Journal of Engineering and Applied Science

RESEARCH **Open Access**

Packing optimization of practical systems using a dynamic acceleration methodology

Christopher Douglas[1], Jae Sung Huh[2], Sang Ook Jun[2] and Il Yong Kim[3*]

*Correspondence:
kimiy@queensu.ca

[1] Queen's University, Kingston, Canada
[2] Korea Aerospace Research Institute, Daejeon, South Korea
[3] Department of Mechanical and Materials Engineering, Queen's University, 130 Stuart Street, Kingston, ON K7L 3N6, Canada

Abstract
System design is a challenging and time-consuming task which often requires close collaboration between several multidisciplinary design teams to account for complex interactions between components and sub-systems. As such, there is a growing demand in industry to create better performing, efficient, and cost-effective development tools to assist in the system design process. Additionally, the ever-increasing complexity of systems today often necessitates a shift away from manual expertise and a movement towards computer-aided design tools. This work narrows the scope of the system design process by focusing on one critical design aspect: the packaging of system components. The algorithm presented in this paper was developed to optimize the packaging of system components with consideration of practical, system-level functionalities and constraints. Using a dynamic acceleration methodology, the algorithm packages components from an initial position to a final packed position inside of a constrained volume. The motion of components from initial to final positions is driven by several acceleration forces imposed on each component. These accelerations are based on physical interactions between components and their surrounding environment. Various system-level performance metrics such as center of mass alignment and rotational inertia reduction are also considered throughout optimization. Results of several numerical case studies are also presented to demonstrate the functionality and capability of the proposed packaging algorithm. These studies include packaging problems with known optimal solutions to verify the efficacy of the algorithm. Finally, the proposed algorithm was used in a more practical study for the packaging of an urban air mobility nacelle to demonstrate the algorithm's prospective capabilities in solving real-world packaging problems.

Keywords: Packaging optimization, Layout optimization, Packing density, Vector fields

Introduction

The design of mechanical systems is often a challenging task with strong incentives and competition within industry to produce cost-effective and best-performing products. This design process often requires several design cycles before a final product is produced and may take a significant about of time and resources per cycle. As a result, there is a growing demand in the industry for better performing, more efficient, and cost-effective design tools to assist designers during the design process. Moreover, the ever-increasing complexity of mechanical systems today often necessitates a shift away from relying solely on manual expertise and a movement towards integrating computer-aided design tools into the design process. Such computational tools not only assist designers in satisfying all performance metrics and reducing total design hours but also aim to generate designs that are optimal.

One of the key drivers of mechanical system performance is the distribution or layout of components within the system, an ideal problem for solving using packaging optimization. Packaging optimization is a class of computational design tools that can be used for solving such component distribution problems. Typically, packaging optimization problems are tasked with determining the positions and orientations of a set of objects within a restricted volume or domain, such that an objective is maximized or minimized.

There exist three primary challenges inherent to packing type problems: their NP-hard computational complexity, high multi-modality, and initial condition dependence. Problems of NP-hard complexity have solutions that can be checked and verified in polynomial time but may not be solvable in polynomial time [1]. Well-known examples of NP-hard type problems are the traveling salesman problem [2], the knapsack problem [3], or the bin packing problem [4], where an optimal solution can always be found, but the number of inputs significantly increases the complexity and computation time of finding an optimal solution. NP-hard problems are also inherently non-deterministic, making it challenging or even impossible to derive an analytic solution [5]. Due to the high multi-modality of packaging problems, design spaces are commonly unpredictable, noisy, and include many locally optimum solutions. As a result, certain solutions may be unlikely or impossible for some packing algorithms to achieve based on certain input conditions. This typically necessitates several individual optimizations with differing initial conditions to ensure that the design space is thoroughly explored.

Packing problems have been a topic of interest in literature and academia for decades. Due to their widespread applicability, it is not surprising that they have found applications in various fields such as biology [6, 7], material science [8, 9], engineering [10, 11], and manufacturing [12, 13]. Packaging problems are often attempted to be solved using non-gradient, heuristic, and stochastic-based optimization methods used to simplify the problem and relax the design space. Arguably, the simplest heuristic algorithms for packing problems are a set of "fitness"-based algorithms such as first fit [14], next fit [15], best fit [16], and last fit [17] packing algorithms. These algorithms are straightforward to implement and easily generalizable to fit most types of packing-related problems. However, fitness-based algorithms rely, in part, on random selection and thus do not generally lead to optimum solutions. Evolutionary, meta-heuristic, and probabilistic optimization methods such as genetic algorithms and simulated annealing are also popular packaging methods. Given their population-based methodology, the design space can be explored in several directions simultaneously: ideal for algorithm parallelization but tend to be slow and require unique tuning of several algorithm parameters for each unique problem [18, 19]. Wodziak and Fadel successfully used a genetic algorithm-based approach for the packaging of boxes into tractor trailers [20]. Their algorithm was shown to provide viable solutions that not only considered packing density but also considered center of gravity. However, optimization runtimes were shown to take upwards of several hours for only a few dozen geometrically simplistic components. Other examples of packaging optimization using genetic algorithms include the novel method for the placement procedure of components by Gonçalves and Resende [21], the hybrid genetic algorithm approach to three-dimensional bin packaging problems by Feng and Moon [22], and hybrid genetic algorithm and linear programming approach by Liu and Si [23]. Multiple novel approaches to packaging using simulated annealing such as the "neighborhood structure" method by Dowsland and Soubeiga [24], or domain-component and component-component spatial consideration strategies by Cagan and Degentesh [25], are just a few examples among dozens in literature. Gomes and Oliveira also utilized simulated annealing to solve irregular strip packing problems with novel methods for handling non-convex shapes [26]. Their algorithm was limited to two-dimensional geometries, and additional heuristic subroutines related to the initial layout of components were required to combat relatively high runtimes due to the simulated annealing process. These types of initial condition rulesets and deterministic pre-solvers are both common inclusions in packaging algorithms in attempts to reduce initial condition dependence and decrease optimization time. Pre-solvers generate initial conditions by first automatically deriving an initial feasible solution to provide the packaging algorithm, used in such work as Torres and Hitschfeld for modeling of rock and porous media [27]. Pre-solvers should be implemented carefully, however, to avoid rapidly converging on less optimal solutions and to ensure the design space remains thoroughly explorable. Feasible packaging solutions also commonly require that objects are positioned without the existence of geometric overlap between them. When this constraint is applied, implementation of packaging algorithms, specifically ones used to pack three-dimensional, non-convex objects, commonly sacrifices geometric accuracy in favor of computational efficiency to avoid the mathematically complex and computationally expensive geometric overlap measurement computations. Several types of methods used for geometric simplification can be found in literature. Such examples include the work by Fernandez and Bennell where objects were non-convex components were approximated into cube-like voxels [28]. Using this method, optimization runtimes could be reduced at the geometric accuracy cost of coarse voxel refinement and vice versa. In the work by Pankratov and Romanova, complex geometries were approximated from the union of a collection of pre-defined basic geometries (e.g., spheres, cylinders, cones) [29]. While this approach has limited capabilities in accurately representing practical geometries, the proposed algorithm was shown to have relatively rapid runtimes. Demir and Aliaga proposed a novel method for the automatic decomposition of objects into near-convex sub-components for use in the packaging of additive manufacturing components [30]. Such geometric simplifications result in conservative solutions with objectives that could be made significantly more optimal if true geometries were considered.

In recent years, the utilization of artificial intelligence in the form of deep and reinforcement learning algorithms has gained significant traction as an approach for tackling general optimization problems. This trend is bolstered by the increasing accessibility of open-source frameworks. When focusing on packaging optimization problems, several examples in literature exist that explore the efficacy of artificial intelligence as a value tool in addressing the diverse array of packaging optimization challenges. For example, deep reinforcement learning approaches have been shown for solving the rectangular strip packaging problem by Fang and Rao [31], for an autonomous ore packing system by Ren and Zhong [32], and the optimal vehicle packing space optimization with a focus on packing sequence of items by Tian and Kang [33]. Like the reliance on tuning parameters of the previously discussed methods, the literature has shown that a primary drawback of these deep learning is the reliance on training data, which may not be readily available for a given type of packaging problem. Even if training data can be used, overfitting may occur if training is done on small data sets, potentially reducing the scalability and flexibility of the packaging optimizer.

