virtual-force system; genetic algorithm

Citation: Gamot, J.; Balesdent, M.; Wuilbercq, R.; Tremolet, A.; Melab, N. Two-Level Approach for Simultaneous Component Assignment and Layout Optimization with Applications to Spacecraft Optimal Layout. *Appl. Sci.* **2024**, *14*, 8120. https://doi.org/10.3390/app14188120

Academic Editor: Kambiz Vafai

Received: 5 July 2024
Revised: 8 August 2024
Accepted: 12 August 2024
Published: 10 September 2024

1. Introduction

General Context

The design of aerospace engineering systems (e.g., launch vehicles, aircraft and satellites) encompasses multiple stages, from initial concept development to manufacturing. Each subsequent phase is composed of numerous analyses and optimizations. This enables designers to determine the most suitable design(s) with respect to the system's objective(s) under the hypotheses of the used models. Moreover, in realistic industrial settings, complex systems are often made up of numerous couplings between the several disciplines that are required for their design [1]. For instance, in the field of aerospace engineering, the preliminary sizing frequently integrates the coupling between vast fields such as aerodynamics, propulsion, structure and/or flight mechanics. While a number of those engineering fields are often tightly integrated at the early design stage, the design of the internal layout of future systems is, however, often set aside and is rarely part of a fully integrated design process.

The internal layout refers to the task of determining the most efficient and effective arrangement of internal components, structures or devices within the given designed space or system [2]. This problem seeks to optimize the spatial organization of these elements so as to achieve specific objectives, such as maximizing capacity, minimizing costs, improving accessibility or enhancing the overall system performance. In fact, internal layout problems are often solved by hand or by a set of simple heuristic rules (e.g., expert system) able to mimic the cognitive process of experts. Although this process is performed in the last steps of the design procedure, it does not guarantee that an overall optimal solution has been identified at preliminary design stages. Indeed, the arrangement of components often has a first-order impact on the performance of the system (e.g., for flying qualities of an aircraft and power efficiency of an electronic chip) and can thus be critical to the feasibility of a concept. Therefore, the primary focus revolves around the automated optimization of the internal layout of complex systems to ensure its compatibility within multidisciplinary design processes.

Optimal layout problems cover a large spectrum of applications: packing [3–5], facility layout [6–8], wind farms [9–11], coverage [12–14], complex systems layout problems [15–17], etc. This paper focuses on complex aerospace systems' layout problems. In the field of complex systems design, internal optimal layout problems [15–17] consist in optimizing the arrangement of components (e.g., various equipment, structure elements and electronics) within the internal structure or physical space of a complex system. The primary goal of addressing internal layout problems is to achieve an arrangement that optimizes various criteria, such as the flying qualities, operational efficiency, accessibility and workflow. Moreover, some geometrical and functional constraints must be satisfied, such as overlapping constraints, balancing constraints or proximity and distance between some components.

Most of the time, the set of components must be positioned within a single container [18]. In this case, the layout problem consists in finding the optimal positions and orientations of the components. However, layout problems can also involve positioning the set of components within several containers [19]. Consequently, the layout problem transforms into the task of assigning each component to a specific container while concurrently identifying the optimal arrangement for the assigned containers. Considering the multi-container configuration gives rise to additional challenges as it complicates the single-container optimal layout problem's design search space in terms of both dimensionality and combinatorics.

Thus, this paper focuses on solving multi-container optimal layout problems. As an illustrative benchmark, the optimal layout of a multi-container satellite module is considered. More precisely, the simplified model of the telecommunication satellite INTELSAT-III is employed [19–21]. This application case has been widely studied in the literature and constitutes a representative optimal layout problem. Figure 1 shows different views of the satellite module. The layout problem is mostly referred to as a Satellite Module Layout Problem (SMLP) in the literature. This involves determining the optimal assignment scheme for components within the module's containers as well as identifying the optimal positions and orientations of these components within the container. The objective is to minimize the overall inertia of the system. The following geometrical constraints have to be satisfied: non-overlapping constraint, balancing constraints and angle of inertia constraints. Functional constraints must also be satisfied in order to keep some components apart from a given distance (e.g., to ensure to satisfy specifications about the radiative environment).

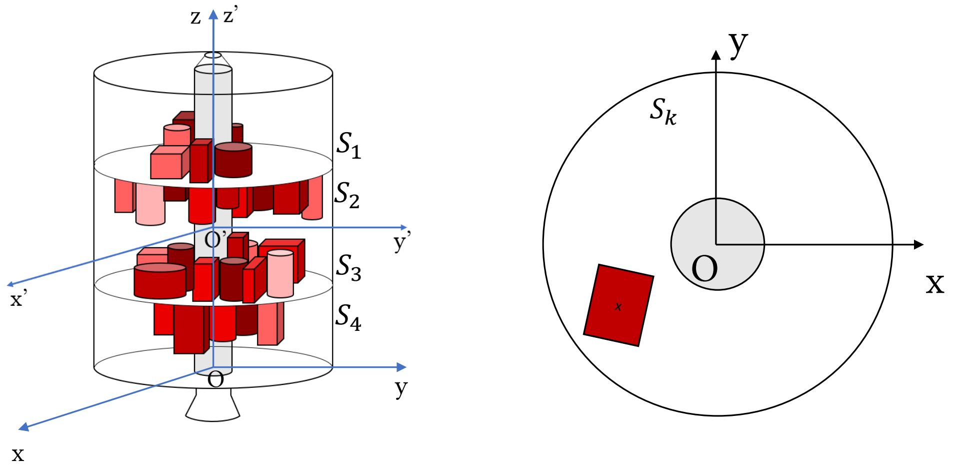


Figure 1. Views of the simplified model of the INTELSAT-III satellite module, as detailed in [20]. The redder the components, the heavier their corresponding masses.

Over the past twenty years, many approaches have been developed to deal with optimal layout problems. The first range of techniques are exact methods that guarantee the optimality of the solution. Among them, the CPLEX solver [22], the Branch and Bound algorithm [23–25] and Dynamic Programming [26–28] have been used to solve Quadratic Assignment Problem [29] and Mixed Integer Linear Programming [30] formulations.

Heuristics have then been developed in order to provide good quality results in a more reasonable computational time in comparison to exact methods. They mainly consist of construction [31–33] and improvement-based algorithms [34–36].

The most used approaches employed for optimal layout problems involve the utilization of metaheuristic algorithms. These problem-independent techniques are designed to explore the search space more comprehensively compared to previously employed heuristic methods, often yielding better solutions. Among them, Simulated Annealing [37–40], Tabu Search [41–43], Genetic Algorithm [16,44–46], Particle Swarm Optimization [47–49], Covariance Matrix Adaptation–Evolution Strategy [47,50], etc. have been used to solve various forms of optimal layout problems. Those techniques have also been hybridized between themselves [51–53] or with heuristic methods [17,54] in order to take advantage of the strengths of different algorithms and improve their overall computational performance.

Another range of techniques are quasi-physical methods, which have their origins in both physics and mathematics and aim at mimicking the laws of physics. Most of the time, they correspond to energy [55] or force-based algorithms [3,56,57].

Eventually, machine learning methods have been applied to layout optimization problems, although their exploration and investigation in this domain are still relatively limited at present [58]. For instance, Bayesian Optimization [59], Neural Networks [60] or Reinforcement Learning [61] have been used to solve optimal layout problems.

In the literature, mostly single-container optimal layout problems are solved. Moreover, considering the SMLP, very few studies include the optimization of the components' assignment. Most of the time, the components are allocated to the containers with an assignment scheme based on human knowledge and a method is derived to solely solve the layout of the assigned containers [20]. Finally, the existing approaches are mostly based on metaheuristic methods. However, it has been highlighted in [62–64] that metaheuristic techniques may present some weaknesses for solving problems with complex combinatorial spaces or a large number of constraints. In that respect, this paper aims to develop a two-stage algorithm combining a genetic algorithm and a quasi-physical approach for component assignment and layout optimization. In this approach, a genetic algorithm is devoted to the assignment task at the upper level of the algorithm while a quasi-physical algorithm based on a virtual-force system is derived to optimize the layout of the assigned containers.

The rest of the paper is organized as follows: Section 2 reviews the existing methods for optimal layout problems with a focus on the SMLP. Section 3 introduces the mathematical formulation of the multi-container SMLP. In Section 4, the proposed algorithm is described. Finally, the proposed approach is applied in Section 5 to the multi-container SMLP and the results are analyzed and compared to classical methods with previously published results.

2. State of the Art

This section is devoted to the review of existing methods for solving optimal layout problems. They are usually classified into different categories: exact methods, heuristic methods, metaheuristics, hybrid approaches, quasi-physical methods and machine learning techniques.

2.1. Methods for Optimal Layout Problems

2.1.1. Exact Methods

The exact methods stand for methods that ensure the optimality of the solution. In that respect, many exact methods have been deployed in order to solve the mathematical models developed to define several optimal layout problems. The very first approach to solve layout problems is the Quadratic Assignment Problem (QAP) model, first introduced in [29]. QAP is developed in order to find the optimal positions of equal-area components on discrete locations [65,66]. An alternative way to formulate layout problems is Mixed Integer Linear Programming (MILP), first introduced in [30]. The MILP model uses a continuous/discrete representation and is thus more representative of the layout problem structure than the QAP model [67–69]. Branch and Bounds (B&B) resolution methods have also been used in order to solve the aforementioned mathematical formulations and have been successfully applied to optimal layout problems [23–25]. Finally, Dynamic Programming techniques have been used in order to solve instance cutting or packing problems [26–28].

2.1.2. Heuristic Methods

In order to obtain results in a reasonable computational budget in comparison to exact methods, a lot of research has focused on heuristic methods. Heuristic algorithms, while lacking a guarantee of optimality, are dedicated to the search for interesting solutions. One of the oldest heuristics is the Construction Method. The principle of construction-type heuristics is to build a single solution from scratch by iteratively selecting and locating the items until the layout is complete. Two representative construction-based heuristic algorithms are the CORELAP [70] and the ALDEP algorithms (Automated Layout Design Program) [31–33]. Another type of heuristic mechanisms corresponds to the Improvement algorithms, which start with an initial feasible layout and then try to improve it by exchanging items. The most widely used improvement-based heuristic procedure is the CRAFT (Computerized Relative Allocation of Facilities Technique) algorithm, which has been successfully applied to many different optimal layout problems [35,36,71]. Certain studies have integrated both construction and improvement heuristics as in [72–74] through the development and application of a greedy randomized adaptive search procedure (GRASP) to various optimal layout packing problems.

2.1.3. Metaheuristics

Metaheuristics are problem-independent techniques that aim at exploring the search space more than heuristic methods and, thus, often provide more interesting solutions. Consequently, during the past 20 years, mostly metaheuristic algorithms have been used in order to solve optimal layout problems.

In [75], Singh et al. reviewed methods that have been applied to facility optimal layout problems, which are mainly metaheuristic techniques. Among them, Simulated Annealing (SA) [37–40], Tabu Search (TS) [41–43], Variable Neighborhood Search (VNS) [76–78], Genetic Algorithms (GA) [15,18,79,80], Particle Swarm Optimization (PSO) [48,49], Differential Evolution [11,81], Covariance Matrix Adaptation–Evolution Strategy [47,50], Ant Colony Optimization [82,83], Scatter Search [84,85] and Quality–Diversity algorithms [86–88] have been successfully applied to various optimal layout problems.

2.1.4. Hybrid Approaches

With the increase in the number of design objectives, constraints or design variables, other techniques have been proposed in order to improve the computational performance of the previously mentioned algorithms. Among them, cooperative co-evolutionary algorithms (CCEAs), first introduced by Potter and De Jong [89], based on the divide-and-conquer method along with the biological model of co-evolution of cooperating species have been developed. CCEAs allow users to tackle a complex optimization problem by decomposing it into multiple subproblems, each solved with a separate evolutionary algorithm. A complete problem solution is constructed by merging the representative members of each subpopulation (associated to one subproblem) through cooperative co-evolution mechanisms. Ma et al. recently proposed a survey on CCEAs [90]. Those algorithms were successfully applied to optimal layout problems [19,21,91]. Metaheuristics have also often been hybridized between themselves, as in [52,53,70]. Finally, those techniques have been hybridized with exact methods and heuristic techniques in [17,54,92].

2.1.5. Quasi-Physical Methods

Quasi-physical methods are techniques that have both physical and mathematical roots and aim to mimic physical laws. Most of the time, they correspond to energy- or force-based algorithms. Among them, local search strategies based on an energetic and physical modeling have been used to solve various packing problems [55,56,93]. The Gravitational Search algorithm (GSA) can also belong to this category as it is based on the law of gravity and mass interactions. It has been successfully applied to optimal layout problems [94,95]. Finally, one other quasi-physical approach is referred to as the virtual-force method and consists in applying forces to the items in order to induce virtual movements thanks to dynamic physical laws. Virtual-force systems-based methods have mostly been applied to coverage and packing problems, where elastic models resulting in virtual forces are applied to the items' positions in order to satisfy the overlapping or balancing constraints [3,56,57].

2.1.6. Machine Learning Techniques

Recently, machine learning methods have been applied to layout optimization problems even if they are less investigated. Burggräf et al. conducted a study on the use of machine learning as resolution techniques for facility layout problems in [58]. Among machine learning techniques, Neural Networks [60], Reinforcement Learning [61,96] and Bayesian Optimization [59,97,98] have been explored for solving optimal layout problems.

2.2. Layout Problem for Aerospace Systems

The creation of a new spacecraft involves a series of steps, beginning with conceptual studies and culminating in its manufacturing [99]. At the conceptual stage, the focus is on evaluating different possible designs for the spacecraft's electrical and mechanical systems to select one that meets the mission's goals. Therefore, the optimal layout of aerospace systems is crucial, as it significantly influences the overall performance of the system [100].

The pioneering works by Ferebbe et al. [101,102] are among the earliest to propose numerical optimization methods for automating the equipment layout determination process in the conceptual phase of spacecraft design.

Several papers have focused on optimizing the structural layout of aerospace systems [103–106]. Moreover, a series of studies [16,20,107,108] investigated the effectiveness of various numerical methods when applied to the optimization of the locations of a given set of components within telecommunication satellites. These studies primarily focused on positioning equipment based on the spacecraft's mass properties, such as the position of the mass center and the magnitude and direction of the principal axis of inertia, while adhering to geometric constraints. In [109–111], thermal considerations and the minimization of wiring between equipment were introduced as additional objectives in the exploration of candidate solutions within the design space.

2.3. Focus on Satellite Module Layout Problems

The application case considered in this paper is the Satellite Module Layout Problem (SMLP) [19,21,112]. The SMLP has been studied for the past 20 years and is considered as a representative optimal layout problem for the aerospace industry. Mainly, metaheuristic techniques have been developed in order to deal with this benchmark, sometimes assisted by other methods.

Grignon et al. [113] adopted a Pareto genetic algorithm for optimizing the compactness, assembly and maintenance of a small SMLP composed of seven components and a single-container. Shafaee et al. [114] proposed a hybrid algorithm combining a genetic algorithm and sequential quadratic programming (SQP) to solve the layout of spacecraft propellant systems comprising six components.

Those studies involved less than a dozen components. With the increase in the number of components, objective functions and constraints, cooperative co-evolutionary and hybrid algorithms were developed in order to deal with up to 60 components. Teng et al. developed a dual-system cooperative co-evolutionary algorithm (named Oboe-CCEA) for the SMLP based on both the cooperative co-evolutionary framework and the variable-grain model [20]. Li et al. proposed a hybrid multi-mechanism optimization approach (HMMOA) for the satellite layout optimization problem [115]. Cui et al. developed a new dual-system cooperative co-evolutionary algorithm for the SMLP based on the cooperative co-evolutionary framework [112].

The previous studies dealt with multi-container configurations of the SMLP, as described in Figure 1. However, the assignment of the components to the containers was fixed and determined thanks to human expert knowledge. The mentioned methods only focused on solving the layout of the components in the four containers. Some recent studies optimized the assignment of the components in addition to their layout. Xu et al. proposed an integrated assignment and layout method for an SMLP [116]. This approach adopts Tabu Search and Differential Evolution to solve the component assignment and layout optimization problems. However, the number of components was limited to 19. Cui et al. proposed a two-stage algorithm called Dynamic FS to solve both assignment and layout tasks of the multi-container SMLP [19]. Heuristic rules are used to assign the components. They are mathematically translated into a multi-objective optimization problem and an NSGA-II algorithm is employed to optimize it. The assignment list is then determined using a fuzzy decision-making method. The layout task is performed using the NDCCDE/DPSO algorithm from [112], which corresponds to a dual-system cooperative co-evolutionary algorithm.

Most of the previous methods used to tackle optimal layout problems and, more particularly, Satellite Module Layout Problems rely on metaheuristic frameworks. Metaheuristic algorithms provide various frameworks that can be easily adapted to the problems at hand and benefit from a large number of references in the literature. However, it has been highlighted in [62–64] that pure metaheuristic techniques may present some weaknesses for solving problems with complex combinatorial spaces or a large number of constraints. Moreover, a small number of studies tackle both assignment and layout tasks of the considered SMLP benchmark [19].

Consequently, in this paper, a two-stage algorithm combining the Genetic Algorithm for optimizing the assignment and a quasi-physical approach for optimizing the layout of multi-container optimal layout problems is proposed. It is applied to the SMLP as a benchmark.

3. Formulation of the Satellite Module Layout Problems

3.1. Geometry of the Containers and the Components

The satellite module corresponds to two bearing plates defined by their radius R_{out} and their thickness H_p. They are disposed at the heights H_1 and H_2 in a cylindrical module of height H_3. Thus, four surfaces S_k, $k \in \{1,2,3,4\}$ are laid out. A central column of radius R_{in} results in circular exclusion zones centered at the z axis on each surface. The whole module is characterized by its mass m_{sat} and its geometrical inertia along each axis—$I_{x,sat}$, $I_{y,sat}$ and $I_{z,sat}$—resulting in a global inertia defined as $I_{sat} = I_{x,sat} + I_{y,sat} + I_{z,sat}$. The center of gravity of the empty module is positioned at $(x_{sat}, y_{sat}, z_{sat})$. Figure 2 illustrates the multi-container configuration.

$N = 60$ components are to be positioned, divided into N_{cub} cuboids and N_{cyl} cylinders, as proposed in [20]. They are listed in Table A1 in Appendix A alongside the numerical values of the dimensions and dynamical features of the container.

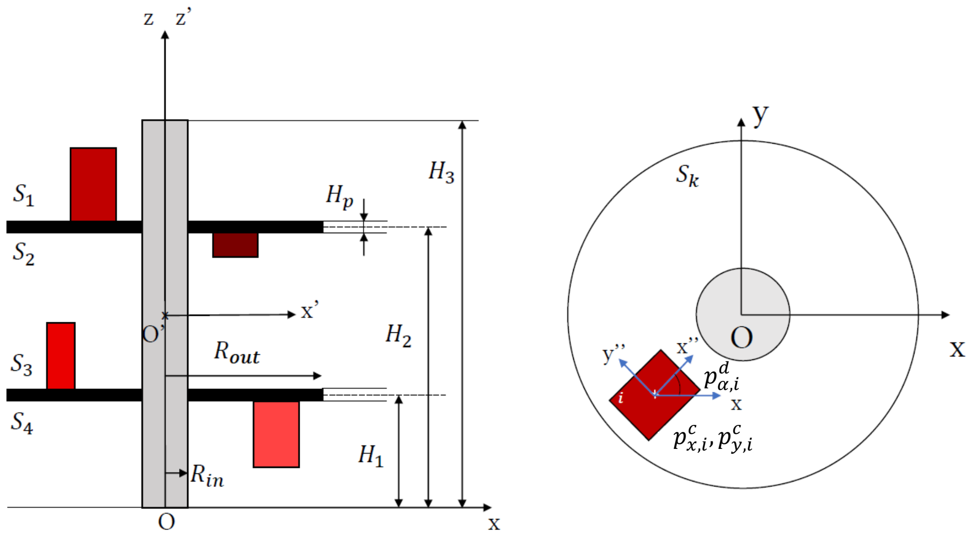


Figure 2. Geometrical definition of the multi-container configuration. The $Oxyz$ system of coordinates is linked to the satellite module while the $O'x'y'z'$ system of coordinates is linked to the system of components. O' stands for the current center of gravity of the components. The $O''x''y''z''$ system of coordinates is linked to one component i, which is located on one plate with the position of its center of inertia $(x_{x,i}, x_{y,i})$ and its orientation $x_{\alpha,i}$. The redder the components, the heavier their corresponding masses.

3.2. Design Variables

In the multi-container configuration, the components must be assigned to one of the four surfaces and be laid out on the corresponding surface. Therefore, the design variables are the following, where purely continuous variables are denoted $\mathbf{p}^c$ and discrete variables are denoted by $\mathbf{p}^d$.

- For each component, the number of its assigned surface $\mathbf{p}^d_s = \{p^d_{s,i}\}, i \in \{1, ..., N\}$, where $p^d_{s,i}$ are defined as unordered discrete variables;
- The location of the center of inertia of each component on the assigned surface considered as continuous variables $\mathbf{p}^c_l = \{p^c_{x,i}, p^c_{y,i}\}, i \in \{1, ..., N\}$;
- The orientations of cuboid components on the assigned surface are considered as discrete variables with values of 0° or 90°, $\mathbf{p}^d_\alpha = \{p^d_{\alpha,i}\}, i \in \{1, ..., N_{cub}\}$.

Thus, the number of design variables n_{var} is defined as:

$$n_{var} = N + 2N_{cyl} + 3N_{cub} = 3N_{cyl} + 4N_{cub} \quad (1)$$

3.3. Objective Function

The objective function to minimize is the total inertia of the module. Three systems of coordinates are considered:

- $Ox''y''z''$: the local coordinate system related to each component. O'' corresponds to the center of inertia of the component and the axes are defined with the symmetry planes of the components.
- $O'x'y'z'$: the coordinate system related to the system of components. O' stands for the current centroid of the system of components and the axes are defined with the symmetry planes of the module.
- $Oxyz$: the coordinate system related to the module. O is the geometric center of the container and the axes are defined using its symmetry planes.

The three coordinate systems are illustrated in Figure 2.

The global inertia I_{tot} to minimize is calculated in the $O'x'y'z'$ coordinate system, as proposed in [19,20,108], and defined as

$$I_{tot}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = I_{x'x'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) + I_{y'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) + I_{z'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \tag{2}$$

With the same formalism as the previous section, the solid inertias specific to each component with relation to their system of coordinates $O''x''y''z''$ are written $I_{x'',i}$, $I_{y'',i}$ and $I_{z'',i}$.

The inertia of the module along each axis of the $Oxyz$ coordinate system are written as follows:

$$I_{xx}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = I_{x,sat} \quad + \sum_{i=1}^{N} I_{x'',i}\cos(p_{\alpha,i}^d)^2 + I_{y'',i}\sin(p_{\alpha,i}^d)^2 + m_i({p_{y,i}^c}^2 + p_{z,i}^d(\mathbf{p}_s^d)^2) \tag{3}$$

$$I_{yy}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = I_{y,sat} \quad + \sum_{i=1}^{N} I_{y'',i}\cos(p_{\alpha,i}^d)^2 + I_{x'',i}\sin(p_{\alpha,i}^d)^2 + m_i({p_{x,i}^c}^2 + p_{z,i}^d(\mathbf{p}_s^d)^2) \tag{4}$$

$$I_{zz}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = I_{z,sat} \quad + \sum_{i=1}^{N} I_{z'',i} + m_i({p_{x,i}^c}^2 + {p_{y,i}^c}^2) \tag{5}$$

where $(p_{x,i}^c, p_{y,i}^c, p_{z,i}^d)$ are the coordinates of component i in the $Oxyz$ system of coordinates. The coordinates along the z-axis, $p_{z,i}^d$, are discrete and calculated thanks to the value of the $\mathbf{p}_s^d$ variables corresponding to the assigned container. Then, the inertias are calculated along the axes of the $O'x'y'z'$ coordinate system using the Huygens theorem:

$$I_{x'x'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = I_{xx}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) - (y_c^2 + z_c^2)m_{tot} \tag{6}$$

$$I_{y'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = I_{yy}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) - (x_c^2 + z_c^2)m_{tot} \tag{7}$$

$$I_{z'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = I_{zz}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) - (x_c^2 + y_c^2)m_{tot} \tag{8}$$

where (x_c, y_c, z_c) are the coordinates of the center of gravity of the module in the $Oxyz$ system of coordinates and $m_{tot} = (m_{sat} + \sum_{i=1}^{N} m_i)$ is the mass of the whole module.

3.4. Constraint Functions

The following geometrical and functional constraints are considered:

- Overlapping constraints between components: No overlapping between components is allowed on each surface. As in [19,20,108], the plates are supposed to be sufficiently spaced to avoid any overlapping between components positioned on the surfaces S_2 and S_3. The overlapping constraint between components is formulated as follows:

$$h_{overlap}^C(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sum_{k=1}^{N_s}\sum_{i=1}^{N_k-1}\sum_{j=1}^{N_k} \Delta A_{ij,k}^C(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \tag{9}$$

  where $N_s = 4$ is the number of surfaces, N_k is the number of components on surface k and $\Delta A_{ij,k}^C$ corresponds to the 2-dimensional projections (along the z axis) of components i and j on the surface k and is a function of the centers of inertia and orientations of the components.

- Overlapping constraints between the components and the exclusion zones: No overlapping between the components and the exclusion zones is allowed on each surface. The overlapping constraint is expressed as

$$h_{overlap}^E(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sum_{k=1}^{N_s}\sum_{i=1}^{N} \Delta A_{i,k}^E(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \tag{10}$$

  where $\Delta A_{i,k}^E$ is the area of intersection between the two-dimensional projection (along the z axis) of a component and the exclusion zone on the surface k and is a function of the centers of inertia and orientations of the components (as well as the positions and shapes of the exclusion zones).

- Balancing constraints: The center of gravity (CG) of the whole laid out module must be positioned near the center of gravity of the empty module within a given tolerance along the x and y axes. Then, the balancing constraint can be formulated as follows:

$$g_{CG}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sqrt{(x_c - x_{sat})^2 + (y_c - y_{sat})^2} - \delta_{CG} \tag{11}$$

where (x_c, y_c, z_c) are the coordinates of the current CG of the whole module, also referred to as O' in the system $Oxyz$; $(x_{sat}, y_{sat}, z_{sat})$ are the coordinates of the empty satellite module in the $Oxyz$ system; and δ_{CG} represents a tolerance that corresponds to a sphere centered at the empty satellite module center of gravity and is set to the numerical value of 3.0, without loss of generality.

- Constraints relative to the angles-of-inertia: In the multi-container configuration, a geometrical constraint corresponding to the angles of inertia is added, with a corresponding tolerance δ_{AI}. The constraint relative to the angles of inertia is defined as follows:

$$g_{AI}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sqrt{\theta_{x'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d)^2 + \theta_{x'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d)^2 + \theta_{y'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d)^2} - \delta_{AI} \tag{12}$$

where $\theta_{x'y'}$, $\theta_{x'y'}$ and $\theta_{y'z'}$ are the angles of inertia defined as

$$\theta_{x'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \arctan\left(-\frac{2I_{x'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d)}{(I_{y'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) - I_{x'x'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d)}\right)\Big/2 \tag{13}$$

$$\theta_{x'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \arctan\left(-\frac{2I_{x'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d)}{(I_{x'x'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) - I_{z'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d))}\right)\Big/2 \tag{14}$$

$$\theta_{y'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \arctan\left(-\frac{2I_{y'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d)}{(I_{y'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) - I_{z'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d))}\right)\Big/2 \tag{15}$$

where the inertias $I_{x'y'}$, $I_{y'z'}$ and $I_{y'z'}$ are expressed as

$$I_{x'y'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sum_{i=1}^{N}\left[m_i p_{x,i}^c p_{y,i}^c + \frac{\begin{matrix} I_{x''i} + m_i({p_{y,i}^c}^2 + {p_{z,i}^d}^2) \\ - I_{y''i} - m_i({p_{x,i}^c}^2 + {p_{z,i}^d}^2)\end{matrix}}{2} \sin(2p_{\alpha,i}^d) \right] - x_c y_c m_{tot} \tag{16}$$

$$I_{x'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sum_{i=1}^{N} m_i p_{x,i}^c p_{z,i}^d - x_c z_c m_{tot} \tag{17}$$

$$I_{y'z'}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sum_{i=1}^{N} m_i p_{y,i}^c p_{z,i}^d - y_c z_c m_{tot} \tag{18}$$

- Functional constraints: These are also defined in terms of incompatibility between several components. Thermal as well as electromagnetic thresholds are defined such that some components responsible for thermal or electromagnetic fields must be spaced away at given distances. More precisely, a set C_H of pairs of incompatible heat components $\{i,j\} \in C_H$ and a set C_E of pairs of incompatible electromagnetic components $\{i,j\} \in C_E$ are defined. Thus, the thermal functional constraint is defined as

$$g_H(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sum_{k=1}^{\text{Card}(C_H)} d_H - d_k(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \tag{19}$$

where $\text{Card}(C_H)$ is the cardinal of the set C_H corresponding to the list of incompatible pairs of thermal radiative components, d_k is the distance between the two components of pair k if positioned on the same surface and d_H is the minimal distance between the two components of each pair of C_H.
With the same formalism, the electromagnetic functional constraint is defined as

$$g_E(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = \sum_{k=1}^{\text{Card}(C_E)} d_E - d_k(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \leq 0 \tag{20}$$

where $\text{Card}(C_E)$ is the cardinal of the set C_E corresponding to the list of incompatible pairs of electromagnetic components, d_k is the distance between the two components of pair k if positioned on the same surface and d_E is the minimal distance between the two components of each pair of C_E. The list of thermal and electromagnetic incompatible components is specified in Appendix A.

3.5. Mathematical Formulation

Thus, multi-container SMLPs can be mathematically defined as follows:

$$\begin{array}{rl} \underset{\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d}{\min} & I_{tot}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \\ \text{w.r.t.} & \mathbf{p}_l^c \in F_{\mathbf{p}_l^c} \subseteq \mathbb{R}^{n_{\mathbf{p}_l^c}}, \mathbf{p}_\alpha^d \in F_{\mathbf{p}_\alpha^d} \subseteq \mathbb{N}^{n_{\mathbf{p}_\alpha^d}}, \mathbf{p}_s^d \in F_{\mathbf{p}_s^d} \subseteq \mathbb{N}^{n_{\mathbf{p}_s^d}} \\ \text{s.t.} & \mathbf{h}_{overlap}^C(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = 0 \\ & \mathbf{h}_{overlap}^E(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = 0 \\ & g_{CG}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \leq 0 \\ & g_{AI}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \leq 0 \\ & \mathbf{g}_H(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \leq 0 \\ & \mathbf{g}_E(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \leq 0 \end{array} \tag{21}$$

Remarks on the Inertia Equations

In all the papers dealing with the SMLP [19,20,48,108,112,117], the inertias are calculated along with the axes of the $O'x'y'z'$ system corresponding to the coordinate system linked to the current centroid of the whole module. They are obtained by calculating the inertia in the $Oxyz$ system and then by applying Huygens theorem, which leads to Equations (6)–(8). However, as the objective function is to minimize the sum of the three aforementioned equations, it will also lead to a maximization of the three positive second terms:

$$\begin{array}{c} (y_c^2 + z_c^2)m_{tot}, \\ (x_c^2 + z_c^2)m_{tot}, \\ (x_c^2 + y_c^2)m_{tot}. \end{array}$$

Thus, the formulation of the inertia also contributes to maximize the coordinates of the global center of gravity of the module. As the balancing constraint (Equation (10)) bounds these coordinates to a tolerance zone, the optimization process will lead to a positioning of the center of gravity to the boundaries of the balancing constraint. However, the goal of the optimization process would rather be to minimize the inertia along with positioning the center of gravity as close as possible to a position of reference, which often corresponds to the geometrical center of the module or to the center of gravity of the empty module if different.

Moreover, the first terms of the inertia equations are calculated in the $Oxyz$ system of coordinates, which is linked to the bottom of the satellite. Minimizing these quantities includes minimizing the terms $m_i {p^c_{x,i}}^2$ and $m_i {p^c_{y,i}}^2$ but also $m_i {p^d_{z,i}}^2$. Then, the optimization process based on these inertia equations might overload the bottom surfaces S_3 and S_4, which leads to an unbalanced satellite module.

Consequently, in order to avoid the two aforementioned issues, the inertia equations should be written in the system of coordinates linked to the position of reference where the center of gravity has to be positioned, as developed in Appendix B. However, in the present study, for the sake of numerical comparison with existing works, the equations employed in the literature, i.e., Equations (6)–(8), are used in the following experiments.

4. Two-Stage Algorithm Combining Genetic Algorithm and Quasi-Physical Approach

4.1. General Structure of the Algorithm

In the case of the multi-container SMLP as well as in the majority of cases addressed in the literature [19,118–122], the assignment and layout tasks are fully separated. Consequently, an objective function related to the assignment that only relies on the design variables of the assignment scheme can be defined, while a layout objective function takes as argument the $\mathbf{p}^c_l$ and $\mathbf{p}^d_\alpha$ layout design variables as well as the assignment scheme found by the upper optimization problem and described by the $\mathbf{p}^d_s$ variables. The resulting formulation is a nested problem, with an outer problem handling the assignment variables and an inner problem handling the variables dealing with the positions and orientations of the components. The mathematical formulation of the resulting nested optimization problem is as follows:

$$\begin{aligned}
\min_{\mathbf{p}^d_s} \quad & f_{assignment}(\mathbf{p}^d_s) \\
\text{s.t.} \quad & \mathbf{h}_{assignment}(\mathbf{p}^d_s) = 0 \\
& \mathbf{g}_{assignment}(\mathbf{p}^d_s) \leq 0 \\
& \mathbf{h}^*_{layout}(\mathbf{p}^{c*}_l, \mathbf{p}^{d*}_\alpha, \mathbf{p}^d_s) = 0 \\
& \mathbf{g}^*_{layout}(\mathbf{p}^{c*}_l, \mathbf{p}^{d*}_\alpha, \mathbf{p}^d_s) \leq 0 \\
\text{w.r.t.} \quad & \mathbf{p}^d_s \in F_z \subseteq \mathbb{N}^{n_{p^d_s}} \\
& \{\mathbf{p}^{c*}_l, \mathbf{p}^{d*}_\alpha\} = \operatorname{argmin} f_{layout}(\mathbf{p}^c_l, \mathbf{p}^d_\alpha, \mathbf{p}^d_s) \\
& \text{w.r.t.} \quad \mathbf{p}^c_l \in F_{p^c_l} \subseteq \mathbb{R}^{n_{p^c_l}}, \mathbf{p}^d_\alpha \in F_{p^d_\alpha} \subseteq \mathbb{N}^{n_{p^d_\alpha}} \\
& \qquad\quad \mathbf{h}_{layout}(\mathbf{p}^c_l, \mathbf{p}^d_\alpha, \mathbf{p}^d_s) = 0 \\
& \qquad\quad \mathbf{g}_{layout}(\mathbf{p}^c_l, \mathbf{p}^d_\alpha, \mathbf{p}^d_s) \leq 0
\end{aligned} \tag{22}$$

where $f_{assignment}$, $\mathbf{h}_{assignment}$ and $\mathbf{g}_{assignment}$ are, respectively, the objective function, the equality and the inequality constraint functions related to the assignment, taking as argument only the $\mathbf{p}^d_s$ design variables. f_{layout} is the objective function related to the layout, taking all design variables as arguments. $\mathbf{h}^*_{layout}$ and $\mathbf{g}^*_{layout}$ ensures that a feasible layout has been found by the inner optimization problem. The star symbol $(^*)$ stands for optimal values of optimization variables (and corresponding constraints) provided by the inner optimization problem.