The state-of-the-art packaging algorithms typically involve multi-physics or multi-objective problem statements; however, few examples of the successful implementation of multi-objective packing can be found in literature. This can be attributed to the high complexity and challenge of modeling several physical effects (thermal, vibrational, etc.) in relation to component layout while simultaneously accounting for the aforementioned challenges inherent to packing problems in general. Despite this, researchers are continuing to strive for novel approaches to this difficult type of optimization problem [34–38].

The proposed packaging optimization algorithm in this work uses the framework developed by Carrick and Kim [39]. Carrick and Kim proposed a novel packaging optimization method aimed to produce a non-probabilistic, heuristic, packaging algorithm which could consider such performance measures. In this work, packaging optimization was achieved through the dynamic simulation of component position and orientation over time. The motion of components from their initial positions and orientations to their final, packed positions and orientations are driven by dynamically updated "packaging accelerations" imposed onto the components. Components are accelerated using heuristic rulesets based on optimization objectives and physical effects between components or the environment. Undesirable interactions between components increase the magnitude of these packaging accelerations which then accelerate components towards positions and orientations that produce favorable interactions. Therefore, throughout optimization, components are continuously accelerated towards positions and orientations that minimize their interaction potential and packaging acceleration magnitudes. Local optimum solutions are then found once all components have collectively settled into a low energy state where component interactions are minimized, resulting in all components coming to rest. An example of the behavior of Carrick and Kim's packaging algorithm is shown in Fig. 1.

The current state-of-the-art packaging optimization methodologies and algorithms lack the extensive and simultaneous consideration of several key aspects of practical packaging problems. First, packaging methodologies are commonly developed to solve individual packaging problems using heuristics tuned to best fit unique packaging problems. Ideally, the packaging optimization algorithm could be utilized to solve a wide variety of practical packaging problems with minimal adjustments or tuning of the algorithm parameters. The ability to accurately represent the commonly three-dimensional, non-convex geometries of practical system components is also uncommonly accounted for in the literature. Most algorithms proposed in the literature either enforce convex approximations or construct geometric representations of components using a limited collection of simplified shapes. Furthermore, maximizing packing density is the sole focus of most packaging algorithms in literature, neglecting any other practical engineering design factors or system performance metrics. The position and orientation of components commonly have a direct effect on system performance in areas such as

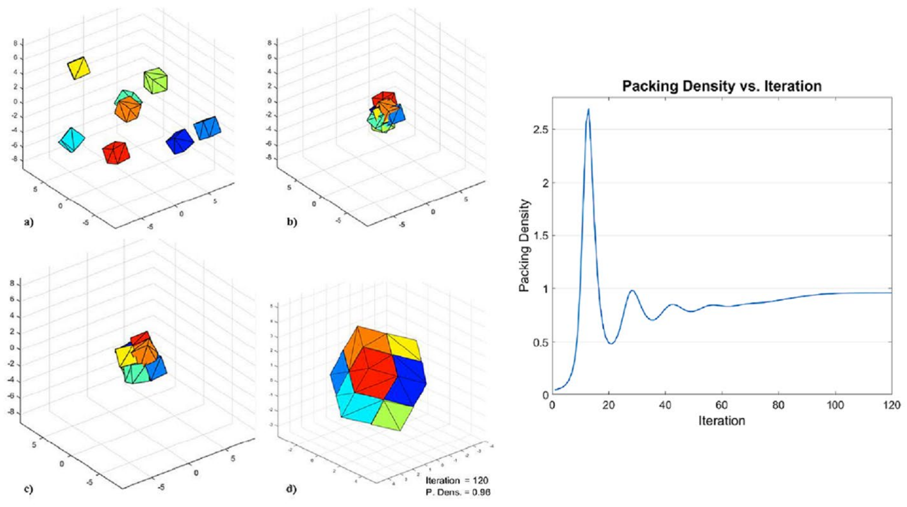


Fig. 1 Left: Dynamic layout of components. Right: Packing density per iteration [39]

stability, center of mass, thermal performance, vibrational considerations, and more. As a result, packing density alone is insufficient for practical packaging solutions, and a multi-objective approach or additional physical constraints are required to properly model the packaging problem and produce meaningful results.

The objective of this work is to address both (i) the time-sensitive and resource-intensive challenges inherent to the cyclical nature of mechanical systems design and (ii) the NP-hard computational complexity, high multi-modality, and initial condition dependence challenges inherent to packaging problems by significantly improving and expanding upon the novel dynamic vector fields methodology framework developed by Carrick and Kim. The continued development of this dynamic acceleration approach aims to contribute to the field of mechanical system design by proposing a new algorithm that is more efficient, improves optimization stability and convergence reliability, and considers practical performance metrics that go beyond packing density. Primary contributions specifically include updated accuracy and efficiency improvements to several of the existing packaging acceleration methodologies, improved methods for the handling of three-dimensional convex and non-convex components, and the simultaneous consideration of center of mass and rotational inertia system performance metrics in addition to packing density. The remainder of this paper begins by providing a theory and methodology overview, followed by several numerical examples to demonstrate the effectiveness of the proposed algorithm. The paper ends with a discussion of the results and recommendations for future improvements that could increase the robustness of the algorithm.

Methods

In packaging optimization problems, efficient distribution of components within a constrained domain is generally measured with packing density, defined as the ratio of the sum of component volumes to the convex hull surrounding all components. The optimization objective used in this algorithm is therefore set as the minimization of negative packing density. The generalized mathematical optimization statement is stated as follows:

$$\textbf{Minimize}\ \rho_p = -\frac{\sum_{i=1}^{n_c} V_i}{V_{CH}}\ \text{(Packing density)}$$

$$\begin{array}{lll}\textbf{w.r.t} & p_i(x, y, z) & \text{(Position)} \\ & q_i(R_x, R_y, R_z, \theta) & \text{(Quaternion orientation)}\end{array}$$

$$\begin{array}{lll}\textbf{Subject to} & s_I \leq I^* & \text{(Rotational inertia)} \\ & \| s_{COM} - s^I_{COM} \| \leq \text{CoM}^* & \text{(Center of mass)} \\ & \| o_{i,j} \| \leq \text{TOL} & \text{(Component geometric overlap)} \\ & \| \frac{d^2 p_i}{dt^2}_{DE} \| = 0 & \text{(Component} - \text{domain acceleration)}\end{array}$$

where ρ_p represents the packing density of the system's components, V_i is the volume of component i, and V_{CH} is the volume of the convex hull encapsulating all components, n_c is the number of components, p_i is the position of the geometric centroid of component i with respect to the coordinates $[x, y, z]$, q_i is the quaternion orientation of component i defined by its axis of rotation $[R_x, R_y, R_z]$ and angle θ, s_I is the system's rotational inertia, s_{COM} is the system's center of mass, s^I_{COM} is the intended center of mass, I^* and COM^* are the upper limits on rotational inertia and center of mass deviation constraints, $o_{i,j}$ is the geometric overlap between component i and component j with geometric overlap tolerance TOL, and $\left(d^2 p_i/dt^2\right)_{DE}$ is the domain-containment acceleration of component i, representing its containment within the domain. The workflow for the presented packaging optimization algorithm in this work can be summarized and grouped into three distinct phases: initialization, main loop, and outputs. The initialization phase retrieves all input data (e.g., geometry files, initial component poses, system elements) required to define the first optimization iteration. Next, the main loop phase executes the packaging optimization algorithm itself. Finally, the output phase extracts and reports the results of the converged optimization. The flowchart in Fig. 2 illustrates this methodology.

The presented packaging optimization algorithm packages components from an initial position to a final packed position inside of a constrained volume, denoted a domain. The dynamic motion of components from their initial to final positions is driven by the summation of several packaging acceleration forces imposed on each component. These individual accelerations are each derived from physical interactions between components, the domain, and any system-level design requirements and constraints. Interactions between components that produce unfavorable performance metrics because of their relative positions and orientations (poses) trigger the acceleration of components towards more favorable poses. This behavior iteratively lowers the interaction potential of the system of components and ultimately results in components coming to rest over time. The optimally packed configuration is then established once all components come to rest.

Component geometries are imported into the algorithm in stereolithography (STL) format which automatically discretizes each component into triangular faces. The position of each component is tracked throughout optimization using its geometric centroid, and each vertex of every triangular face is tracked relative to this geometric

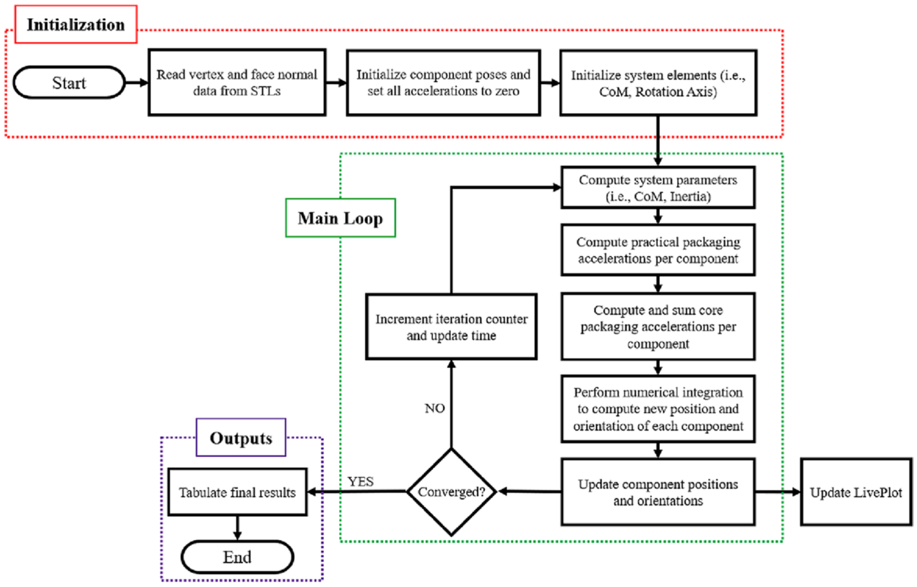

Initialization
Start
Read vertex and face normal data from STLs
Initialize component poses and set all accelerations to zero
Initialize system elements (i.e., CoM, Rotation Axis)
Main Loop
Compute system parameters (i.e., CoM, Inertia)
Compute practical packaging accelerations per component
Compute and sum core packaging accelerations per component
Perform numerical integration to compute new position and orientation of each component
Update component positions and orientations
Update LivePlot
Converged?
NO
Increment iteration counter and update time
YES
Outputs
Tabulate final results
End

Fig. 2 Packaging optimization algorithm flowchart

centroid. As several of the packaging acceleration methodologies rely on convex geometries, non-convex/concave components must be discretized into sets of convex sub-components. Each sub-component's position is then tracked relative to an arbitrary "base" sub-component.

However, components found within practical, real-world systems are more likely to have complex, non-convex/concave geometries that can be mathematically challenging to handle. Furthermore, several geometry-related computations and sub-algorithms within the presented packaging algorithm are only valid for convex geometries. As a result, the presented algorithm handles non-convex components by discretizing them into several convex sub-components. All sub-components are then subjected to all packaging accelerations individually. Translational connectivity throughout the dynamic motion of a set of sub-components is then achieved by averaging the total acceleration of all sub-components within the same group and applying the result to each sub-component equally before positional updates.

To produce physically meaningful solutions, no two components may have overlapping geometries when in their final packed positions. The geometric overlap resolution (GOR) acceleration is therefore tasked with repelling overlapping components and is derived using a two-step process: overlap detection and overlap resolution. The first step of overlap detection is implemented using the Gilbert-Johnson-Keerthi (GJK) algorithm [40]. The GJK algorithm not only allows for the detection of overlap between two convex shapes but also provides the minimum distance between them if they are not overlapping. Although the GOR acceleration only utilizes GJK's overlap detection output, the minimum distance output is also stored for use later in the algorithm. If GJK detects two components are overlapping by any amount, the expanding polytope algorithm (EPA) is then utilized to compute the maximum amount of overlap [41]. The penetration vector output from EPA, representing the maximum overlap, is then used to derive the GOR acceleration for each overlapping component as follows:

$$\overrightarrow{\frac{d^2p_i}{dt^2}}_{GOR} = S_{GOR}\sum_{j=1}^{n_c}\vec{o}_{i,j} \tag{1}$$

where $\left(d^2p_i/dt^2\right)_{GOR}$ is the GOR acceleration of component i, S_{GOR} is the GOR global acceleration scaler, $o_{i,j}$ is the penetration vector of component i with respect to component j, and n_c is the number of components. By default, GOR acceleration is applied when the magnitude of $o_{i,j}$ is greater than zero (i.e., constraining any amount of geometric overlap). To improve convergence stability and reduce runtime, the user may optionally decide to relax this constraint by permitting some allowable geometric overlap tolerance. If a nonzero tolerance is applied, GOR acceleration is instead applied when the magnitude of $o_{i,j}$ is greater than the tolerance value. A two-dimensional illustration of the behavior of the GOR acceleration is shown in Fig. 3.

To achieve densely packed configurations, components are encouraged to group together through attraction-based accelerations. The component attraction (CA) acceleration accelerates all components towards each other's geometric centroids as a function of component proximity. The farther apart two components are from each other, the stronger they will be attracted to one another by the CA acceleration. Component proximity, set as the minimum distance between the surfaces of two components, is pre-computed in the GJK algorithm utilized in the GOR acceleration computations. This minimum distance is then set as the magnitude of the CA acceleration such that when two components come into contact, no further CA acceleration between them is applied. However, the magnitudes of all CA accelerations are first penalized based on

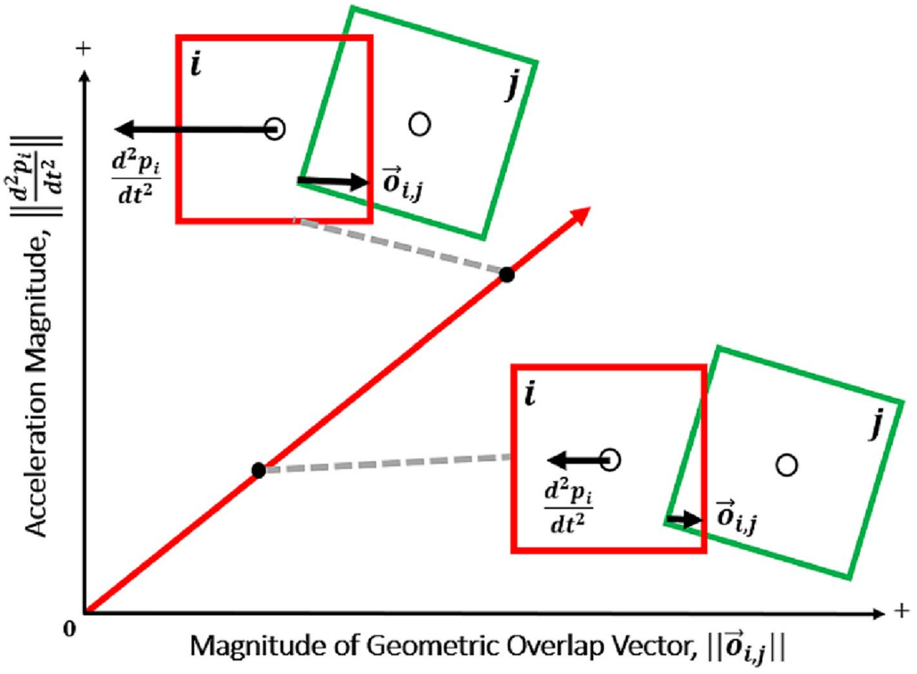

+
i
j
$\frac{d^2p_i}{dt^2}$
$\vec{o}_{i,j}$
Acceleration Magnitude, $\left\|\frac{d^2p_i}{dt^2}\right\|$
i
j
$\frac{d^2p_i}{dt^2}$
$\vec{o}_{i,j}$
0
+
Magnitude of Geometric Overlap Vector, $\|\vec{o}_{i,j}\|$