To address the formulation detailed in the equation system (22), a two-stage algorithm is proposed. The upper and lower stages aim at solving the assignment task and the layout task, respectively. Each stage is described as follows.

4.2. Assignment Task

4.2.1. Formulation of the Assignment Subproblem

The upper stage solves the assignment task. It takes as input the list of geometrical features of the components and the containers (dimensions, masses and exclusion zones). The output of the upper stage is an archive of N_{AS} assignment schemes corresponding to N_{AS} lists of assigned containers to each component. Thus, the design variables to optimize are $\mathbf{p}_s^d = \{p_{s,i}^d\}_{i\in\{1,\dots,N\}}$, where $p_{s,i}^d \in \{1,\dots,n\}$.

In order to minimize the global inertia, the components should be distributed among the containers such that their center of gravity along the z axis is centered at the center of gravity of the empty module. Then, the assignment objective function consists in minimizing the terms of the global inertia equations depending on the $p_{z,i}^c$ variables while positioning the center of gravity of the components along the z axis in a tolerance zone centered at the empty module center of gravity and while not exceeding a 65% occupation rate per container (without loss of generality). Indeed, for the SMLP problem, larger occupation rates might lead to unfeasible layout configurations [123]. The corresponding optimization subproblem is formulated as follows:

$$\begin{aligned} \min_{\mathbf{p}_s^d} \quad & \sum_{i=1}^{N} m_i (p_{z,i}^d(\mathbf{p}_s^d) - z_e)^2 \\ \text{w.r.t.} \quad & \mathbf{p}_s^d \in F_{\mathbf{p}_s^d} \subseteq \{1,2,3,4\}^N \\ \text{s.t.} \quad & g_{CG}(\mathbf{p}_s^d) \le 0 \\ & g_O^j(\mathbf{p}_s^d) \le 0 \quad \text{for } j \in \{1,2,3,4\} \end{aligned} \tag{23}$$

where $p_{z,i}^d$ are the positions of the center of inertia of the components of mass m_i that are defined by the values of $\mathbf{p}_s^d$ in the assignment list. z_e is a position of reference, which is taken as the center of gravity of the empty module. The inequality constraints are, respectively, as follows:

- The balancing constraint:

$$g_{CG}(\mathbf{p}_s^d) = \left| \frac{\sum_{i=1}^{N} m_i p_{z,i}^d}{\sum_{i=1}^{N} m_i} - z_e \right| - \delta_{CG,z} \tag{24}$$

  The value of $\delta_{CG,z}$ represents the size of the tolerance zone for the positioning of the center of gravity along the z-axis. Without loss of generality, it is taken as equal to 3.0, which corresponds to the size of the tolerance zones along the x- and y-axes [19,20,48,108,112,117] so that the center of gravity must be positioned in a sphere of tolerance with a radius of 3.0 mm centered at the geometrical center of gravity of the empty module.
- The occupation rate constraints defined for each surface j:

$$g_O^j(\mathbf{p}_s^d) = \frac{A_j^{components}}{A_j^{container}} - OR_{lim} \tag{25}$$

  where $A_j^{components}$ is the total area of the components assigned to container j and $A_j^{container}$ is the area of container j. OR_{lim} corresponds to the occupation rate that should not be exceeded in each container. It is taken equal to 0.65 to ensure a compromise between container filling and problem feasibility, as shown in [123,124].

4.2.2. Algorithm for the Assignment Task

The problem to solve is a constrained categorical combinatorial problem. The size of the design space is n^N. Metaheuristics have been widely employed to solve this kind of problem [125]. Among these methods, Genetic Algorithms are preferred due to their capacity for global exploration, especially when dealing with multi-modal objective functions. Additionally, Genetic Algorithms can effectively handle a substantial number of variables, distinguishing them from other methods such as Bayesian Optimization. Moreover, they have been widely employed to solve highly combinatorial problems and can inherently deal with categorical variables in comparison to other metaheuristic techniques like PSO.

4.3. Layout Task

4.3.1. Formulation of the Layout Subproblem

The lower stage solves the layout task. This stage takes as input the assignment list provided by the upper stage. The outputs are the lists of positions and orientations of each component in its assigned container. In the multi-container SMLP, the layout of each container can be independently solved. In other words, the constraints are individually defined for each container and there is no constraint function that requires a simultaneous resolution of the containers' layouts. Consequently, the design variables to optimize are $\mathbf{p}_l^c = \{p_{x,i}^c, p_{y,i}^c\}, i \in \{1, ..., N\}$ and $\mathbf{p}_\alpha^d = \{p_{\alpha,i}^d\}, i \in \{1, ..., N_{cub}\}$, where $\mathbf{p}_{\alpha,i}^d \in \{0, 90\}$. Those variables are optimized with respect to the assignment scheme $\mathbf{p}_s^d$ from the upper level.

The corresponding optimization subproblem solved for each container is formulated as follows:

$$\begin{aligned}
\min_{\mathbf{p}_l^c, \mathbf{p}_\alpha^d} \quad & I_{tot,SC}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \\
\text{w.r.t.} \quad & \mathbf{p}_l^c \in F_{p_l^c} \subseteq \mathbb{R}^{n_{p_l^c}}, \mathbf{p}_\alpha^d \in F_{p_\alpha^d} \subseteq \{0, 90\}^{n_{p_\alpha^d}} \\
\text{s.t.} \quad & \mathbf{h}_{overlap,SC}^C(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = 0 \\
& \mathbf{h}_{overlap,SC}^E(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = 0 \\
& \mathbf{h}_{functional,SC}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) = 0 \\
& g_{CG,SC}(\mathbf{p}_l^c, \mathbf{p}_\alpha^d, \mathbf{p}_s^d) \leq 0
\end{aligned} \tag{26}$$

where $I_{tot,SC}$ stands for the inertia of one container and $\mathbf{h}_{overlap,SC}^C$, $\mathbf{h}_{overlap,SC}^E$, $\mathbf{h}_{functional,SC}$ and $g_{CG,SC}$ are the overlapping, functional and balancing constraints calculated for one container.

4.3.2. Algorithm for the Layout Task

The proposed algorithm to solve the layout of each assigned container is a Component Swarm Optimization algorithm based on a Virtual-force system (CSO-VF) [124]. This algorithm is in line with other quasi-physical algorithms based on the virtual-forces system, as discussed in [3,56,57,126].

The main focus of the algorithm is to define dedicated operators for the evolution of a dynamical system of the components based on the fundamental principle of dynamics to efficiently satisfy the constraints while minimizing the objective function. In CSO-VF, each component is assumed to be a particle in a swarm. At each iteration of the algorithm, depending on the virtual forces that are applied to the particle, each component moves within the container until the objective function is minimized and all constraints are satisfied.

It is important to note that this type of algorithms differs from classical Particle Swarm Optimization (PSO) algorithms. Indeed, in the CSO-VF algorithm, each particle of the swarm corresponds to a single component; thus, to a subset of the entire solution while in PSO algorithms, a particle corresponds to an entire solution, i.e., an entire layout.

Virtual-Force System

In the CSO-VF algorithm, each component i is described by its dynamic features:

- Its translational and rotational accelerations $\mathbf{a}_i$ and $\mathbf{L}_i$;
- Its translational and rotational speeds $\mathbf{v}_i$ and $\boldsymbol{\omega}_i$;
- Its position $\mathbf{p}_{l,i}^c$ and orientation $\mathbf{p}_{\alpha,i}^d$.

N_F forces ($\mathbf{F}_i^k, k \in \{1, ..., N_F\}$, for component i) with a resultant $\mathbf{F}_i$ as well as N_T torques ($\mathbf{T}_i^k, k \in \{1, ..., N_T\}$, for component i) with a resultant $\mathbf{T}_i$ are applied to the component in order to induce a movement at each iteration, as depicted in Figure 3. It must be noted that the torques are applied to non-circular components only.

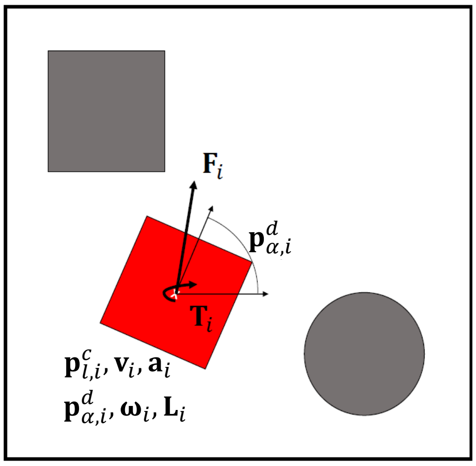

$\mathbf{F}_i$
$\mathbf{p}_{\alpha,i}^d$
$\mathbf{T}_i$
$\mathbf{p}_{l,i}^c, \mathbf{v}_i, \mathbf{a}_i$
$\mathbf{p}_{\alpha,i}^d, \boldsymbol{\omega}_i, \mathbf{L}_i$

Figure 3. Definition of a component.

At the end of an iteration, the resulting force $\mathbf{F}_i$ and torque $\mathbf{T}_i$ are calculated for each component using Equations (27) and (28). The Fundamental Principles of the Dynamics of rotation and translation are applied in order to update the positions and orientations of the swarm of components at step $t+1$ (separated from step t by Δt, which corresponds to a time unit) according to Equations (29)–(31) (detailed for component i with mass m_i and solid inertia I_i).

$$\mathbf{F}_i = \begin{cases} \sum_{k=1}^{N_F} \mathbf{F}_i^k & \text{if } \left\| \sum_{k=1}^{N_F} \mathbf{F}_i^k \right\| \leq F_{max} \\ \frac{\sum_{k=1}^{N_F} \mathbf{F}_i^k}{\left\| \sum_{k=1}^{N_F} \mathbf{F}_i^k \right\|} F_{max} & \text{otherwise.} \end{cases} \tag{27}$$

$$\mathbf{T}_i = \begin{cases} \sum_{k=1}^{N_T} \mathbf{T}_i^k & \text{if } \left\| \sum_{k=1}^{N_T} \mathbf{T}_i^k \right\| \leq T_{max} \\ \frac{\sum_{k=1}^{N_T} \mathbf{T}_i^k}{\left\| \sum_{k=1}^{N_T} \mathbf{T}_i^k \right\|} T_{max} & \text{otherwise.} \end{cases} \tag{28}$$

where F_{max} and T_{max} are hyperparameters corresponding to the maximum values of the norm of the resulting force and torque vectors, respectively.

$$\mathbf{a}_{i,t+1} = \frac{\mathbf{F}_i}{m_i} \tag{29}$$

$$\mathbf{v}_{i,t+1} = \mathbf{v}_{i,t} + \mathbf{a}_{i,t+1}\Delta t \tag{30}$$

$$\mathbf{p}_{l,i,t+1}^c = \mathbf{p}_{l,i,t}^c + \mathbf{v}_{i,t+1}\Delta t \tag{31}$$

$$\mathbf{L}_{i,t+1} = \frac{\mathbf{T}_i}{I_i} \tag{32}$$

$$\boldsymbol{\omega}_{i,t+1} = \boldsymbol{\omega}_{i,t} + \mathbf{L}_{i,t+1}\Delta t \tag{33}$$

$$\mathbf{p}_{\alpha,i,t+1}^d = \mathbf{p}_{\alpha,i,t+1}^d + \boldsymbol{\omega}_{i,t+1}\Delta t \tag{34}$$

Each of the forces and torques of the VFS aims at solving for the constraints and optimizing the objective function(s). If the objective or constraint functions take as arguments the positions of the components, a force is applied. If they take as arguments the orientations of the components, a torque is applied.

The forces and torques applied to a component i are employed to achieve the following:

- Minimize an objective function: force $\mathbf{F}_i^{objective}$ and torque $\mathbf{T}_i^{objective}$;
- Solve for the overlapping constraints between components i and j: force $\mathbf{F}_{ij}^{overlap}$ and torque $\mathbf{T}_{ij}^{overlap}$;
- Solve for the overlapping constraints between component i and an exclusion zone (EZ): force $\mathbf{F}_{i/EZ}^{overlap}$ and torque $\mathbf{T}_{i/EZ}^{overlap}$;
- Solve for the overlapping constraints between component i and the container: force $\mathbf{F}_i^{container}$ and torque $\mathbf{T}_i^{container}$;
- Solve for the functional constraints between two components i and j: force $\mathbf{F}_i^{functional}$ and torque $\mathbf{T}_i^{functional}$;
- Solve for the balancing constraint: force $\mathbf{F}_i^{CG}$ (CG stands for center of gravity);
- Solve for the angle of inertia constraint: force $\mathbf{F}_i^{AI}$ and torque $\mathbf{T}_i^{AI}$;
- More generally, solve for any constraint function: force $\mathbf{F}_i^{constraint}$ and torque $\mathbf{T}_i^{constraint}$.

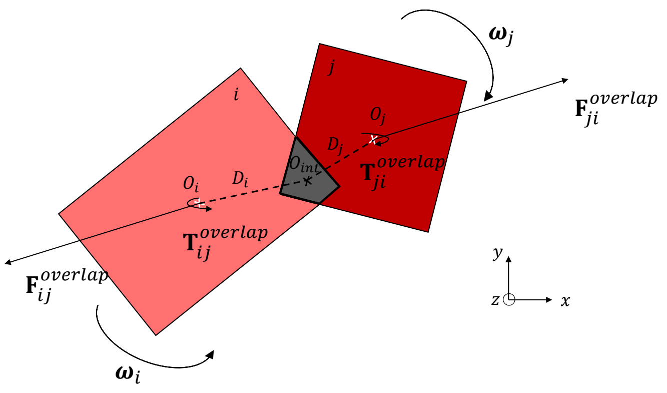
Generally speaking, any force or torque can be added to the VFS in order to address additional constraints or objective functions. This makes the proposed algorithm very generic to other layout problems.

The forces and torques of the virtual-force system are detailed and formulated as follows (for a component i):

- The overlappingconstraint force and torque between two components: If two components i and j are overlapping each other, repulsive forces $\mathbf{F}_{ij}^{overlap}$ and $\mathbf{F}_{ji}^{overlap}$ are applied to each of them, as illustrated in Figures 4 and 5. The overlap force is expressed as

$$\mathbf{F}_{ij}^{overlap} = \begin{cases} -\dfrac{\mathbf{p}_{l,j}^c - \mathbf{p}_{l,i}^c}{\|\mathbf{p}_{l,j}^c - \mathbf{p}_{l,i}^c\| + \epsilon} v_{max} - \mathbf{v}_i & \text{if } \Delta S_{ij}(\mathbf{p}_{l,i}^c, \mathbf{p}_{l,j}^c) \neq 0, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{35}$$

  where v_{max} is a hyperparameter corresponding to the maximum value of the norm of the speed vector, ϵ ensures numerical stability and $\mathbf{0} = (0,0)$ is the null vector. ΔS_{ij} is the area of intersection between components i and j.
  Moreover, similar to [127], additional torques are applied to non-circular components in order to solve the overlapping constraint, as illustrated in Figure 4.



$\boldsymbol{\omega}_j$
j
i
$\mathbf{F}_{ji}^{overlap}$
O_j
D_j
O_{int}
$\mathbf{T}_{ji}^{overlap}$
O_i
D_i
$\mathbf{T}_{ij}^{overlap}$
$\mathbf{F}_{ij}^{overlap}$
y
z
x
$\boldsymbol{\omega}_i$

Figure 4. Definition of the overlapping forces and torques.

The overlapping forces are initially applied at the O_{int} point, which corresponds to the geometrical center of the polygon of intersection between the two components. Then, the resulting torque applied at the point O_i is given by

$$\mathbf{T}_{ij}^{overlap} = \mathbf{O_i O_{int}} \wedge \mathbf{F}_{ij}^{overlap} \tag{36}$$

where $\mathbf{O_i O_{int}}$ is the vector from O_i to O_{int} and the symbol $\wedge$ stands for the vector product.

- The overlapping constraint force and torque between a component and an exclusion zone: If a component is overlapping an exclusion zone, a repulsive force $\mathbf{F}_{i/EZ}^{overlap}$ is applied to the component, as illustrated in Figure 5. As the exclusion zone can be seen as a fixed component, the corresponding force and torque expressions are similar to the previous overlapping forces and torques between the two components and written with Equations (35) and (36).
- The overlapping constraint force and torque between a component and the container: If a component is not fully overlapping the container, an attractive force $\mathbf{F}_i^{container}$ is applied to the component, as illustrated in Figure 5. The container force is expressed as

$$\mathbf{F}_i^{container} = \begin{cases} \dfrac{\mathbf{p}_{l,C}^c - \mathbf{p}_{l,i}^c}{\|\mathbf{p}_{l,C}^c - \mathbf{p}_{l,i}^c\| + \epsilon} v_{max} - \mathbf{v}_i, & \text{if } \Delta S_{i/C}(\mathbf{p}_{l,i}^c, \mathbf{p}_{l,C}^c) < S_i, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{37}$$

where $\mathbf{p}_{l,C}^c$ corresponds to the position vector of the geometrical center of the container. $\Delta S_{i/C}$ is the intersection area between the component i and the exclusion zone, and S_i is the surface of the component i. The related torque is calculated as follows:

$$\mathbf{T}_i^{container} = \mathbf{O_i O_{out}} \wedge \mathbf{F}_i^{container} \tag{38}$$

where O_i is the center of inertia of component i and O_{out} is the geometrical center of the area of the component i situated outside the container.

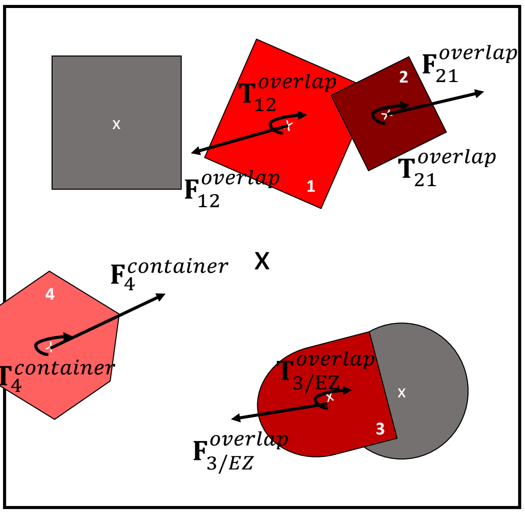

$\mathbf{F}_{21}^{overlap}$
$\mathbf{T}_{12}^{overlap}$
2
$\mathbf{T}_{21}^{overlap}$
$\mathbf{F}_{12}^{overlap}$
1
$\mathbf{F}_{4}^{container}$
4
$\mathbf{T}_{4}^{container}$
$\mathbf{T}_{3/EZ}^{overlap}$
3
$\mathbf{F}_{3/EZ}^{overlap}$

Figure 5. Illustration of the forces and torques to solve overlapping constraints between two components, between a component and an exclusion zone, and between a component and the container.

- The functional constraint force and torque between two components: If a component i is too close to an incompatible component j, a repulsive force $\mathbf{F}_i^{functional}$ is applied to the component i, as illustrated in Figure 6. The functional force acts as an overlapping force between the component i and the influence zone of the component j. Then, the functional force is expressed as

$$\mathbf{F}_i^{functional} = \begin{cases} -\dfrac{\mathbf{p}_{l,j}^c - \mathbf{p}_{l,j}^c}{\|\mathbf{p}_{l,j}^c - \mathbf{p}_{l,i}^c\| + \epsilon} v_{max} - \mathbf{v}_i, & \text{if } \Delta S_{i/SZj}(\mathbf{p}_{l,i}^c, \mathbf{p}_{l,j}^c) \neq 0, \\ \mathbf{0} & \text{otherwise.} \end{cases} \quad (39)$$

where $\Delta S_{i/SZj}$ is the intersection area between component i and the functional security zone surrounding component j.
The related torque is calculated as follows:

$$\mathbf{T}_i^{functional} = \mathbf{O_i O_{zj}} \wedge \mathbf{F}_i^{functional} \quad (40)$$

where O_i is the center of inertia of component i and O_{zj} is the geometrical center of the area of component i overlapping the functional security zone surrounding component j.

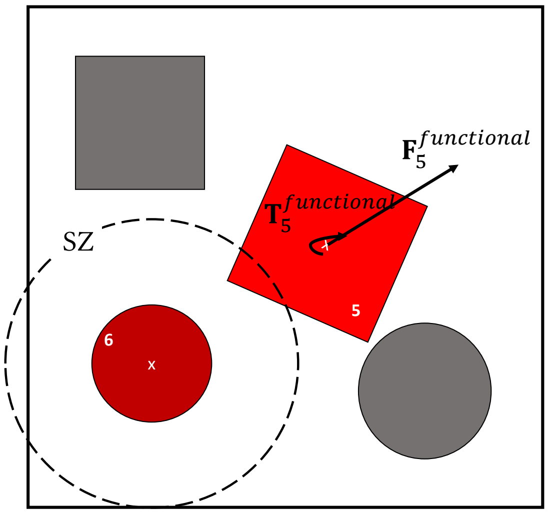


Figure 6. Illustration of the functional force and torque.

- The balancing constraint force: In order to position the center of mass of the components in a tolerance zone centered at the geometrical center of the container, gradient-based forces are applied along the opposite of the gradient of the position of the global center of mass according to the position of the center of inertia of each component. This force is named $\mathbf{F}_i^{CG}$ and is illustrated in Figure 7. The balancing force is expressed as

$$\mathbf{F}_i^{CG} = \begin{cases} -\alpha_{CG} \nabla \mathbf{h}_{CG}^F(\mathbf{p}_{l,i}^c), & \text{if } \sqrt{(x_{CG} - x_e)^2 + (y_{CG} - y_e)^2} \leq \delta_{CG}, \\ \mathbf{0} & \text{otherwise.} \end{cases} \quad (41)$$

where α_{CG}^F is a "step-size" hyperparameter of the algorithm and $\nabla \mathbf{h}_{CG}(\mathbf{p}_{l,i}^c)$ corresponds to the gradient of the position of the center of gravity of the components with respect to the position of the considered component i. $\{x_{CG}, y_{CG}\}$ are the coordinates of the current center of gravity; $\{x_e, y_e\}$ are the coordinates of the position of reference on which the tolerance zone defined by δ_{CG} is centered. This type of force is inspired by gradient-based descent algorithms.

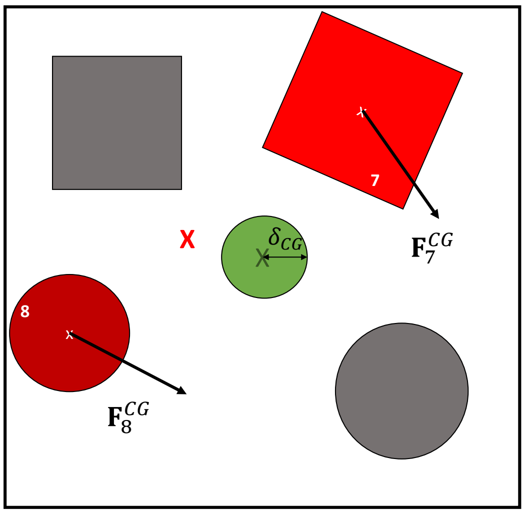


Figure 7. Illustration of the balancing forces. The green area corresponds to the tolerance zone where the current center of gravity must be positioned. The red cross is the current center of gravity of the components.

- The differentiable constraint function force and torque: Generally, any differentiable constraint function (DF), as the angle of inertia constraints, can be addressed thanks to gradient-based forces and torques, expressed as

$$\mathbf{F}_i^{DF} = -\alpha_{DF}^F \nabla \mathbf{h}_{DF}(\mathbf{p}_{l,i}^c) \tag{42}$$

$$\mathbf{T}_i^{DF} = -\alpha_{DF}^T \nabla \mathbf{h}_{DF}(\mathbf{p}_{\alpha,j}^d) \tag{43}$$

for constraints expressed as $\mathbf{g}(\cdot) \leq 0$ and where α_{DF}^F and α_{DF}^T are "step-size" hyperparameters of the algorithm. $\nabla \mathbf{h}_{DF}(\mathbf{p}_{l,i}^c)$ corresponds to the gradient of any differentiable constraint function with respect to the position of the considered component i. $\nabla \mathbf{h}_{DF}(\mathbf{p}_{\alpha,j}^d)$ corresponds to the gradient of any differentiable constraint function with respect to the orientation of the considered component i.

- The objective function forces and torques: To minimize one objective function f, a gradient-based force and a gradient-based torque can be applied. The objective function force and torque are expressed as

$$\mathbf{F}_i^{obj} = -\alpha_{obj}^F \nabla f_{obj}(\mathbf{p}_{l,i}^c) \tag{44}$$

$$\mathbf{T}_i^{obj} = -\alpha_{obj}^T \nabla f_{obj}(\mathbf{p}_{\alpha,i}^d) \tag{45}$$

where α_{obj}^F and α_{obj}^T are "step-size" hyperparameters of the algorithm. $\nabla f_{obj}(\mathbf{p}_{l,i}^c)$ corresponds to the gradient of the objective function with respect to the position of the considered component i, and $\nabla f_{obj}(\mathbf{p}_{\alpha,i}^d)$ stands for the gradient of the objective function with respect to the orientation of the considered component.

It must be noted that in the case where the orientation variables are discrete (i.e., non-cylinder components taking orientation values in a defined subset), they are handled as relaxed continuous orientations. The updated orientation is calculated as a continuous variable and a threshold is set to define the discrete orientation from the subset. For instance, in the case where the non-cylinder components take their orientation in the subset (0°,90°), if the positive updated continuous orientation is greater (respectively, lower) than 45° the updated discrete orientation is 90° (respectively, 0°).

Swap Operator

The swap operator introduced in CSO-VF is an extended version of the previous swap operators [128–130]. The swap operator exchanges the positions of a pair of components after a given number of iterations if it is improving the objective function(s) or decreasing the violation of some selected constraint(s). Thus, the swap operator can be used to find promising configurations in terms of constraint satisfaction and objective function; therefore, it does not necessarily need the configuration to be somewhat stuck in order to be employed in CSO-VF algorithm. For instance, the swap operator can be used to exchange components such that the global center of gravity of the system quickly enters the given tolerance zone, as illustrated in Figure 8.

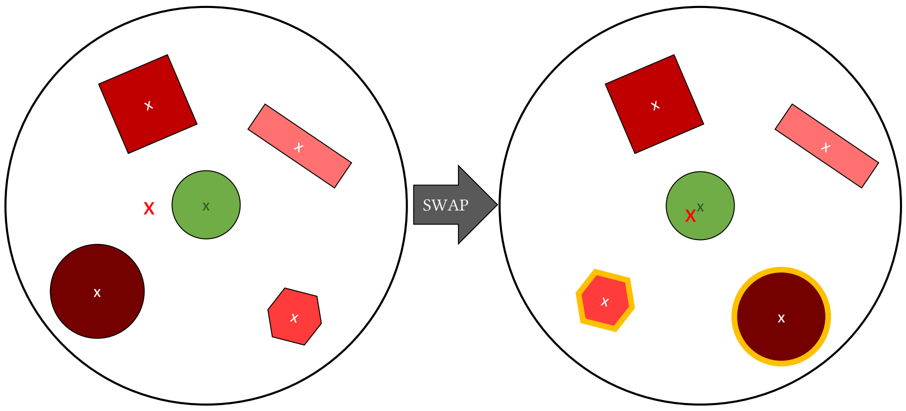


Figure 8. Example of the swap operator employed to improve the balancing constraint. The green area is the tolerance zone in which the current center of gravity of the components, illustrated thanks to the red cross, must be placed. The redder the components, the heavier their corresponding masses. The swapped components are highlighted in yellow.

Thus, the swap operator implemented in the CSO-VF algorithm has the following characteristics:

- The swap operator can exchange all the components by pairs.
- Two components are swapped if the swap leads to an improvement of the objective function or a decrease in the violation of some chosen constraint(s) while not deteriorating the other constraint(s) from a relaxation factor *r*.
- The swap operator is called straight from the first iteration. Then, it can occur throughout the optimization process.

The swap operator is called at a frequency depending on the convergence status. Indeed, the swap operator induces a reconfiguration. Consequently, too frequent swaps might prevent the layout from converging. On the contrary, too spaced swaps might lead to configurations being stuck for too many iterations and, thus, to a stagnating or slow convergence rate. The equations related to the call frequency of the operator are detailed in Appendix C.

Initialization

As the CSO-VF algorithm locally refines a solution for non-convex problems with multiple local minima, the initialization of the design variables might have a critical impact on the optimization process. Rather than a random initialization of the positions and orientations of the components, optimized Latin Hypercube Sampling (LHS) is employed [131–133]. This technique should improve the constraint resolution as well as provide a feasible solution more rapidly, i.e., in less iterations than would otherwise be necessary with a random initialization.

Multistart

As each instance of the algorithm depends on the initialization, a multistart technique is employed. Indeed, N_{start} independent initializations are run and the output of the CSO-VF algorithm corresponds to the best layout obtained out of the N_{start} initializations.

Various termination criteria can be used in the CSO-VF algorithm. In the following, a maximum number of iterations is set as a termination criterion. Figure 9 describes the CSO-VF algorithm.

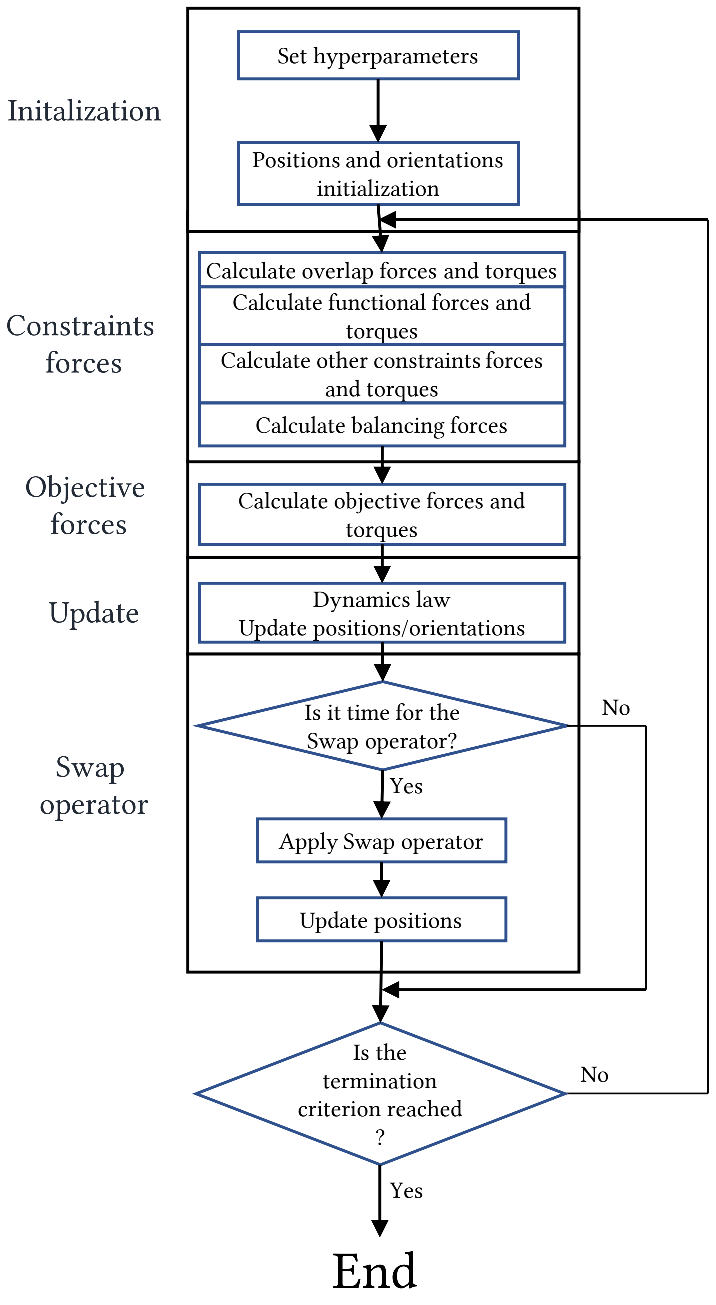

Initalization
Set hyperparameters
Positions and orientations initialization
Constraints forces
Calculate overlap forces and torques
Calculate functional forces and torques
Calculate other constraints forces and torques
Calculate balancing forces
Objective forces
Calculate objective forces and torques
Update
Dynamics law Update positions/orientations
Swap operator
Is it time for the Swap operator?
No
Yes
Apply Swap operator
Update positions
Is the termination criterion reached ?
No
Yes
End

Figure 9. The CSO-VF algorithm.

4.4. *Algorithm Framework*

Figure 10 illustrates the proposed algorithm applied to the SMLP problem.