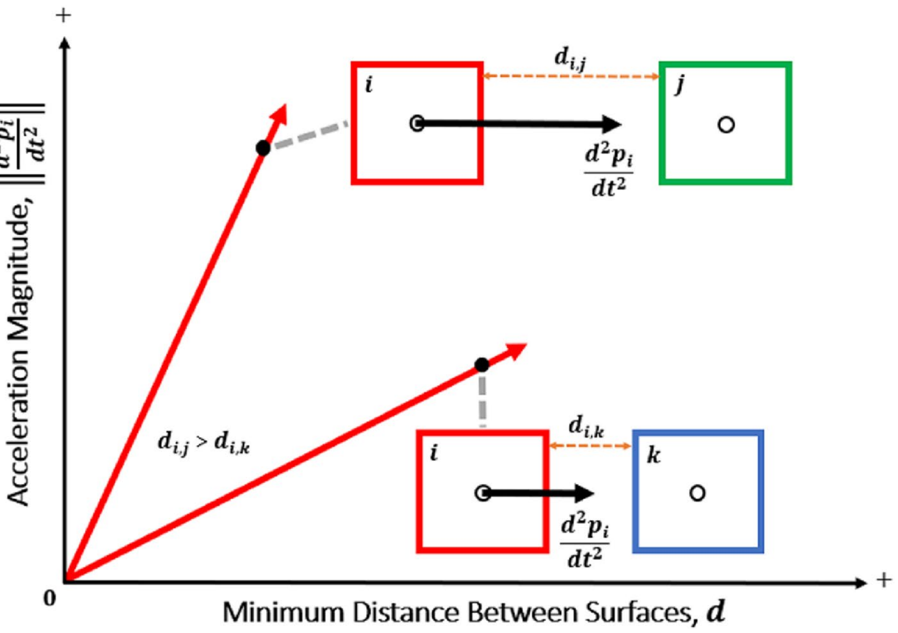
Fig. 3 Illustration of the acceleration magnitude for geometric overlap resolution acceleration

relative component distances. This penalization process minimizes attraction of components located very far from each other and encourages attraction of components located close to each other, effectively only permitting local component attraction. The CA acceleration is then mathematically formulated as follows:

$$\overrightarrow{\frac{d^2p_i}{dt^2}}_{CA} = S_{CA}\sum_{j=1}^{n_c} P_{i,j}\overrightarrow{(p_j - p_i)} \quad \text{where } P_{i,j} = \left(\frac{d_{max}-d_{i,j}}{d_{max}-d_{min}}\right)^{\gamma} \tag{2}$$

where $\left(d^2p_i/dt^2\right)_{CA}$ is the CA acceleration of component i, S_{CA} is the global CA acceleration scalar, $P_{i,j}$ is the distance penalization factor for component i with respect to component j, and p_i and p_j are the positions of the geometric centroids of component i and component j, respectively, d_{max} and d_{min} are the maximum and minimum distances with respect to all components, $d_{i,j}$ is the minimum distance between component i and component j, and γ is the penalization exponent. A two-dimensional illustration of the behavior of the CA acceleration is shown in Fig. 4.

Solving practical packaging problems generally necessitates the containment of all components into a domain with a fixed volume and orientation. The domain encapsulation (DE) acceleration is therefore tasked with accelerating components into the domain using a modified methodology first developed by Carrick and Kim. If a component exists entirely or partially outside of the domain, identified using the GJK algorithm, the vector from the vertex of the component furthest outside of the domain to the closest domain surface is stored. This vector is then used to set each component's DE acceleration magnitude and direction. However, if a component exists entirely within the domain, identified by checking if each vertex of the component is located within the domain, no DE acceleration is applied. The DE acceleration is therefore formulated as follows:



+
Acceleration Magnitude, $\left\|\frac{d^2p_i}{dt^2}\right\|$
$d_{i,j}$
i
j
$\frac{d^2p_i}{dt^2}$
$d_{i,j} > d_{i,k}$
$d_{i,k}$
i
k
$\frac{d^2p_i}{dt^2}$
0
+
Minimum Distance Between Surfaces, d

Fig. 4 Illustration of the acceleration magnitude for component attraction acceleration

$$\overrightarrow{\frac{d^2p_i}{dt^2}}_{DE} = S_{DE}\overrightarrow{x_{i,d}} \tag{3}$$

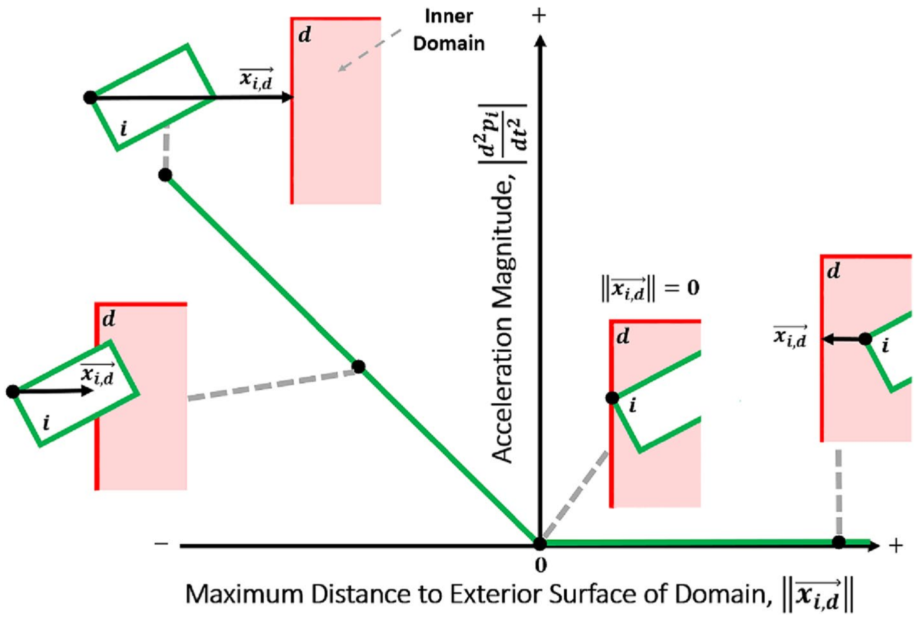
where $\left(d^2p_i/dt^2\right)_{DE}$ is the DE acceleration of component i, S_{DE} is the DE global acceleration scaler, and $x_{i,d}$ is the minimum distance of component i to the exterior surface domain d. A two-dimensional illustration of the behavior of the DE acceleration is shown in Fig. 5.

Center of mass alignment is an optional system-level performance metric included in this algorithm. The center of mass alignment (COMA) acceleration is tasked with accelerating components towards positions that minimize the deviation between the intended center of mass (manually input by the user) and the current center of mass of components throughout optimization. The current center of mass at any given iteration is first computed using a point mass assumption:

$$s_{COM} = \sum_{i=1}^{n_c} \frac{m_i p_i}{m_T} \tag{4}$$

where s_{COM} represents the current center of mass of the system, n_c represents the number of components, p_i is the position of the geometric centroid of component i, m_i is the mass of component i, and m_T is the total mass of all components. For each component, its position that would individually align the intended center of mass of the system (assuming all other components were fixed) is determined using the following equation:

$$p_i{}^{COM} = \frac{m_T\left(s^I_{COM} - s_{COM-i}\right)}{m_i} \tag{5}$$



Inner Domain
d
$\overrightarrow{x_{i,d}}$
i
+
Acceleration Magnitude, $\left|\frac{d^2p_i}{dt^2}\right|$
$\|\overrightarrow{x_{i,d}}\| = 0$
−
0
+
Maximum Distance to Exterior Surface of Domain, $\|\overrightarrow{x_{i,d}}\|$

Fig. 5 Illustration of the acceleration magnitude for domain encapsulation acceleration

where p_i^{com} represents the position of component i that would align the system's current center of mass to the intended center of mass s_{COM}^I, s_{COM-i} is the system's center of mass when component i is excluded, and m_i and m_T represent the mass of component i and the total mass of all components respectively. The COMA acceleration is then mathematically formulated as follows:

$$\overrightarrow{\frac{d^2p_i}{dt^2}}_{COMA} = S_{COMA}\left(\overrightarrow{s_{COM}^I} - \overrightarrow{s_{COM}}\right)\frac{\left(\overrightarrow{p_i^{COM}} + \left(\overrightarrow{s_{COM}^I} - \overrightarrow{s_{COM}}\right) + \left(\overrightarrow{s_{COM}^I} - \overrightarrow{p_i}\right)\right)}{\|\left(\overrightarrow{p_i^{COM}} + \left(\overrightarrow{s_{COM}^I} - \overrightarrow{s_{COM}}\right) + \left(\overrightarrow{s_{COM}^I} - \overrightarrow{p_i}\right)\right)\|} \tag{6}$$

where $\left(d^2p_i/dt^2\right)_{COMA}$ is the COMA acceleration of component i, S_{COMA} is the COMA global acceleration scaler, s_{COM}^I is the intended center of mass, s_{COM} is the current center of mass of the system, p_i^{com} is the position of component i that would align the system's current center of mass to the intended center of mass, and p_i is the position of the geometric centroid of component i. The COMA acceleration methodology is illustrated in Fig. 6.