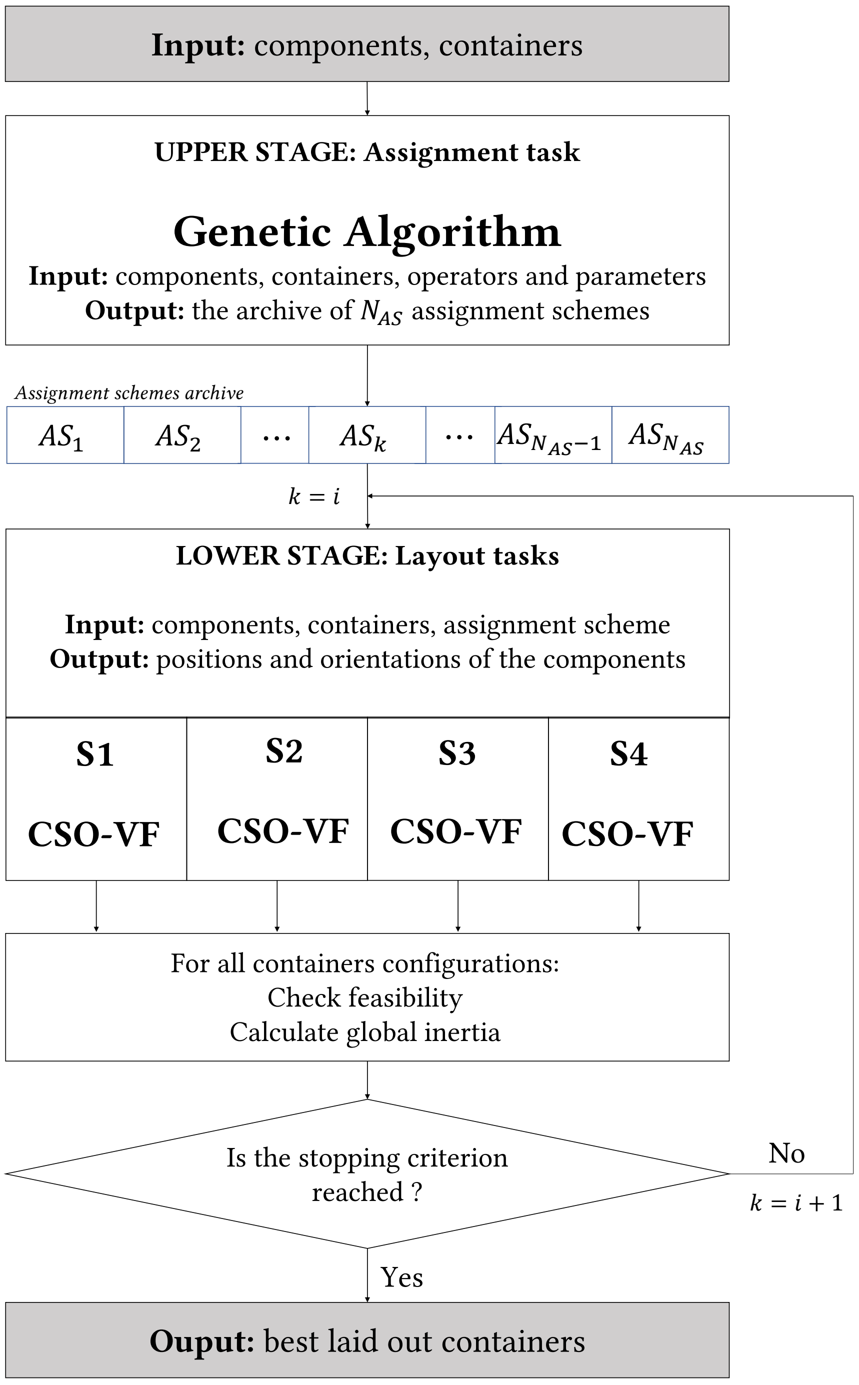

Input: components, containers
UPPER STAGE: Assignment task
Genetic Algorithm
Input: components, containers, operators and parameters
Output: the archive of N_{AS} assignment schemes
Assignment schemes archive
AS_1
AS_2
$\cdots$
AS_k
$\cdots$
$AS_{N_{AS}-1}$
$AS_{N_{AS}}$
$k = i$
LOWER STAGE: Layout tasks
Input: components, containers, assignment scheme
Output: positions and orientations of the components
S1 CSO-VF
S2 CSO-VF
S3 CSO-VF
S4 CSO-VF
For all containers configurations:
Check feasibility
Calculate global inertia
Is the stopping criterion reached ?
No
$k = i + 1$
Yes
Ouput: best laid out containers

Figure 10. The two-stage algorithm for the multi-container SMLP. In the algorithm framework, i stands for the current iteration.

The algorithm proceeds as follows: the upper stage optimizes the assignment list using the Genetic Algorithm. The output corresponds to an archive with N_{AS} assignment schemes corresponding to the N_{AS} final best solutions and sorted based on their objective function values. Subsequently, the CSO-VF algorithm is called to optimize the layout of each of the containers for the first assignment scheme of the archive. The CSO-VF instances dedicated to each container are launched in parallel. Then, all the possible combinations of laid out containers are compared in terms of feasibility and objective function values, and the combination leading to the smaller global inertia corresponds to the output of the algorithm. In the specific case where no feasible combination is found, the following assignment list from the archive is considered, the containers' layouts are optimized once again, and so on. The termination criterion is defined as a maximum number of sequential resolutions, i.e., the size N_{AS} of the archive given by the upper level.

5. Application to the Satellite Module Layout Problem

In this section, the two-stage algorithm is applied to the multi-container SMLP. The upper and lower stages settings are first configured and their performances are shown and analyzed. The global results and performance are also shown and compared with existing results from the literature.

5.1. Configurations of Both Stages

5.1.1. Upper Stage Settings

The operators of the GA are set using a parametric study, and the hyperparameters are optimized using Bayesian Optimization [134]. Among the obtained settings, the population size is set to 250 individuals and the number of generations is set to 700.

5.1.2. Lower Stage Settings

The lower stage solving the layout task for each container corresponds to the CSO-VF algorithm. For each container, $P = 10$ independently initialized instances solve the corresponding single-container optimal layout problem.

The number of iterations should be adapted along with the occupation rate of the container given that rising occupation rates make it increasingly difficult to solve the constraints. Consequently, the maximum number of iterations allocated for each container is a function of its occupation rate determined by the assignation of the components to the container. The number of iterations varies linearly with the occupation rate and is calculated as follows:

$$N_{it}(OR) = 15{,}000 \cdot OR \tag{46}$$

where N_{it} is the number of iterations and OR is the occupation rate from 0 to 1.

Moreover, the hyperparameters of the CSO-VF algorithm are optimized using Bayesian Optimization, following the same procedure as for the upper stage. The CSO-VF instances dedicated to each container are launched in parallel. Then, all the possible combinations of laid out containers are compared in terms of feasibility and objective function values, and the combination leading to the smaller global inertia corresponds to the output of the algorithm. In the specific case where no feasible combination is found, another assignment list is proposed and the containers' layouts are optimized once again. In this study, the maximum number of attempts is set to 1, i.e., the algorithm must find a solution in one assignment scheme.

5.2. Results and Analysis

The previously detailed algorithm is applied to the described multi-container SMLP and the results are compared to those published in the following:

- Ref. [20]: a dual-system cooperative co-evolutionary algorithm (called Oboe-CCEA) is developed for the multi-container SMLP based on both Potter's coevolutionary framework [89] and the variable-grain model [135]. However, the assignment is an input of the algorithm, i.e., it is not optimized but taken from [108] and based upon human experience. It must be noted that this study does not consider the functional constraints $\mathbf{g}_H$ and $\mathbf{g}_E$. The rest of the mathematical formulation remains identical.
- Ref. [19]: a two-stage algorithm called Dynamic FS is developed to solve both assignment and layout tasks of the multi-container SMLP. Heuristic rules are used to assign the components. They are mathematically translated into a multi-objective optimization problem and an NSGA-II algorithm is employed to solve it. The assignment list is then determined using a fuzzy decision-making method. The layout task is performed using the NDCCDE/DPSO algorithm [112], which corresponds to a dual-system cooperative co-evolutionary algorithm.

5.2.1. Global Performance

The algorithm detailed in Section 4 is run over 50 independently initialized instances. The obtained results are compared with those published in [19,20] according to the following indicators and reported in Table 1:

- The mean value of the final obtained layout objective functions, i.e., their global inertia;
- The standard deviation (STD) of the final obtained layout inertia;
- The best layout obtained in terms of inertia (among the 50 repetitions);
- The worst layout obtained in terms of inertia (among the 50 repetitions);
- The success rate, i.e., the percentage of runs leading to a feasible layout (all the constraints are satisfied).

Table 2 reports the allocated budget in terms of the number of objective function evaluations for the assignment and layout tasks and for the three compared algorithms.

Table 1. Numerical results for the multi-container SMLP obtained thanks to the three compared algorithms. In bold, the best obtained results according to each algorithm. For each metric, bold characters are used to highlight the best obtained value.

Metrics	Oboe-CCEA [20]	Dynamic FS [19]	GA+CSO-VF
Mean of final inertia	718.93	694.06	**682.96**
STD of final inertia	**2.64**	2.96	2.73
Best layout	712.99	689.00	**676.75**
Worst layout	726.59	700.37	**688.45**
Success rate	60%	80%	**100%**

Table 2. Allocated budget in terms of function evaluations for the assignment and layout tasks and the three compared algorithms. ″x″ is written when the metric is irrelevant.

	Oboe-CCEA [20]	Dynamic FS [19]	GA+CSO-VF
Assignment	x	1×10^6	1.75×10^5
Layout	1×10^6	1×10^6	2.7×10^5

The proposed algorithm allows to improve the average final inertia by 5% and 1.6%, respectively, in comparison to the Oboe-CCEA algorithm with the fixed component assignment scheme based on human experience and to the Dynamic FS algorithm with optimized assignment schemes. It also improves the best obtained layout by 5.08% and 1.78%, respectively, in comparison with the two same algorithms. Moreover, it must be highlighted that the worst layout obtained using the GA assisted-CSO-VF algorithm remains better than the best layouts obtained by both the Oboe-CCEA and Dynamic FS algorithms. The success rate reaches 100% for the proposed algorithm against 80% for the Dynamic FS algorithm and 60% for the Oboe-CCEA algorithm. In other words, the proposed algorithm provides a feasible solution for each of the 50 runs. It means that, for each run, at least one feasible layout has been found by the lower stage, corresponding to an optimized assignment list given by the upper stage. On the contrary, a success rate not equal to 100% shows that the corresponding algorithm fails to provide a feasible solution at every run.

However, the Oboe-CCEA has the smaller standard deviation, which is increased by 12.1% by the Dynamic FS algorithm and 3.1% by the proposed algorithm. It should be noted that the objective function considered in this paper is a multimodal function and that the problem geometry involves rotational symmetry. Consequently, local minima can be reached. However, the standard deviation of the final objective function values over the different experiments is quite low, showing that the algorithm finds globally good and relatively equivalent solutions. In addition, as shown in [124,136], the proposed algorithm uses the swap operator as well as multistart to find new configurations in cases of blocked layout and non-symmetric geometries; thus, it manages to escape from local minima.

Finally, Table 2 shows that the proposed algorithm has a lower computational budget in terms of number of objective function evaluations for both assignment and layout tasks than the two other algorithms.

5.2.2. Analysis of the Assignment

This section aims at analyzing the upper stage performance. Table 3 reports the total masses of each container for the fixed assignment list [20], the best assignment list obtained using the Heuristic+NSGA-II algorithm proposed in [19] and the best assignment list obtained using the Genetic Algorithm as detailed in Section 4.2 of this paper. Table 4 reports the coordinates of the center of gravity along the z axis considering only the components or the whole module, i.e., the components in addition to the empty module, and for the three aforementioned assignment lists.

Table 3. Masses distributions on surfaces.

Reference	Surface 1	Surface 2	Surface 3	Surface 4
Fixed [20]	150.63	231.18	235.22	198.42
Heuristic+NSGA-II [19]	88.33	314.20	317.29	95.63
GA (This paper)	85.95	289.02	326.67	113.78

Table 4. Center of gravity along z axis without and with the empty module. Coordinate along z axis of the center of gravity of the empty module: 553.56.

z_{CG}	Fixed [20]	Heuristic + NSGA-II [19]	Proposed Method
Components	543.02	562.9	550.92
Whole module	547.38	559.03	552.01

For the three assignment schemes, it can be noticed that surfaces 2 and 3 are always the heaviest. This contributes to minimize the global inertia along the z axis. Then, the bottom plate, i.e., surfaces S_3 and S_4, is more loaded than the top plate, i.e., surfaces S_1 and S_2. Indeed, the geometrical center between the two plates is situated at the coordinate of 565 along the z axis, but all the assignment schemes have centers of gravity located lower. The assignment scheme proposed in [19] has the most balanced two plates. This is due to the fact that the heuristic assignment rules described in this paper impose that the two plates are both balanced (with a delta) in terms of mass and component area. This results in positioning the center of gravity of the components close to the geometrical center of the two plates, i.e., close to coordinate 565. However, as the two plates are not centered at the geometrical center of the empty module in the INTELSAT-III model, this results in pulling away the center of gravity of the components from the center of gravity of the empty module. Then, the assignment task resolution proposed in this paper results in the positioning of the components on the two plates such that both the center of gravity of the components and of the empty module coincide. This should provide a better global inertia in the end as well as a more balanced satellite module.

Figure 11 shows the median and inter quartile range (IQR) of 50 independently initialized convergence curves of the upper level. The IQR is used to evaluate the dispersion of the results.

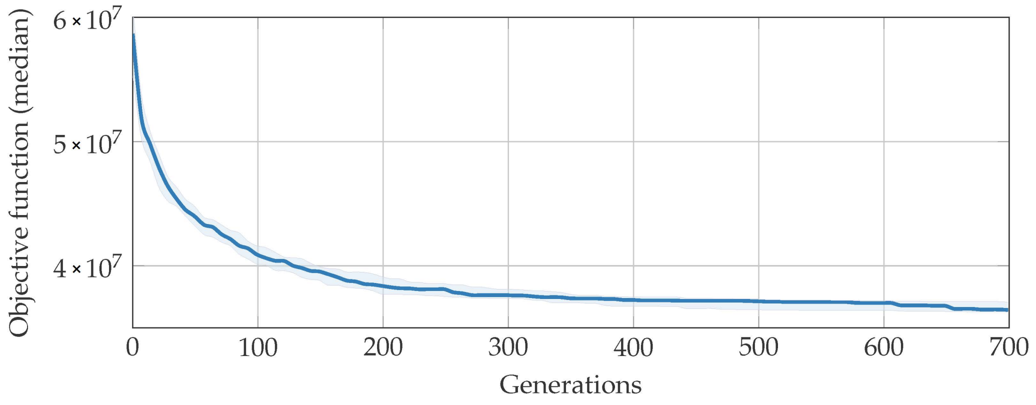
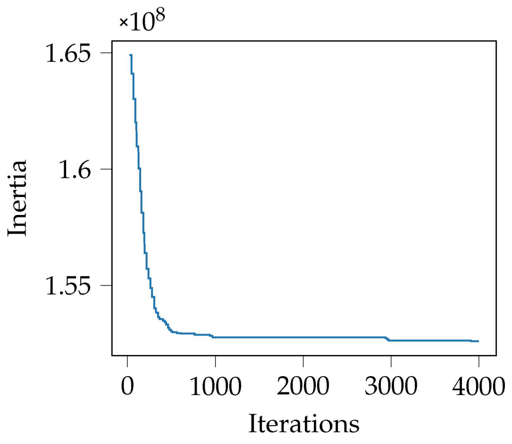


Figure 11. Median and IQR of convergence curves of the upper level for 50 independent runs.

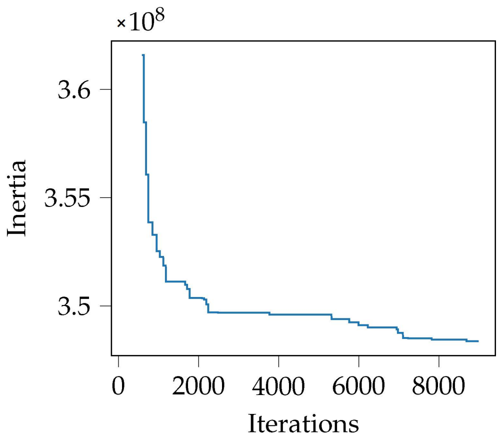
It is notably observed that the IQR remains low during all the optimization process, which characterizes a good robustness of the upper level with respect to the final interquartile range.

5.2.3. Analysis of the Layout

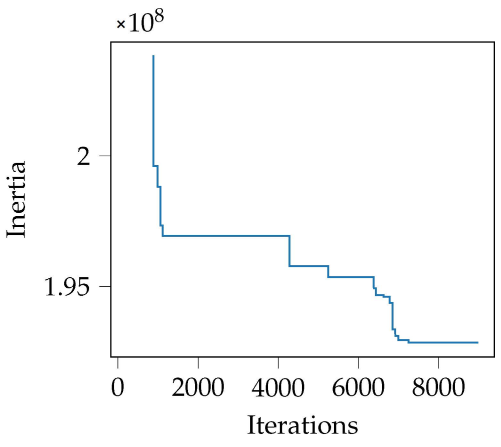
The convergence curves related to each of the containers and for the best optimal layout (among the 50 repetitions) are shown in Figure 12.



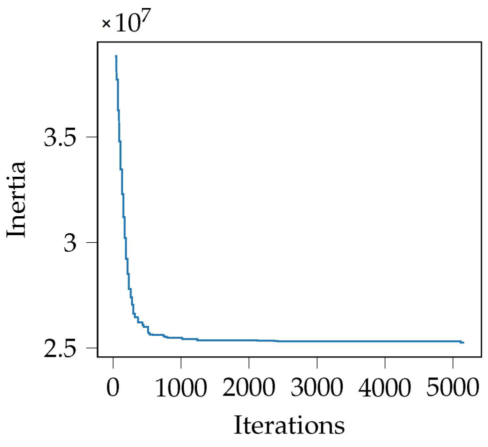
(**a**) Surface 1.



(**b**) Surface 2.



(**c**) Surface 3.



(**d**) Surface 4.

Figure 12. Convergence curves for the four containers of the satellite module.

Figure 12 shows that the convergence speed is higher for Surface 1 and Surface 4 than for Surface 2 and Surface 3. Indeed, as illustrated in Figures 13 and 14, the Surfaces 2 and 3 located near the geometrical center of the satellite module have the highest occupation rates due to the assignation of the components to the surface in order to minimize the terms of the inertia equations along the z axis. The higher the occupation rates, the more difficult it is to solve the constraints. Consequently, the number of iterations is increased and the convergence speed is slower. Moreover, for the same reasons, the first feasible layout is found in less iterations for the less occupied surfaces (1 and 2).

Figure 13 shows the best obtained layout using the Oboe-CCEA algorithm [19] and the proposed algorithm. Figure 14 illustrates the best obtained layout using the Dynamic FS algorithm [19] and the proposed algorithm. In both figures, the functionally incompatible components are highlighted in orange and blue for the heat constraints and orange for the electromagnetic constraint. Moreover, the redder the components, the heaviest their corresponding masses.