Rotational inertia is another optional system-level performance metric included in this packaging optimization algorithm. The rotational inertia reduction (RIR) acceleration is tasked with minimizing the rotational inertia about a rotation axis (manually input by the user) by accelerating components towards the rotation axis. The rotational inertia about the rotation axis is approximated using a point mass assumption:

$$s_I = \sum_{i=1}^{n_c} m_i \overrightarrow{r_i}^2 \tag{7}$$

where s_I is the rotational inertia of the system, n_c is the number of components in the system, m_i is the mass of component i, and r_i is the perpendicular vector from

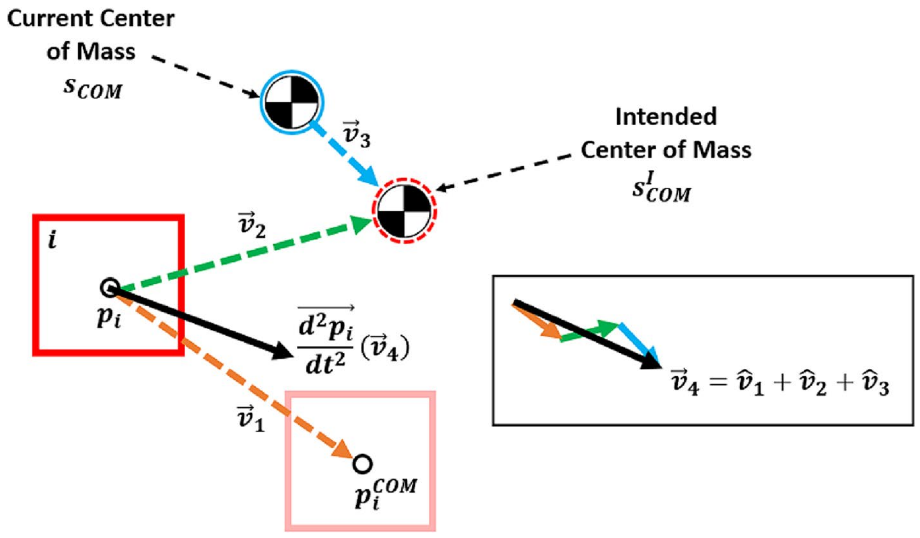

Current Center of Mass s_{COM}
$\vec{v}_3$
Intended Center of Mass s_{COM}^I
i
$\vec{v}_2$
p_i
$\overrightarrow{\frac{d^2p_i}{dt^2}}\ (\vec{v}_4)$
$\vec{v}_1$
p_i^{COM}
$\vec{v}_4 = \hat{v}_1 + \hat{v}_2 + \hat{v}_3$

Fig. 6 Vector summation method used to dictate the direction of the center of mass alignment acceleration

component i to the axis of rotation. The RIR acceleration is then mathematically formulated as follows:

$$\overrightarrow{\frac{d^2 p_i}{dt^2}}_{RIR} = S_{RIR}\overrightarrow{r_i} \tag{8}$$

where $\left(d^2p_i/dt^2\right)_{RIR}$ is the RIR acceleration of component i, S_{RIR} is the RIR global acceleration scaler, and r_i is the perpendicular vector from component i to the axis of rotation. The RIR acceleration methodology is illustrated in Fig. 7.

Components are also dynamically rotated throughout optimization to align their faces, a process which generally increases packing density. Rotational alignment in this algorithm is performed using the methodology first developed by Carrick and Kim. All components are rotationally aligned to all other components. For each component pair, two faces (one per component) are identified and used for rotational alignment. These two faces are selected as the faces which identify the minimum distance between the two aligning components. The angle between the two normal vectors defined by these selected faces is then used to set the angle of rotation, and the cross product of these two normal vectors sets the axis of rotation positioned at the component's geometric centroid. The rotational alignment (RA) acceleration then iteratively rotates components such that the angle between the selected faces is minimized. For each component, this process is repeated with respect to all other components. The resulting set of rotations is applied using quaternion rotation to avoid gimbal locking. For discretized components, rotational connectivity is enforced using the same rotational alignment methodology;

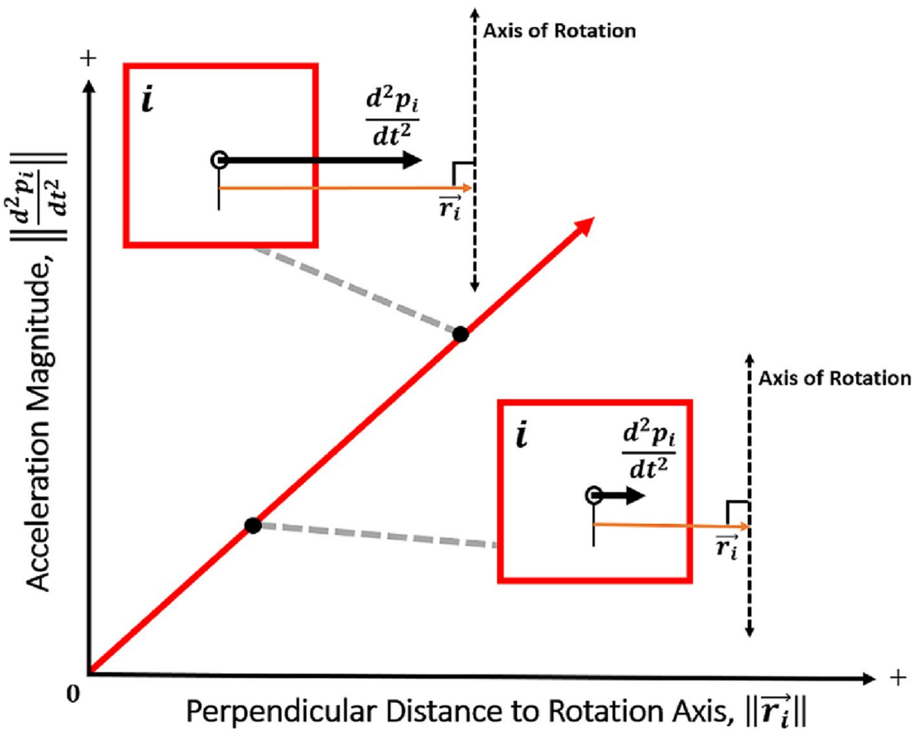

Axis of Rotation
+
i
$\frac{d^2p_i}{dt^2}$
$\vec{r_i}$
Acceleration Magnitude, $\left\|\frac{d^2p_i}{dt^2}\right\|$
Axis of Rotation
i
$\frac{d^2p_i}{dt^2}$
$\vec{r_i}$
0
+
Perpendicular Distance to Rotation Axis, $\|\vec{r_i}\|$

Fig. 7 Illustration of the acceleration magnitude for rotational inertia acceleration

however, only one normal vector is selected for the entire set of sub-components. Sub-components are then rotated equally about the average position of all sub-component's geometric centroids. An illustration of this rotational alignment process is shown in Fig. 8.