5.2.4. Global Analysis

The proposed two-stage algorithm combining Genetic Algorithm and CSO-VF provides general better performance than the Oboe-CCEA and Dynamic FS algorithm regarding two aspects:

- The assignment scheme: The proposed assignment task resolution allows to position the components such that their center of gravity coincides with the center of gravity of the empty module, which contributes to a better global inertia.
- The layout: As highlighted in [124], the CSO-VF outperforms the population-based counterparts like GAs thanks to its ability to solve the constraints with dedicated operators. Indeed, Figure 14 shows that the CSO-VF algorithm systematically positions the heavier components closer to the center of the surfaces, which contributes to minimize the global inertia of each container. On the contrary, it is observed that the Dynamic FS algorithm sometimes positions small and light components around the central bus.

However, despite the highlighted strengths of the proposed algorithm, some limitations can also be identified. First of all, the structure of the algorithm requires that the assignment and layout tasks can be addressed separately, which is not necessarily the case for all multi-container optimal layout problems. Moreover, the upper stage also supposes that the assignment task can be mathematically formulated, which involves problem-specific knowledge in this paper. Finally, in its current formulation, the algorithm is unable to deal with some constraints that involve several surfaces as their layouts are solved in independent optimization processes. For instance, overlapping constraints between components positioned on facing surfaces as well as functional constraints with three-dimensional safety zones could not be handled.

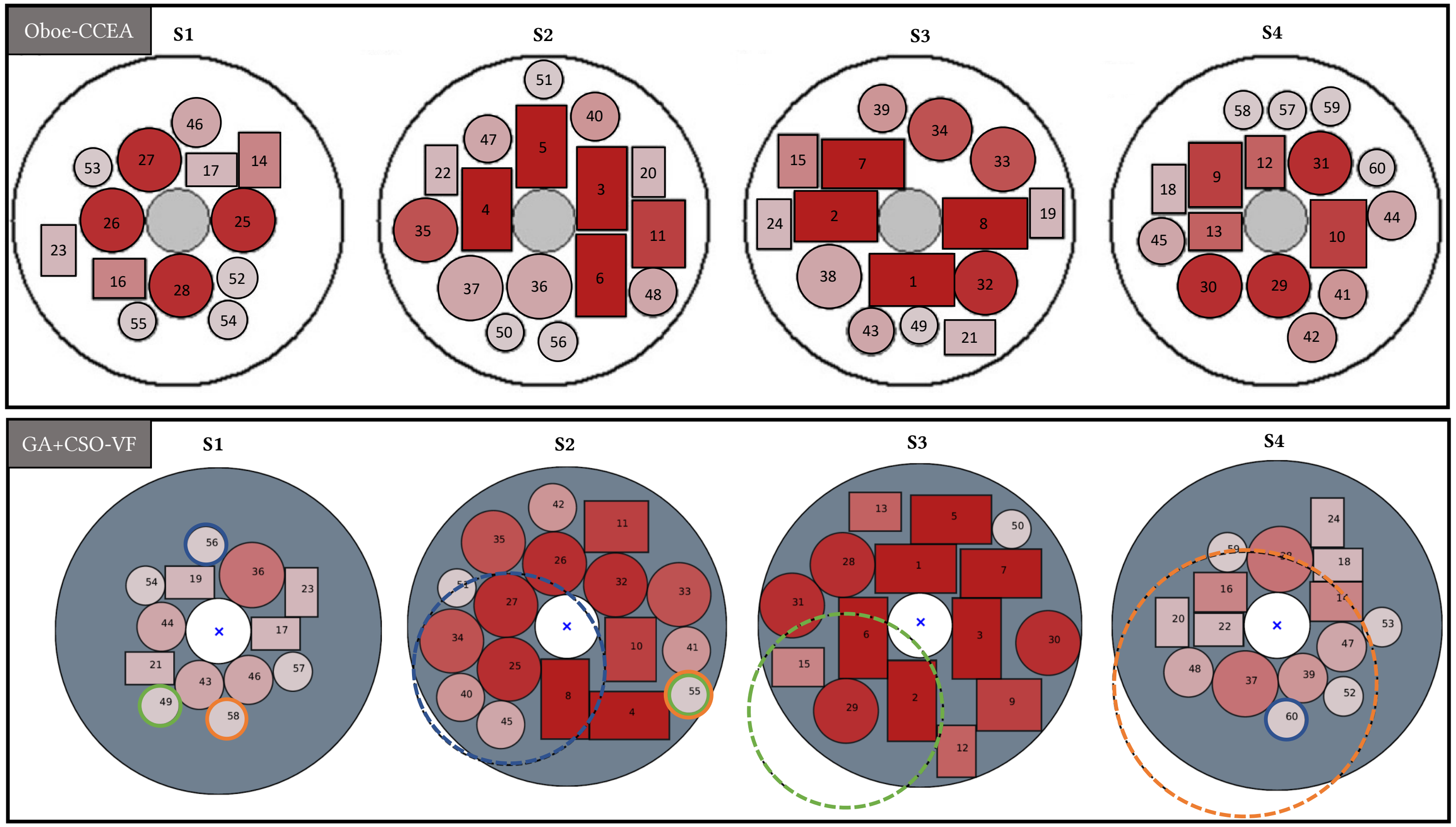


Figure 13. Best final layouts obtained with the Oboe-CCEA algorithm [20] and the proposed algorithm. The redder the components, the higher their corresponding masses. The functionally incompatible components are highlighted in orange and blue for the heat constraints and orange for the electromagnetic constraint. The redder the components, the heaviest their corresponding masses.

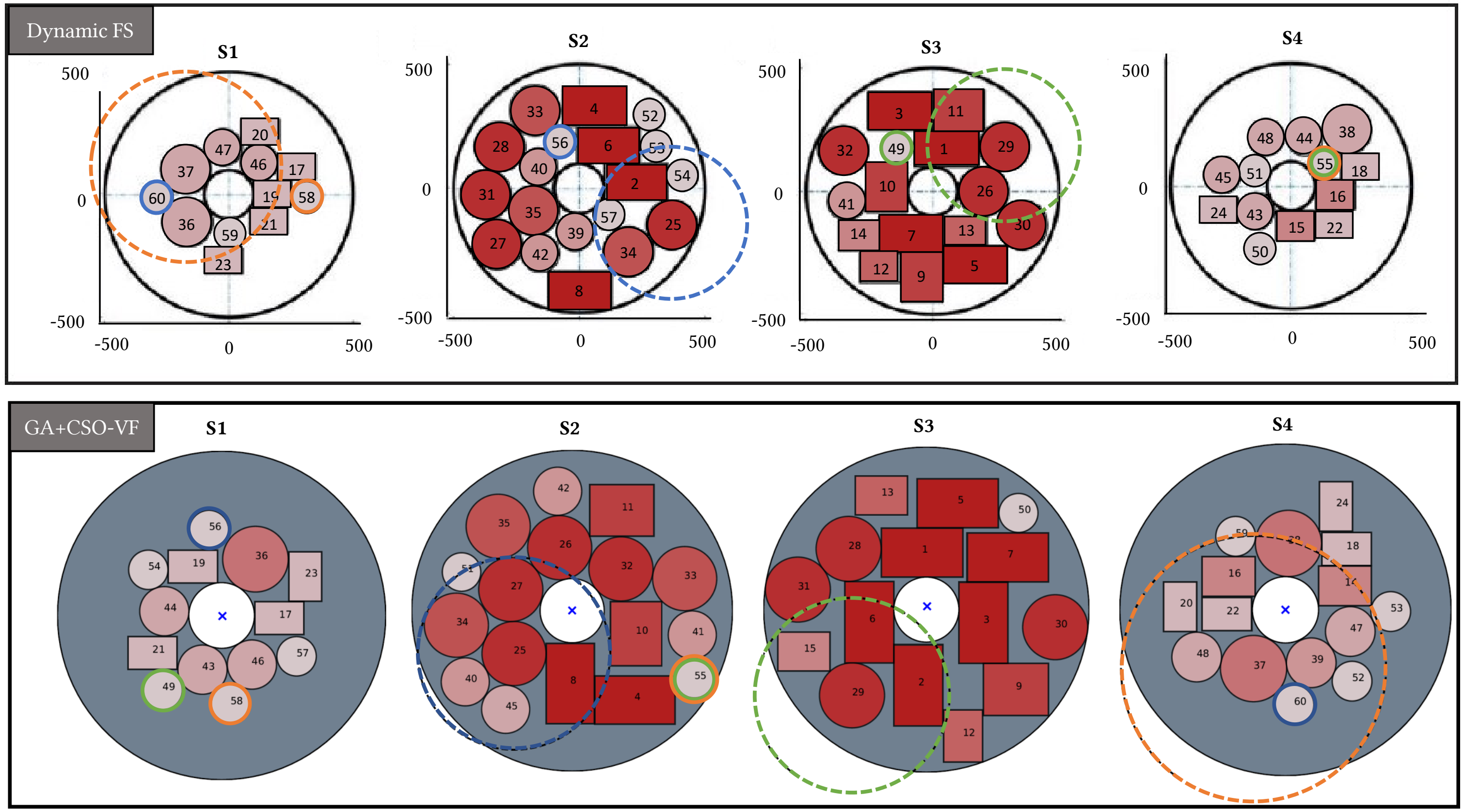


Figure 14. Best final layouts obtained with the Dynamic FS algorithm [19] and the proposed algorithm. The redder the components, the higher their corresponding masses. The functionally incompatible components are highlighted in orange and blue for the heat constraints and orange for the electromagnetic constraint. The redder the components, the heaviest their corresponding masses.

6. Conclusions

In this paper, a two-stage algorithm based on a Genetic Algorithm (GA) and a quasi-physical approach based on a virtual force system (CSO-VF) is proposed for solving component assignment and layout optimization. The multi-container satellite module layout problem was detailed in order to evaluate the performance of the proposed two-stage algorithm in comparison with previous published results. The GA upper stage responsible for the assignation task was configured using an analysis of the objective function inertia equations. The CSO-VF was employed as a lower stage to solve the layout of the components within the four assigned surfaces. This algorithm based on a quasi-physical approach provides constraint-handling abilities thanks to its dedicated virtual-forces system and operators.

The proposed two-stage algorithm outperforms the previously published results in terms of success rate, convergence accuracy and best layout obtained in less objective function evaluations. The analysis of the equations and the aforementioned strengths of the CSO-VF algorithm allowed to find both a satisfying assignment scheme and layouts.

Further studies will focus extending the virtual-force system of the CSO-VF algorithm in order to inherently deal with multi-container configurations. Moreover, it would be interesting to apply the proposed approach to other application cases.

Author Contributions: Conceptualization: J.G., M.B., R.W., A.T. and N.M.; Methodology: J.G., M.B., R.W. and A.T.; Software: J.G., M.B., R.W. and A.T.; Validation: J.G., M.B., R.W. and A.T.; Formal analysis: J.G., M.B., R.W. and A.T.; Resources, J.G., M.B., R.W. and A.T.; data curation, J.G., M.B., R.W. and A.T.; writing—original draft preparation, J.G., M.B., R.W. and A.T.; writing—review and editing, J.G., M.B., R.W., A.T. and N.M.; visualization, J.G., M.B., R.W., A.T. and N.M.; supervision, M.B., R.W. and A.T.; project administration, M.B. and N.M.; funding acquisition, M.B., R.W. and N.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work is part of a PhD thesis cofunded by INRIA, Franceand ONERA, France.

Data Availability Statement: Data will be made available on reasonable request.

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

Appendix A. Components and Containers

The components used for the multi-container satellite layout benchmark are listed in Table A1.

Table A1. Components for the fixed search space multi-container satellite benchmark.

Index	Geometry	Dimension 1	Dimension 2	Height	Mass
1	Cuboid	250	150	250	28.13
2	Cuboid	250	150	250	28.13
3	Cuboid	250	150	250	28.13
4	Cuboid	250	150	250	28.13
5	Cuboid	250	150	250	28.13
6	Cuboid	250	150	250	28.13
7	Cuboid	250	150	250	28.13
8	Cuboid	250	150	250	28.13
9	Cuboid	200	160	200	19.2
10	Cuboid	200	160	200	19.2

Table A1. *Cont.*

Index	Geometry	Dimension 1	Dimension 2	Height	Mass
11	Cuboid	200	160	200	19.2
12	Cuboid	160	120	250	15.36
13	Cuboid	160	120	250	15.36
14	Cuboid	160	120	150	8.64
15	Cuboid	160	120	150	8.64
16	Cuboid	160	120	150	8.64
17	Cuboid	150	100	100	5.40
18	Cuboid	150	100	100	5.40
19	Cuboid	150	100	100	5.40
20	Cuboid	150	100	100	5.40
21	Cuboid	150	100	100	5.40
22	Cuboid	150	100	100	5.40
23	Cuboid	150	100	100	5.40
24	Cuboid	150	100	100	5.40
25	Cylinder	100		250	23.56
26	Cylinder	100		250	23.56
27	Cylinder	100		250	23.56
28	Cylinder	100		250	23.56
29	Cylinder	100		250	23.56
30	Cylinder	100		250	23.56
31	Cylinder	100		250	23.56
32	Cylinder	100		250	23.56
33	Cylinder	100		200	18.85
34	Cylinder	100		200	18.85
35	Cylinder	100		200	18.85
36	Cylinder	100		160	15.08
37	Cylinder	100		160	15.08
38	Cylinder	100		160	15.08
39	Cylinder	75		160	8.48
40	Cylinder	75		160	8.48
41	Cylinder	75		160	8.48
42	Cylinder	75		160	8.48
43	Cylinder	75		150	7.95
44	Cylinder	75		150	7.95
45	Cylinder	75		150	7.95
46	Cylinder	75		150	7.95
47	Cylinder	75		150	7.95
48	Cylinder	75		150	7.95
49	Cylinder	60		150	5.09
50	Cylinder	60		150	5.09

Table A1. *Cont.*

Index	Geometry	Dimension 1	Dimension 2	Height	Mass
51	Cylinder	60		150	5.09
52	Cylinder	60		150	5.09
53	Cylinder	60		150	5.09
54	Cylinder	60		150	5.09
55	Cylinder	60		150	5.09
56	Cylinder	60		150	5.09
57	Cylinder	60		150	5.09
58	Cylinder	60		150	5.09
59	Cylinder	60		150	5.09
60	Cylinder	60		150	5.09

The functional constraint imposes that pairs of components must be positioned a certain distance apart. The indices of the pairs of components, their types and the minimal distance between them is reported in Table A2.

Table A2. Functional constraint for the fixed search space multi-container satellite benchmark.

Index 1	Index 2	Type	Distance
25	56	Heat	200
25	60	Heat	200
29	49	Heat	200
29	55	Heat	200
37	55	Electromagnetic	300
37	58	Electromagnetic	300

The geometrical and dynamical features of the container illustrated in Figure A1 are summarized below.

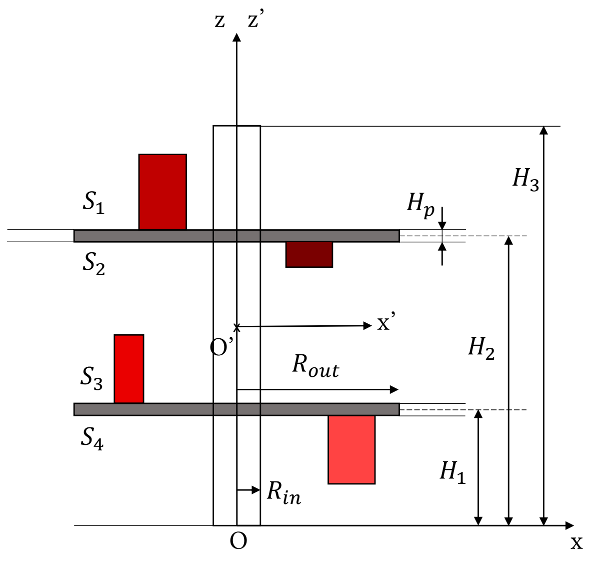


Figure A1. Geometry of the satellite module.

$$H_1 = 300 \text{ mm} \tag{A1}$$

$$H_2 = 830 \text{ mm} \tag{A2}$$

$$H_3 = 1150 \text{ mm} \tag{A3}$$

$$H_p = 20 \text{ mm} \tag{A4}$$

$$R_{in} = 100 \text{ mm} \tag{A5}$$

$$R_{out} = 500 \text{ mm} \tag{A6}$$

Moreover, the empty module is defined by the following:

- Its mass: 576.53 kg;
- Its center of mass located at $(0, 0, 553.56)$ in the $Oxyz$ system of coordinates;
- Its matrix of inertia:

$$I_0 = \begin{bmatrix} 352.2 & 0 & 0 \\ 0 & 352.2 & 0 \\ 0 & 0 & 106.8 \end{bmatrix}$$

Appendix B. Correction of Inertia Equations

The inertia equations are expressed in the system of coordinates attached to the current centroid of the system. However, it has been shown that the equations should rather be expressed in the system of coordinates attached to either the theoretical centroid of the system or to the geometrical center of the container if no balancing constraints are considered.

With the simplified model of the INTELSAT-III considered as a benchmark in this thesis, the inertia should then be expressed in the system of coordinates with its origin being the center of mass of the empty module and denoted $O_{sat}x_{sat}y_{sat}z_{sat}$, as illustrated in Figure A2.

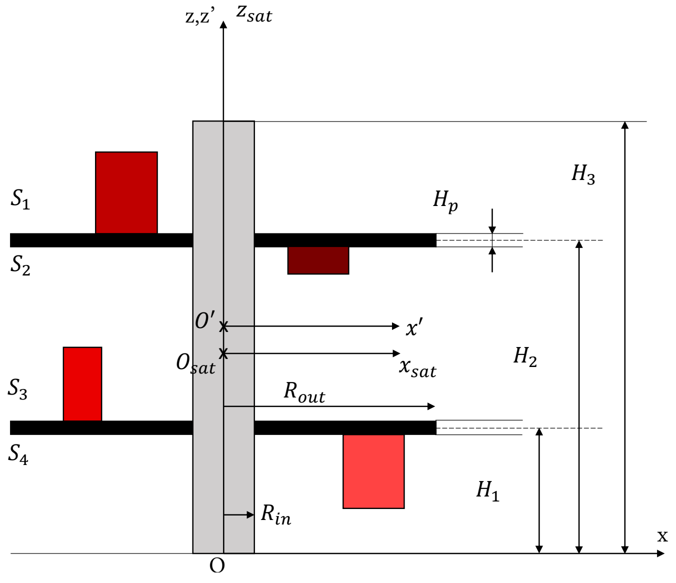


Figure A2. Simplified model of the INTELSAT-III satellite module.In both figures, the functionally incompatible components are highlighted in orange and blue for the heat constraints and orange for the electromagnetic constraint. Moreover, the redder the components, the heaviest their corresponding masses.

Thus, the inertia equations for a component i in the $O_{sat}x_{sat}y_{sat}z_{sat}$ system of coordinates are expressed as follows:

$$I_{x_{sat},i} = I_{x'',i}\cos(p^d_{\alpha,i})^2 + I_{y'',i}\sin(p^d_{\alpha,i})^2 + m_i({p^c_{y,i}}^2 + {p^d_{z,i}}^2) \tag{A7}$$

$$I_{y_{sat},i} = I_{y'',i}\cos(p^d_{\alpha,i})^2 + I_{x'',i}\sin(p^d_{\alpha,i})^2 + m_i({p^c_{x,i}}^2 + {p^d_{z,i}}^2) \tag{A8}$$

$$I_{z_{sat},i} = I_{z'',i} + m_i({p^c_{x,i}}^2 + {p^c_{y,i}}^2) \tag{A9}$$

where $\{p^c_{x,i}, p^c_{y,i}, p^d_{z,i}\}$ are the coordinates of the ith component's center of inertia in the $O_{sat}x_{sat}y_{sat}z_{sat}$ system of coordinates. $p^d_{\alpha,i}$ corresponds to its orientation.

If N components have to be laid out in the container, the inertias of the module along each axis calculated at the point O_{sat} are expressed as follows:

$$I_x = I_{x_{sat},0} + \sum_{i=1}^{N} I_{x_{sat},i} \tag{A10}$$

$$I_y = I_{y_{sat},0} + \sum_{i=1}^{N} I_{y_{sat},i} \tag{A11}$$

$$I_z = I_{z_{sat},0} + \sum_{i=1}^{N} I_{z_{sat},i} \tag{A12}$$

where $I_{x_{sat},0}, I_{y_{sat},0}, I_{z_{sat},0}$ are the inertias of the empty module calculated in its local system of coordinates, which coincide with $O_{sat}x_{sat}y_{sat}z_{sat}$.

Appendix C. Swap

The minimal step between two iterations during which the swap is called is s_{min} and the maximum step is s_{max}. Each time the swap operator is used, it can only be applied at a subsequent iteration, which is calculated based on s_{min} and s_{max} with the following procedure:

- While no feasible layout is found, the step is fixed to s_{min}. Then, at any current iteration t_c during which the swap operator is employed, the next swap iteration noted t_{swap} is calculated as

  $$t_{swap} = t_c + s_{min} \tag{A13}$$

  Consequently, the swap operator is called at a lower frequency and helps to find a feasible solution.
- Once a feasible solution is found, each time the swap operator is called, the convergence curve is interpolated and the next iteration of the swap operator is calculated as

  $$t_{swap} = t + (1-\gamma)s_{min} + \gamma s_{max} \tag{A14}$$

  where γ is a factor characterizing the convergence state as

  $$\gamma = -\frac{\arctan\left(\left.\frac{\mathrm{d}f_{interp}(t)}{\mathrm{d}t}\right|_{t=t_c}\right)}{\pi/2} \tag{A15}$$

  where f_{interp} is the interpolation function of the convergence curve. Therefore, at the beginning of the convergence, the derivative of the convergence curve might be high with γ close to 1; thus, the step size is close to s_{max}. On the contrary, when the convergence stagnates, the factor γ may be close to 0 and the swap is called more frequently. Consequently, as soon as a feasible solution is found, the swap operator is employed to explore new configurations when convergence is stagnating and when the layout is not improving anymore.

Algorithm A1 gives the pseudo code of the swap operator in the case where the swap operator is used to improve the objective function f_{obj} to minimize. It can be summarized as follows: if the current iteration corresponds to t_{swap}, the swap operator is called. If the objective function is improved and if the corresponding constraint violation CV_{Swap} of the N_{cst} constraints' functions ($\mathbf{g}_k$ for $k \in \{1, ..., N_{cst}\}$) remains lower than or equal to the previous constraint violation CV_{NoSwap} times the relaxation factor r, then the swap between the two components occurs.

Algorithm A1 The Swap Operator

```
Input: $t_{swap}$, $r$, $\mathbf{p}$
Output: new vector of positions $\mathbf{p}$
if $t$ is $t_{swap}$ then
  for $i = 0$ to $N$ do
    for $j = i + 1$ to $N$ do
      $\mathbf{x}_{p,swap} \leftarrow \mathbf{x}_p$ with $\mathbf{x}_{p,i}$ and $\mathbf{x}_{p,j}$ exchanged.
      $CV_{Swap} \leftarrow 0$
      $CV_{NoSwap} \leftarrow 0$
      for $k = 0$ to $N_k$ do
        $CV_{Swap} = CV_{Swap} + \mathbf{g}_k(\mathbf{x}_{p,swap})$ #Constraint violation if the swap occurs
        $CV_{NoSwap} = CV_{NoSwap} + \mathbf{g}_k(\mathbf{x}_p)$ #Constraint violation if not
      end for
      if $f_{obj}(\mathbf{x}_{p,swap}) < f_{obj}(\mathbf{x}_p)$ and $CV_{Swap} \leq r \times CV_{NoSwap}$ then
        $\mathbf{x}_p \leftarrow \mathbf{x}_{p,swap}$
      end if
    end for
  end for
  Update $t_{swap}$
end if
```

References

1. Deremaux, Y.; Pietremont, N.; Negrier, J.; Herbin, E.; Ravachol, M. Environmental MDO and uncertainty hybrid approach applied to a supersonic business jet. In Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, BC, Canada, 10–12 September 2008; p. 5832.
2. Cagan, J.; Shimada, K.; Yin, S. A survey of computational approaches to three-dimensional layout problems. *Comput.-Aided Des.* **2002**, *34*, 597–611. [CrossRef]
3. He, K.; Mo, D.; Ye, T.; Huang, W. A coarse-to-fine quasi-physical optimization method for solving the circle packing problem with equilibrium constraints. *Comput. Ind. Eng.* **2013**, *66*, 1049–1060. [CrossRef]
4. Jacquenot, G. Méthode générique pour l'optimisation d'agencement géométrique et fonctionnel. Doctoral Dissertation, Ecole Centrale de Nantes (ECN), Nantes, France, 2010. (In French)
5. Peralta, J.; Andretta, M.; Oliveira, J. Packing Circles and Irregular Polygons using Separation Lines. In Proceedings of the International Conference on Operations Research and Enterprise Systems, 7th International Conference on Operations Research and Enterprise Systems (ICORES 2018), Funchal, Portugal, 24–26 January 2018; pp. 71–77.
6. Benabes, J.; Bennis, F.; Poirson, E.; Ravaut, Y. An interactive-based approach to the layout design optimization. In *Global Product Development, Proceedings of the 20th CIRP Design Conference, Ecole Centrale de Nantes, Nantes, France, 19–21 April 2010*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 511–520.
7. Meng, X.; Sun, H.; Kang, J. Equipment layout optimization based on human reliability analysis of cabin environment. *J. Mar. Sci. Eng.* **2021**, *9*, 1263. [CrossRef]
8. Tongur, V.; Hacibeyoglu, M.; Ulker, E. Solving a big-scaled hospital facility layout problem with meta-heuristics algorithms. *Eng. Sci. Technol.* **2020**, *23*, 951–959. [CrossRef]
9. Feng, J.; Shen, W. Wind farm layout optimization in complex terrain: A preliminary study on a Gaussian hill. *J. Phys. Conf. Ser.* **2014**, *524*, 012146. [CrossRef]
10. Ghaisas, N.; Archer, C. Geometry-based models for studying the effects of wind farm layout. *J. Atmos. Ocean. Technol.* **2016**, *33*, 481–501. [CrossRef]
11. Wang, Y.; Liu, H.; Long, H.; Zhang, Z.; Yang, S. Differential evolution with a new encoding mechanism for optimizing wind farm layout. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1040–1054. [CrossRef]
12. Phan, N.; Bui, T.; Jiang, H.; Li, P.; Pan, Z.; Liu, N. Coverage optimization of LTE networks based on antenna tilt adjusting considering network load. *China Commun.* **2017**, *14*, 48–58. [CrossRef]

13. Soman, R.; Kudela, P.; Balasubramaniam, K.; Singh, S.; Malinowski, P. A study of sensor placement optimization problem for guided wave-based damage detection. *Sensors* **2019**, *19*, 1856. [CrossRef]
14. Zhao, Q.; Li, C.; Zhu, D.; Xie, C. Coverage optimization of wireless sensor networks using combinations of PSO and chaos optimization. *Electronics* **2022**, *11*, 853. [CrossRef]
15. Man, K.; Tang, K.; Kwong, S. Genetic algorithms: Concepts and applications [in engineering design]. *IEEE Trans. Ind. Electron.* **1996**, *43*, 519–534. [CrossRef]
16. Sun, Z.; Teng, H. Optimal layout design of a satellite module. *Eng. Optim.* **2003**, *35*, 513–529. [CrossRef]
17. Xu, Y.; Xiao, R.; Amos, M. A novel genetic algorithm for the layout optimization problem. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 3938–3943.
18. Jacquenot, G.; Dennis, F.; Maisonneuve, J.; Wenger, P. 2d multi-objective placement algorithm for free-form components. *Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.* **2009**, *49026*, 239–248.
19. Cui, F.Z.; Zhong, C.Q.; Wang, X.-K.; Teng, H.-F. A collaborative design method for satellite module component assignment and layout optimization. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2019**, *233*, 5471–5491. [CrossRef]
20. Teng, H.; Chen, Y.; Zeng, Y.; Shi, Y.; Hu, Q. A dual-system variable-grain cooperative coevolutionary algorithm: Satellite-module layout design. *IEEE Trans. Evol. Comput.* **2009**, *14*, 438–455. [CrossRef]
21. Wang, Y.; Teng, H.; Shi, Y. Cooperative co-evolutionary scatter search for satellite module layout design. *Eng. Comput.* **2009**, *26*, 761–785. [CrossRef]
22. Bliek1ú, C.; Bonami, P.; Lodi, A. Solving mixed-integer quadratic programming problems with IBM-CPLEX: A progress report. In Proceedings of the Twenty-Sixth RAMP Symposium, Tokyo, Japan, 16–17 October 2014; pp. 16–17.
23. Costa, A.; Hansen, P.; Liberti, L. On the impact of symmetry-breaking constraints on spatial branch-and-bound for circle packing in a square. *Discret. Appl. Math.* **2013**, *161*, 96–106. [CrossRef]
24. Kohara, D.; Yamamoto, H.; Suzuki, A. Efficient algorithms based on branch and bound methods for multi floor facility layout problems. In Proceedings of the 9th Asia Pacific Industrial Engineering & Management Systems Conference, Bali, Indonesiam, 3–5 December 2008; pp. 387–395.
25. Xie, W.; Sahinidis, N. A branch-and-bound algorithm for the continuous facility layout problem. *Comput. Chem. Eng.* **2008**, *32*, 1016–1028. [CrossRef]
26. Cintra, G.; Miyazawa, F.; Wakabayashi, Y.; Xavier, E. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *Eur. J. Oper. Res.* **2008**, *191*, 61–85. [CrossRef]
27. Cui, Y.; Huang, L. Dynamic programming algorithms for generating optimal strip layouts. *Comput. Optim. Appl.* **2006**, *33*, 287–301. [CrossRef]
28. Furini, F.; Ljubić, I.; Sinnl, M. An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem. *Eur. J. Oper. Res.* **2017**, *262*, 438–448. [CrossRef]
29. Koopmans, T.; Beckmann, M. Assignment problems and the location of economic activities. *Econom. J. Econom. Soc.* **1957**, 53–76. [CrossRef]
30. Montreuil, B. A Modelling Framework for Integrating Layout Design and Flow Network Design. In Proceedings of the Material Handling Research Colloquium, Hebron, Kentucky, 19–21 June 1990; pp. 43–58.
31. Budianto, F.; Halim, J.; Sembiring, A. Redesigning Furniture Production Floors Using Systematic Layout Planning and ALDEP Method to Minimize Material Handling Costs. In Proceedings of the 2020 3rd International Conference on Mechanical, Electronics, Computer, and Industrial Technology (MECnIT), Medan, Indonesia, 25–27 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 84–90.
32. Cambron, K.; Evans, G. Layout design using the analytic hierarchy process. *Comput. Ind. Eng.* **1991**, *20*, 211–229. [CrossRef]
33. Deshpande, V.; Patil, N.; Baviskar, V.; Gandhi, J. Plant layout optimization using CRAFT and ALDEP methodology. *Product. J. Natl. Product. Counc.* **2016**, *57*, 32–42.
34. Bozer, Y.; Meller, R.; Erlebacher, S. An improvement-type layout algorithm for single and multiple-floor facilities. *Manag. Sci.* **1994**, *40*, 918–932. [CrossRef]
35. Lekan, O.; Kayode, O.; Morenikeji, A.A. Analysis of plant layout design for operational efficiency with craft algorithms. *Acta Univ. Danubius. Acon.* **2017**, *13*, 15–27.
36. Prasad, N.; Rajyalakshmi, G.; Reddy, A. A typical manufacturing plant layout design using CRAFT algorithm. *Procedia Eng.* **2014**, *97*, 1808–1814. [CrossRef]
37. Baykasoğlu, A.; Gindy, N. A simulated annealing algorithm for dynamic layout problem. *Comput. Oper. Res.* **2001**, *28*, 1403–1426. [CrossRef]
38. Burkard, R.; Rendl, F. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *Eur. J. Oper. Res.* **1984**, *17*, 169–174. [CrossRef]
39. Meller, R.; Bozer, Y. A new simulated annealing algorithm for the facility layout problem. *Int. J. Prod. Res.* **1996**, *34*, 1675–1692. [CrossRef]
40. Tam, K. A simulated annealing algorithm for allocating space to manufacturing cells. *Int. J. Prod. Res.* **1992**, *30*, 63–87. [CrossRef]
41. Carrabs, F.; Cerrone, C.; Cerulli, R. A tabu search approach for the circle packing problem. In Proceedings of the 2014 17th International Conference on Network-Based Information Systems, Salerno, Italy, 10–12 September 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 165–171.