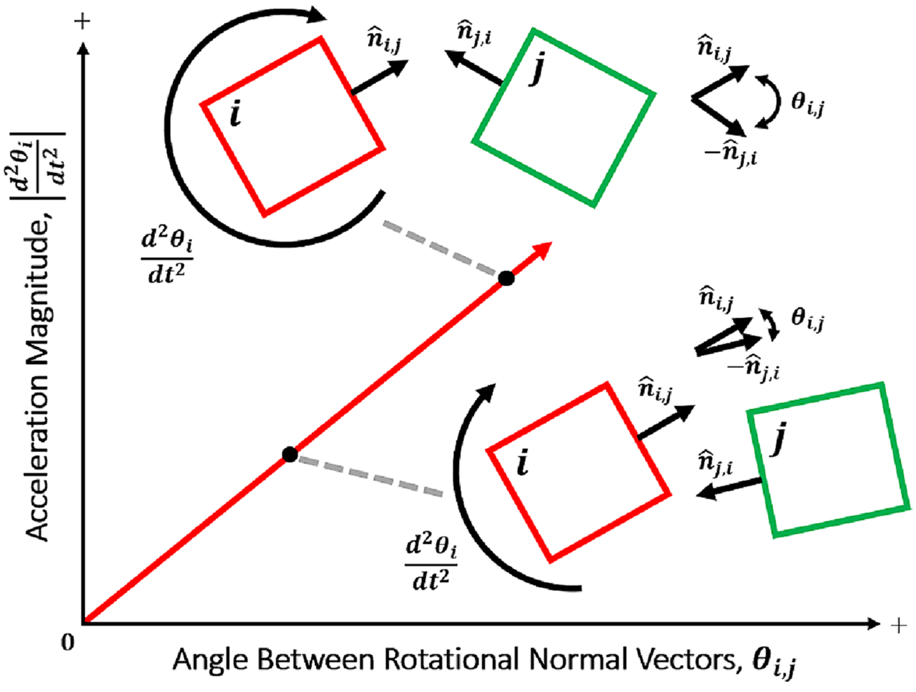
After the total acceleration for each component is determined by the summation of all individual packaging accelerations, numerical integration is used to update the velocity and position of each component using the Euler method. Damping of the translational velocities is also included to avoid oscillations of components nearing their final positions. The formulation for this integration scheme is as follows:

$$\frac{dx}{dt}_{t_o+h} = \zeta\left(\frac{dx}{dt}_{t_0} + h\frac{d^2x}{dt^2}_{t_0}\right) + O\left(h^2\right) \tag{9}$$

$$x_{t_0+h} = x_{t_0} + \mathrm{h}\frac{dx}{dt}_{t_0} + O\left(h^2\right) \tag{10}$$

where x is the positional term, (dx/dt) is the velocity term, (d^2x/dt^2) is the acceleration term, t_0 is the start time, h is the timestep, ζ is the damping factor, and $O\left(h^2\right)$ is the truncation error of order 2.

Two methods of optimization convergence are implemented: maximum iteration count and minimal positional deviation. If all components come to rest, the interaction potential between components has been minimized, and thus, no further iterations will significantly change the configuration. Identifying whether all components have come to rest is accomplished by tracking the magnitude of the acceleration of each component



Acceleration Magnitude, $\left|\frac{d^2\theta_i}{dt^2}\right|$
Angle Between Rotational Normal Vectors, $\theta_{i,j}$
0
+
+
$\frac{d^2\theta_i}{dt^2}$
i
j
$\hat{n}_{i,j}$
$\hat{n}_{j,i}$
$-\hat{n}_{j,i}$
$\theta_{i,j}$

Fig. 8 Illustration of the component rotational alignment methodology

and ensuring each component's acceleration is below a specified tolerance value for several sequential iterations. A secondary rotational check is also conducted to ensure that all components are no longer rotationally aligning for several sequential iterations.

Results and discussion

Several packaging problems with well-defined and known optimal solutions were used to verify the efficacy of the current implementation of the algorithm. The presented results were collected using a computer with 12 cores running at 3.79 GHz and 128 GB of RAM, and the algorithm is currently written in the Python programming language.

Empirical proof has shown that for spheres of equal size, the maximum possible packing density based on hexagonal close packing or face-centered cubic lattice structures is $\pi/(3\sqrt{2})$ or approximately 74% [42]. For spherical packing with arbitrarily and irregularly packed spheres, literature has shown a maximum packing density of 64% is achievable [43]. Therefore, a study was conducted to determine if this packaging algorithm can produce a packing density of 64% using a set of 12 spheres of equal size. The spheres used in this study were approximated using an STL of 120 faces and were provided random initial positions. No domain was included in this study. Out of 100 individual optimizations, 91 achieved the optimal packing density of 64% for irregularly packed spheres with an average computation time of approximately 4 min.

The packing of eight equally sized cubes has an empirically optimal packing density of 100%, which can be achieved with several possible configurations. A domain was included in this study to force the result into only permitting the formation of one of the optimal configurations. The domain-restricted optimal configuration selected for this study was the formation of a larger cube with side lengths equal to two times the side length of the cubes it is composed of. The domain was placed at the origin, with all cube components placed randomly outside of the domain and with random initial orientations. Out of 100 individual optimizations, 83 achieved a maximum packing density of 95% with an average computation time of approximately 3 min. The densest configuration achieved was 97%. Although the optimal packing density of 100% was not achieved, the final configuration of the cubes was in the expected optimal arrangement. Upon closer inspection, minor face misalignments and gaps between faces of components could not resolve before accelerations were small enough to trigger convergence.

A third study was performed to test the efficacy of the packaging algorithm using non-convex components. Orthogonally connected cubes known as tetracubes, a class of polycubes, were selected for this study because of their clear non-convexity and ease of discretization into convex sub-components. Two of these tetracubes are capable of fitting together such that they form a cube-shaped configuration, resulting in an expected optimal packing density of 100%. The tetracubes were provided random initial positions and orientations. No domain was included in this study. Out of 100 individual optimizations, 72 achieved a packing density over 95% in the cube-shaped configuration. The densest configuration achieved was 98%. Notably, the remaining 18 solutions did not form the optimal cube-shaped configuration because of poor initial orientations that did not facilitate enough rotation time to permit the interlocking of components before coming into contact. This is an example of the current initial condition dependance limitations of the algorithm. An example of the dynamic layout of the components for a single optimization for each verification study is shown in Fig. 9.

Several packaging algorithms described in literature execute packaging optimization through the approach of adding new geometry to the problem sequentially, as opposed to including all geometry at the initial iteration [12, 22, 33]. To assess this packing optimization strategy, an additional study was conducted using the proposed algorithm in this work. This study involved the packing of eight equally sized cubes into a rectangular prism-shaped domain, where an optimal solution would force a flat-packed 2×4 configuration with a packing density of 100%. First, optimization was run to convergence using two of the eight cubes. In each subsequent optimization, the positional results of the previous optimization were utilized as initial conditions, and a new cube was introduced to the problem with a randomized initial position. This sequential process was repeated until all eight cubes were integrated and resulted in a maximum packing density of 99% with the expected 2×4 layout. Total optimization runtime took approximately 11 min or approximately 1.5 min per optimization. The layout of these cubes throughout sequential optimizations is shown in Fig. 10.

This study was then repeated but instead employing the default single optimization approach where all eight cubes were added at the initial iteration. This resulted in a

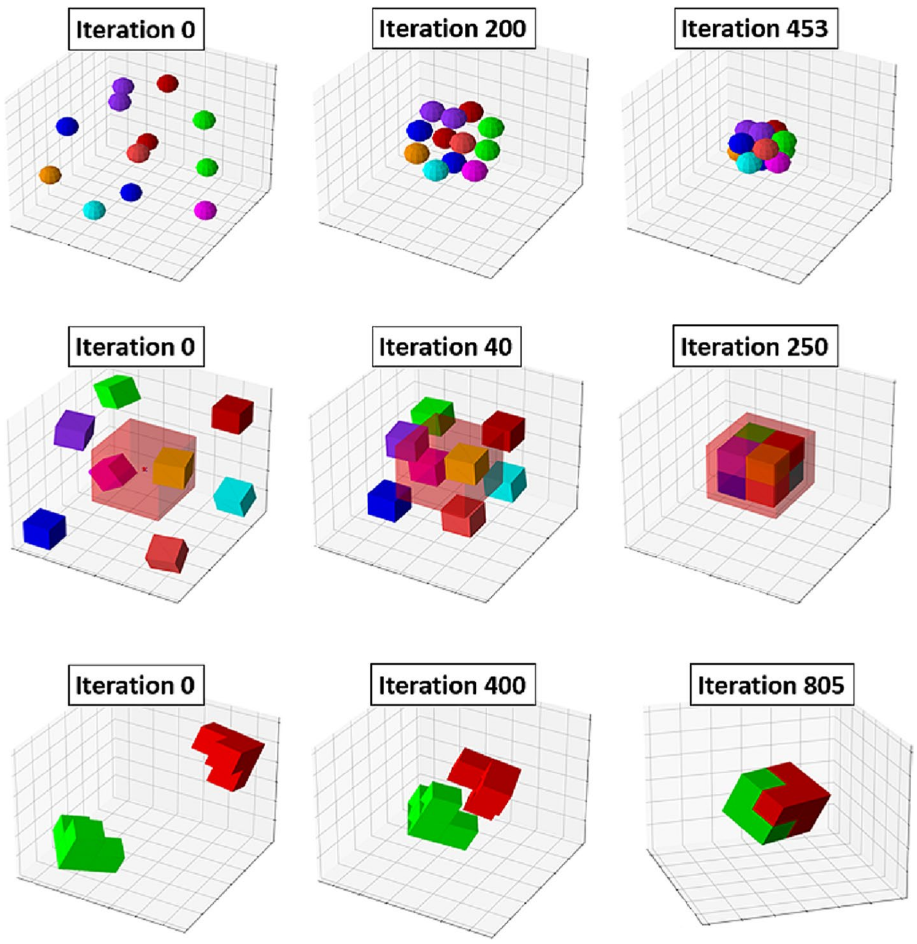


Fig. 9 Dynamic layout of sphere (top row), cube (middle row), and tetracube (bottom row) components

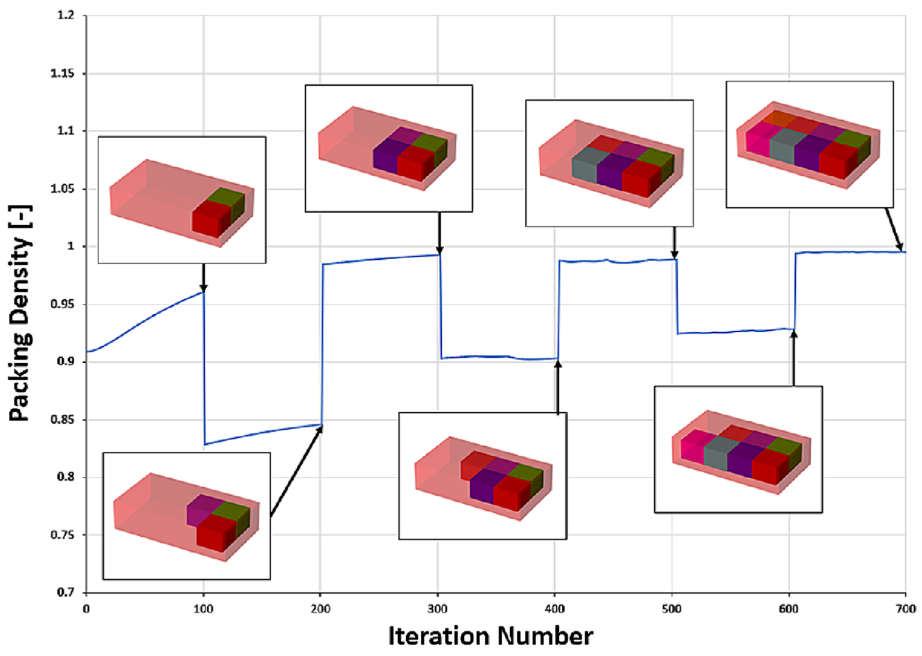


Fig. 10 Converged results of sequential optimization approach

Table 1 Results of packaging verification studies

Study	Avg. packing density [%]	Densest packing [%]	Optimal packing density [%]	Avg. computation time [s]
12 spheres	63.03 (100 trials)	63.98	64.00	239.15
8 cubes (2 × 2 × 2)	96.21 (100 trials)	97.44	100.00	183.32
2 tetracubes	91.49 (100 trials)	98.19	100.00	230.67
8 cubes (2 × 4)	96.23 (1 trial)	96.23	100.00	244.47
8 cubes (2 × 4) seq	99.09 (1 trial)	99.09	100.00	681.14

maximum packing density of 96% due to minor gaps and misalignments between component faces. Convergence was achieved in approximately 4 min. Consequently, while the sequential approach yielded a more densely packed configuration, it also incurred a longer total optimization runtime. This implies a potential trade-off between packing density and optimization runtime between the two optimization approaches. Depending on the scale of the optimization problem, it may be advantageous to run one method over the other if runtime becomes a limiting factor. It should be noted that the algorithm presented in this work is not primarily intended for use with the sequential optimization approach; currently, setting up a new optimization manually is necessary for each subsequent optimization. If future studies reveal further compelling advantages to the sequential approach in terms of performance metrics, it is recommended for future work to automate and refine this sequential approach as a permanent secondary optimization method within the algorithm.

The results of the previous verification studies have been summarized in Table 1 to exhibit the efficacy of the current implementation of the proposed algorithm.

A final study was conducted with the objective of exhibiting the algorithm's prospective capabilities in addressing practical, real-world packaging problems. A fictitious, simplified, hybrid urban air mobility (UAM) nacelle was used for this study. A set of components commonly found within hybrid UAM nacelles were included as follows: a data acquisition unit (DAU), cooling fan, water tank, inverter, pinion gear with a linear actuator for nacelle rotation about a rotation axel, heat exchanger, and water pump. The motor was excluded for this study, as it was assumed the motor must exist in a fixed position at the nose of the nacelle and would exist partly outside of the nacelle domain. As this was a fictitious nacelle packaging problem, the dimensions and densities of all components were arbitrarily set. Each component's geometry was represented by a simplified bounding volume approximating the volume that the real component may take up within the domain. All components were also approximated to have uniform densities for this study. Auxiliary components, such as electrical wiring, air ducts, or bracketing, were not considered in this study. The pinion gear and linear actuator for the tilting mechanism were treated as attached components, as their proximity to one another would be critical for tilting functionality. To facilitate tilting about the nacelle's rotation axel, the pinion gear was also translationally and rotationally fixed to the intended center of mass positioned along the center of the rotation axis. The fan-cake type heat exchanger was constructed as a non-convex component, discretized into several convex sub-components. It was also assumed that the heat exchanger must be axially aligned to the cooling fan through their midpoints to promote efficient airflow from the cooling fan through the heat exchanger. The packaging algorithm was run 100 times for this study. For each run, the initial positions and orientations of each component were randomized. Out of the 100 runs, 79 converged before the maximum iteration limit of 1000 was reached, 11 reached this maximum iteration limit, and 10 did not produce a feasible solution. A selection of three different, feasible results from the set of optimization runs has been presented below. These three configurations were specifically selected because they resulted in common local minimum solutions shared between several optimization runs. The final configuration of all three selected designs can be found in Fig. 11, and the system-level performance metrics for each design are summarized in Tables 2 and 3, respectively.

The most important user-defined parameters in this proposed algorithm are the relative acceleration scaling values for the CA and GOR accelerations, S_{CA} and S_{GOR}, respectively. If too large of a value of S_{CA} is used, the components may be too strongly attracted to one another. This can result in significant geometric overlaps that may never be resolved, thus disallowing the optimization convergence criteria to be met. Similarly, too of large of a S_{GOR} value will cause components to rapidly "bounce" off one another at the point of contact, introducing chaotic motion that may never settle to convergence. Conversely, too small values of these two scaling factors can result in unnecessarily long runtimes. A balance between the relative magnitudes of S_{CA} and S_{GOR} should therefore be achieved in order to improve convergence reliability and decrease optimization runtimes. To demonstrate this effect, a parameter study was conducted using the packing of eight equally sized cubes problem. The values of S_{CA} and S_{GOR} were varied from 0.1 to 2.0. For each pair of scaling values, a total of 5 separate optimizations were run with random initial position and orientations of the cubes and with a maximum of 500 iterations.