42. Kaku, B.; Mazzola, J. A tabu-search heuristic for the dynamic plant layout problem. *INFORMS J. Comput.* **1997**, *9*, 374–384. [CrossRef]
43. Liang, L.; Chao, W. The strategies of tabu search technique for facility layout optimization. *Autom. Constr.* **2008**, *17*, 657–669. [CrossRef]
44. Bortfeldt, A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *Eur. J. Oper. Res.* **2006**, *172*, 814–837. [CrossRef]
45. Peng, Y.; Zeng, T.; Fan, L.; Han, Y.; Xia, B. An improved genetic algorithm based robust approach for stochastic dynamic facility layout problem. *Discret. Dyn. Nat. Soc.* **2018**, 1, 1–8. [CrossRef]
46. Yoon, Y.; Kim, Y. An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks. *IEEE Trans. Cybern.* **2013**, *43*, 1473–1483. [CrossRef]
47. Asl, A.D.; Wong, K.; Tiwari, M. Unequal-area stochastic facility layout problems: Solutions using improved covariance matrix adaptation evolution strategy, particle swarm optimisation, and genetic algorithm. *Int. J. Prod. Res.* **2016**, *54*, 799–823.
48. Liu, J.; Zhang, H.; He, K.; Jiang, S. Multi-objective particle swarm optimization algorithm based on objective space division for the unequal-area facility layout problem. *Expert Syst. Appl.* **2018**, *102*, 172–192. [CrossRef]
49. Xiao, R.; Xu, Y.; Amos, M. Two hybrid compaction algorithms for the layout optimization problem. *BioSystems* **2007**, *90*, 560–567. [CrossRef]
50. Wagner, M.; Veeramachaneni, K.; Neumann, F.; O'Reilly, U. Optimizing the layout of 1000 wind turbines. *Eur. Wind Energy Assoc. Annu. Event* **2011**, *205209*.
51. Lee, Y.; Lee, M. A shape-based block layout approach to facility layout problems using hybrid genetic algorithm. *Comput. Ind. Eng.* **2002**, *42*, 237–248. [CrossRef]
52. Lien, L.; Cheng, M. A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization. *Expert Syst. Appl.* **2012**, *39*, 9642–9650. [CrossRef]
53. Lim, Z.Y.; Ponnambalam, S.G.; Izui, K. Multi-objective hybrid algorithms for layout optimization in multi-robot cellular manufacturing systems. *Knowl.-Based Syst.* **2017**, *120*, 87–98. [CrossRef]
54. Chen, X.; Yao, W.; Zhao, Y.; Chen, X.; Zhang, J.; Luo, Y. The hybrid algorithms based on differential evolution for satellite layout optimization design. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; IEEE: Piscataway, NJ, USA 2018; pp. 1–8.
55. He, K.; Ye, H.; Wang, Z.; Liu, J. An efficient quasi-physical quasi-human algorithm for packing equal circles in a circular container. *Comput. Oper. Res.* **2018**, *92*, 26–36. [CrossRef]
56. Huang, W.; Ye, T. Global optimization method for finding dense packings of equal circles in a circle. *Eur. J. Oper. Res.* **2011**, *210*, 474–481. [CrossRef]
57. Wang, H.; Huang, W.; Zhang, Q.; Xu, D. An improved algorithm for the packing of unequal circles within a larger containing circle. *Eur. J. Oper. Res.* **2002**, *141*, 440–453. [CrossRef]
58. Burggraäf, P.; Wagner, J.; Heinbach, B. Bibliometric study on the use of machine learning as resolution technique for facility layout problems. *IEEE Access* **2021**, *9*, 22569–22586. [CrossRef]
59. Sajedi, S.; Liang, X. Deep generative Bayesian optimization for sensor placement in structural health monitoring. *Comput. Civ. Infrastruct. Eng.* **2022**, *37*, 1109–1127. [CrossRef]
60. Tsuchiya, K.; Bhariktar, S.; Takefuji, Y. A neural network approach to facility layout problems. *Eur. J. Oper. Res.* **1996**, *89*, 556–5563. [CrossRef]
61. Vashisht, D.; Rampal, H.; Liao, H.; Lu, Y.; Shandbhag, D.; Fallon, E.; Kara, L. Placement integrated circuits using cyclic reinforcement learning and simulated annealing. *arXiv* **2020**, arXiv:2011.07577.
62. Salcedo-Sanz, S. A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Comput. Sci. Rev.* **2009**, *3*, 175–192. [CrossRef]
63. Mezura-Montes, E.; Coello, C. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm Evol. Comput.* **2011**, *1*, 173–194. [CrossRef]
64. Krasnogor, N.; Smith, J. A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Trans. Evol. Comput.* **2005**, *9*, 474–488. [CrossRef]
65. Kusiak, A.; Heragu, S. The facility layout problem. *Eur. J. Oper. Res.* **1987**, *29*, 229–251. [CrossRef]
66. Zhou, J.; Love, P.; Teo, K.; Luo, H. An exact penalty function method for optimising QAP formulation in facility layout problem. *Int. J. Prod. Res.* **2017**, *55*, 2913–2929. [CrossRef]
67. Barbosa-Póvoa, A.; Mateus, R.; Novais, A. Optimal 3D layout of industrial facilities. *Int. J. Prod. Res.* **2002**, *40*, 1669–1698. [CrossRef]
68. Georgiadis, M.; Schilling, G.; Rotstein, G.; Macchietto, S. A general mathematical programming approach for process plant layout. *Comput. Chem. Eng.* **1999**, *23*, 823–840. [CrossRef]
69. Patsiatzis, D.; Papageorgiou, L. Optimal multi-floor process plant layout. *Comput. Chem. Eng.* **2002**, *26*, 575–583. [CrossRef]
70. Lee, R. CORELAP-computerized relationship layout planning. *J. Ind. Eng.* **1967**, *8*, 195–200.
71. Johnson, R. SPACECRAFT for multi-floor layout planning. *Manag. Sci.* **1982**, *28*, 407–417. [CrossRef]
72. Alvarez-Valdés, R.; Pareño, F.; Tamarit, J. Reactive GRASP for the strip-packing problem. *Comput. Oper. Res.* **2008**, *35*, 1065–1083. [CrossRef]

73. Delorme, X.; Gandibleux, X.; Rodriguez, J. GRASP for set packing problems. *Eur. J. Oper. Res.* **2004**, *153*, 564–580. [CrossRef]
74. Layeb, A.; Chenche, S. A novel grasp algorithm for solving the bin packing problem. *Int. J. Inf. Eng. Electron.* **2012**, *4*, 8. [CrossRef]
75. Singh, S.; Sharma, R. A review of different approaches to the facility layout problems. *Int. J. Adv. Manuf. Technol.* **2006**, *30*, 425–433. [CrossRef]
76. Cazzaro, D.; Psisinger, D. Variable neighborhood search for large offshore wind farm layout optimization. *Comput. Oper. Res.* **2013**, *8*, 105588. [CrossRef]
77. Herrán, A.; Colmenar, J.; Duarte, A. An efficient variable neighborhood search for the space-free multi-row facility layout problem. *Eur. J. Oper. Res.* **2021**, *295*, 893–907. [CrossRef]
78. Ripon, K.; Glette, K.; Khan, K.; Hovin, M.; Torresen, J. Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities. *Swarm Evol. Comput.* **2013**, *8*, 1–12. [CrossRef]
79. Luo, Q.; Rao, Y.; Peng, D. GA and GWO algorithm for the special bin packing problem encountered in field of aircraft arrangement. *Appl. Soft Comput.* **2022**, *114*, 108060. [CrossRef]
80. Tong, D.; Murray, A.; Xiao, N. Heuristics in spatial analysis: A genetic algorithm for coverage maximization. *Ann. Assoc. Am. Geogr.* **2009**, *99*, 698–711. [CrossRef]
81. Zhao, Y.; Li, M.; Lu, X.; Tian, L.; Yu, Z.; Huang, K.; Li, T. Optimal layout design of obstacles for panic evacuation using differential evolution. *Phys. A: Stat. Mech. Its Appl.* **2017**, *465*, 175–194.
82. Hasan, R.; Mohammed, M.; Ţăpuş, N.; Hammood, O. A comprehensive study: Ant colony optimization (ACO) for facility layout problem. In Proceedings of the 16th RoEduNet Conference: Networking in Education and Research (RoEduNet), Targu-Mures, Romania, 21–23 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–8.
83. Levine, J.; Ducatelle, F. Ant colony optimization and local search for bin packing and cutting stock problems. *J. Oper. Res. Soc.* **2004**, *55*, 705–716. [CrossRef]
84. Kothari, R.; Ghosh, D. A scatter search algorithm for the single row facility layout problem. *J. Heuristics* **2014**, *20*, 125–142. [CrossRef]
85. Wang, Y.; Shi, Y.; Teng, H. An improved scatter search for circles packing problem with the equilibrium constraint. *Chin. J. Comput.* **2009**, *32*, 1214–1221. [CrossRef]
86. Galanos, T.; Liapis, A.; Yannakakis, G.; Koenig, R. ARCH-Elites: Quality-diversity for urban design. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Online, 10–14 July 2021; pp. 313–314.
87. Nikfarjam, A.; Do, A.V.; Neumann, F. Analysis of Quality Diversity Algorithms for the Knapsack Problem. In *Parallel Problem Solving from Nature–PPSN XVII, Proceedings of the 17th International Conference, PPSN 2022, Dortmund, Germany, 10–14 September 2022*; Proceedings, Part II; Springer International Publishing: Cham, Switzerland, 2022; pp. 413–427.
88. Zhang, Y.; Fontaine, M.; Bhatt, V.; Nikolaidis, S.; Li, J. Multi-Robot Coordination and Layout Design for Automated Warehousing. *arXiv* **2023**, arXiv:2305.06436.
89. Potter, M.A.; Jong, D.; Kenneth, A. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 249–257.
90. Ma, X.; Li, X.; Zhang, Q.; Tang, K.; Liang, Z.; Xie, W.; Zhu, Z. A survey on cooperative co-evolutionary algorithms. *IEEE Trans. Evol. Comput.* **2018**, *23*, 421–441. [CrossRef]
91. Dunker, T.; Radons, G.; Westkämper, E. A coevolutionary algorithm for a facility layout problem. *Int. J. Prod. Res.* **2003**, *41*, 3479–3500. [CrossRef]
92. Xu, Y.C.; Dong, F.M.; Liu, Y.; Xiao, R.B.; Amos, M. Ant colony algorithm for the weighted item layout optimization problem. *arXiv* **2010**, arXiv:1001.4099.
93. Guo, Z.; Li, B. Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system. *Front. Archit. Res.* **2017**, *6*, 53–62. [CrossRef]
94. Abdessamia, F.; Zhang, W.; Tian, Y. Energy-efficiency virtual machine placement based on binary gravitational search algorithm. *Clust. Comput.* **2020**, *23*, 1577–1588. [CrossRef]
95. Sajedi, H.; Razavi, S. DGSA: Discrete gravitational search algorithm for solving knapsack problem. *Oper. Res.* **2017**, *17*, 563–591. [CrossRef]
96. Mirhoseini, A.; Goldie, A.; Yazgan, M.; Jiang, J.; Songhori, E.; Wang, S.; Dean, J. A graph placement methodology for fast chip design. *Nature* **2021**, *594*, 207–212. [CrossRef] [PubMed]
97. Deshwal, A.; Belakaria, S.; Doppa, J.; Kim, D.H. Bayesian optimization over permutation spaces. *Proc. AAAI Conf. Artif.* **2022**, *36*, 6515–6523. [CrossRef]
98. Otaki, D.; Nonaka, H.; Yamada, N. Thermal design optimization of electronic circuit board layout with transient heating chips by using Bayesian optimization and thermal network model. *Int. J. Heat Mass Transf.* **2022**, *184*, 122263. [CrossRef]
99. NASA. *System Engineering Handbook*; *NASA SP-2007-6105, Rev 1;* NASA: Washington, DC, USA, 2007.
100. Sousa, F.D.; Galski, R.; Rocco, E.; Becceneri, J.; Santos, W.; Sandri, S. A toll for multidisciplinary design conception of spacecraft equipment layout. In Proceedings of the 22nd International Congress of Mechanical Engineering (COBEM, 2013), Ribeirão Preto, Brazil, 3–7 November 2013.
101. Ferebee, M.; Allen, C. Optimization of payload placement on arbitrary spacecraft. *J. Spacecr. Rocket.* **1991**, *28*, 612–614. [CrossRef]
102. Ferebee, M.; Powers, R. *Optimization of Payload Mass Placement in a Dual Keel Space Station*; NASA Technical Memorandum 89051, March; NASA: Washington, DC, USA, 1987.

103. Wang, Y.; Liu, L.; Xing, Y.; Yang, Z. Investigation of wing structure layout of aerospace plane based on the finite element method. *Adv. Mech. Eng.* **2017**, *9*. [CrossRef]
104. Zhu, J.; Zhang, W.; Xia, L. Topology optimization in aircraft and aerospace structures design. *Arch. Comput. Methods Eng.* **2016**, *23*, 595–622. [CrossRef]
105. Zhu, J.; Guo, W.; Zhang, W.; Liu, T. Integrated layout and topology optimization design of multi-frame and multi-component fuselage structure systems. *Struct. Multidiscip. Optim.* **2017**, *56*, 21–45. [CrossRef]
106. Zhu, J.; Yu, X.; Wang, Y. Layout optimization for blended wing body aircraft structure. *Int. J. Aeronaut. Space Sci.* **2019**, *20*, 879–890. [CrossRef]
107. Teng, H.; Sun, S.; Liu, D.; Li, Y. Layout optimization for the objects located within a rotating vessel–A three-dimensional packing problem with behavioral constraints. *Comput. Oper. Res.* **2001**, *28*, 521–535. [CrossRef]
108. Zhang, B.; Teng, H.; Shi, Y. Layout optimization of satellite module using soft computing techniques. *Appl. Soft Comput.* **2008**, *8*, 507–521. [CrossRef]
109. Hengeveld, D.; Braun, J.; Eckard, A.; Williams, A. Optimal placement of electronic components to mininize heat flux nonuniformities. *J. Spacecr. Rocket.* **2011**, *48*, 556–563. [CrossRef]
110. Jackson, B.; Norgard, J. A stochastic optimization for determining spacecraft avionics box placement. *IEEE Aerosp. Conf.* **2002**, *5*, 2373–2382.
111. der Velden, C.V.; Bil, C.; Yu, X.; Smith, A. An intelligent system for automatic layout routing in aerospace design. *Innov. Syst. Softw. Eng.* **2007**, *3*, 117–128. [CrossRef]
112. Cui, F.; Xu, Z.; Wang, X. A dual-system cooperative co-evolutionary algorithm for satellite equipment layout optimization. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2018**, *232*, 2432–2457. [CrossRef]
113. Grignon, P.; Fadel, G. A GA based configuration design optimization method. *J. Mech. Des.* **2004**, *126*, 6–15. [CrossRef]
114. Shafaee, M.; Mohammadzadeh, P.; Elkaie, A.; Abbasi, S. Layout design optimization of a space propulsion system using hybrid optimization algorithm. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2017**, *231*, 338–349. [CrossRef]
115. Li, Z.; Zeng, Y.; Wang, Y. A hybrid multi-mechanism optimization approach for the payload packing design of a satellite module. *Appl. Soft Comput.* **2016**, *45*, 11–26. [CrossRef]
116. Xu, Z.; Zhong, C.; Teng, H. Assignment and layout integration optimization for simplified satellite re-entry module component layout. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2019**, *233*, 4287–4301. [CrossRef]
117. Liu, J. Constrained Layout Optimization in Satellite Cabin Using a Multiagent Genetic Algorithm. In *Asia-Pacific Conference on Simulated Evolution and Learning*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 440–449.
118. Ahmadi, A.; Jokar, M. An efficient multiple-stage mathematical programming method for advanced single and multi-floor facility layout problems. *Appl. Math. Model.* **2016**, *40*, 5605–5620. [CrossRef]
119. Ahmadi, A.; Pishvaee, M.; Jokar, M. A survey on multi-floor facility layout problems. *Comput. Ind. Eng.* **2017**, *107*, 158–170. [CrossRef]
120. Karateke, H.; Sahin, R.; Niroomand, S. A hybrid Dantzig-Wolfe decomposition algorithm for the multi-floor facility layout problem. *Expert Syst. Appl.* **2022**, *206*, 117845. [CrossRef]
121. Zhang, J.; Guo, J.; Liu, Y.; Feng, Y.; Li, H. Component assignment and layout optimization for multi-module microsatellite considering variable module size. *Acta Astronaut.* **2022**, *198*, 36–44. [CrossRef]
122. Zhong, C.; Xu, Z.; Teng, H. Multi-module satellite component assignment and layout optimization. *Appl. Soft Comput.* **2019**, *75*, 148–161. [CrossRef]
123. Gamot, J.; Balesdent, M.; Tremolet, A.; Wuilbercq, R.; Melab, N.; Talbi, E.G. Hidden-variables genetic algorithm for variable-size design space optimal layout problems with application to aerospace vehicles. *Eng. Appl. Artif. Intell.* **2023**, *121*, 105941. [CrossRef]
124. Gamot, J.; Wuilbercq, R.; Balesdent, M.; Tremolet, A.; Melab, N.; Talbi, E.G. Component Swarm Optimization Using Virtual Forces for Solving Layout Problems. In *Swarm Intelligence, Proceedings of the 13th International Conference, ANTS 2022, Málaga, Spain, 2–4 November 2022*; Springer International Publishing: Cham, Switzerland, 2022; pp. 292–299.
125. Baghel, M.; Agrawal, S.; Silakari, S. Survey of metaheuristic algorithms for combinatorial optimization. *Int. J. Comput. Appl.* **2012**, *58*. [CrossRef]
126. Xie, J.; Wei, D.; Huang, S.; Bu, X. A sensor deployment approach using improved virtual force algorithm based on area intensity for multisensor networks. *Math. Probl. Eng.* **2019**, *1*, 8015309 [CrossRef]
127. Sun, S.; Chen, H.; Lin, J. A Universal Method for Modeling and Characterizing Non-Circular Packing Systems Based on n-Point Correlation Functions. *Materials* **2022**, *15*, 5991. [CrossRef]
128. Fu, Z.; Huang, W.; Lü, Z. Iterated tabu search for the circular open dimension problem. *Eur. J. Oper. Res.* **2013**, *225*, 236–243. [CrossRef]
129. Wang, Y.; Wang, Y.; Sun, J.; Zhang, C.H.X. A stimulus-response-based allocation method for the circle packing problem with equilibrium constraints. *Phys. A Stat. Mech. Its Appl.* **2019**, *522*, 232–247. [CrossRef]
130. Zeng, Z.; Yu, X.; He, K.; Huang, W.; Fu, Z. Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. *Eur. J. Oper. Res.* **2016**, *250*, 616–627. [CrossRef]
131. Damblin, G.; Couplet, M.; Iooss, B. Numerical studies of space-filling designs: Optimization of Latin Hypercube Samples and subprojection properties. *J. Simul.* **2013**, *7*, 276–289. [CrossRef]

132. Li, W.; Lu, L.; Xie, X.; Yang, M. A novel extension algorithm for optimized Latin hypercube sampling. *J. Stat. Comput. Simul.* **2017**, *87*, 2549–2559. [CrossRef]
133. Pronzato, L.; Müller, W. Design of computer experiments: Space filling and beyond. *Stat. Comput.* **2012**, *22*, 681–701. [CrossRef]
134. Shariari, B.; Swersky, K.; Wang, Z.; Adams, R.; Freitas, N.D. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* **2015**, *104*, 148–175. [CrossRef]
135. Nain, P.; Deb, K. A computationally effective multiobjective search and optimization technique using coarse-to-fine grain modeling. In *Proceedings on Parallel Problem Solving Nature Workshop Evolutionary Multiobjective Optimization*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 2081–2088.
136. Gamot, J. Algorithms for Conditional Search Space Optimal Layout Problems. Doctoral Dissertation, Université de Lille, Lille, France, 2023.