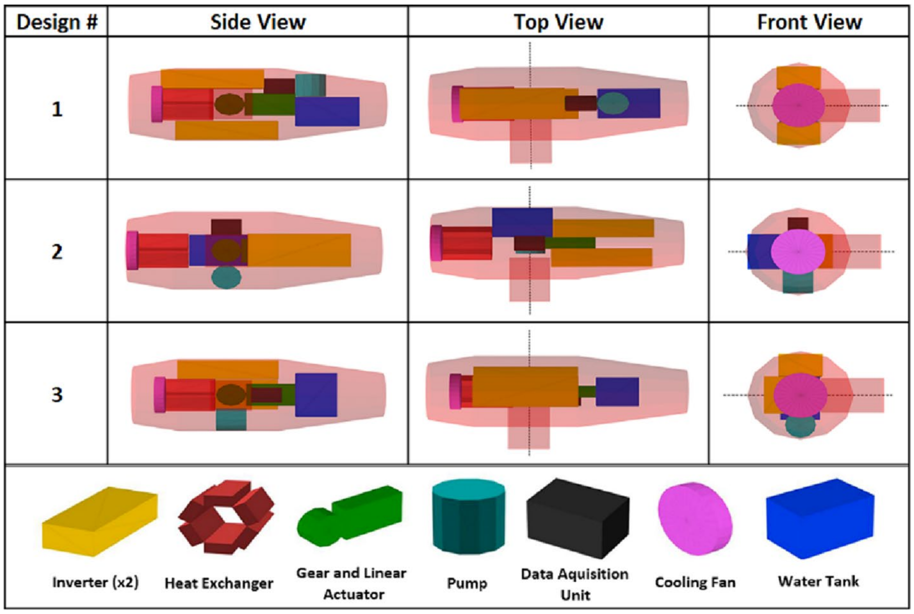


Fig. 11 Configuration of nacelle components for three selected packaging optimization results

Table 2 Solver performance metrics for each selected design

Solver performance metrics		
Design no	**Number of iterations**	**Computation time [s]**
1	313	614
2	451	906
3	404	810

Table 3 System-level performance metrics for each selected design

System-level performance metrics		
Design no	**Performance metric**	**Output**
1	Packing density [-]	0.60
	Center of mass offset [mm]	117.72
	Inertia [kg·m^2]	11.46
2	Packing density [-]	0.45
	Center of mass offset [mm]	67.47
	Inertia [kg·m^2]	8.16
3	Packing density [-]	0.58
	Center of mass offset [mm]	70.50
	Inertia [kg·m^2]	7.55

The average number of iterations for each pair of scaling values is shown in Figs. 12 and 13. Based on the results of this study, it is recommended to use magnitudes between 1.0 and 1.5 for both scaling factors, with S_{GOR} approximately 1.5 times larger than S_{CA}.

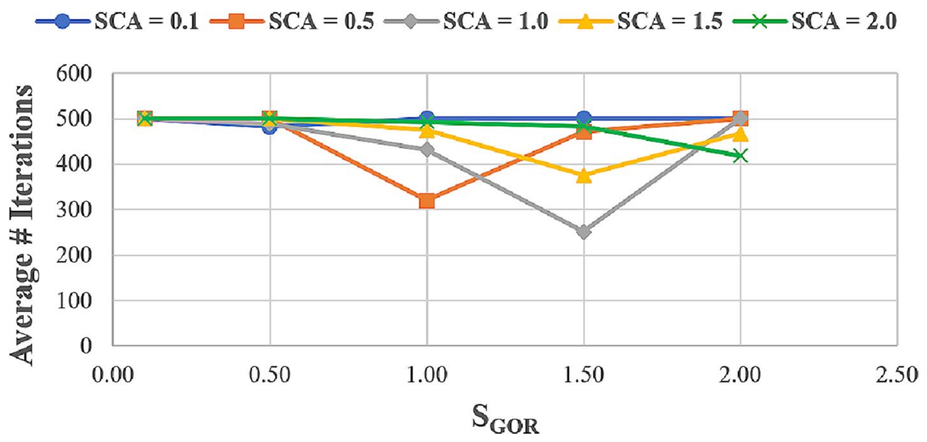


Fig. 12 Average number of iterations for packing of eight equally sized cubes, S_{GOR} and $= S_{CA}$[0.1, 0.5, 1.0, 1.5, 2.0]

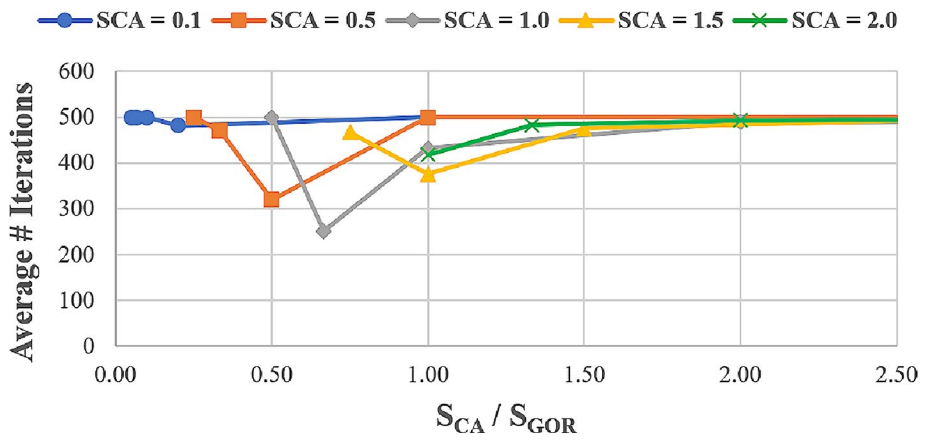


Fig. 13 Average number of iterations for packing of eight equally sized cubes with varying ratios of S_{CA}/S_{GOR}

Further testing is required to properly tune the remaining acceleration scale values, but it is recommended to establish baseline values for S_{CA} and S_{GOR} beforehand.

Currently, the proposed algorithm only accounts for a limited set of practical performance metrics (i.e., center of mass and rotational inertia). However, the modular structure of this algorithm facilitates future development efforts for the inclusion of additional system performance metrics. If certain physical metrics (e.g., thermal, structural, vibrational) can be abstracted into acceleration field representations, this algorithm lays the groundwork for implementing such metrics and even multi-objective optimization. Newly introduced acceleration fields can be seamlessly integrated into the existing set of acceleration fields, where these new accelerations could simply be added to the sum of all activated acceleration fields using the same methodology as described in this work. Future development efforts focused on this approach would provide users with a toolbox of performance metrics akin to those found in traditional computer-aided design software, enhancing the flexibility and customization options for solving a more diverse set of packaging scenarios. Integration with commercial analysis software could also be used to provide analytical data throughout optimization. For example, component poses could be input into an external thermal finite element analysis or computational fluid dynamic solver to compute component body temperature data via conduction and/or convection and output this temperature data to the packaging optimization algorithm at each iteration. New, thermally related acceleration fields as functions of component body temperature, for example, could then be added to the existing set of accelerations being simultaneously applied on all components. This would provide the ability to consider thermally related objective functions such as minimized system temperature or addition of thermally related constraints such as minimum or maximum component temperatures. It should be noted that tuning parameter studies, like the parameter studies conducted in this work, for these new acceleration fields would be required to ensure that new acceleration fields do not over (or under) power existing acceleration fields. Additional studies are also proposed for future work with respect to the performance of the algorithm as a function of the number of unique acceleration fields activated. This is because it is conceivable that incorporating an excessive number of simultaneous acceleration fields could deteriorate optimization performance. In this context, given the acceleration summation-based methodology proposed in this work, as more acceleration fields are applied, the likelihood of some fields nullifying the effects of others increases. Consequently, this phenomenon could lead to a significant rise in optimization convergence time or conversely premature convergence to nonoptimal solutions. Therefore, an investigation on the impact of the number of activated acceleration fields on algorithm performance is crucial for ensuring its effectiveness and scalability in real-world applications.

Conclusions

This paper proposes a novel methodology for solving packaging optimization problems using a dynamic, acceleration-based approach. The algorithm was shown to be effective in addressing three-dimensional packaging scenarios, accommodating components of varying geometrical complexity (i.e., convex and non-convex), preventing geometric overlap between components, and proper encapsulation within design domains. The algorithm was also shown to be capable of yielding optimal results for a variety of packaging problems with known optimal solutions. Furthermore, the algorithms' prospective capability in solving real-world packaging problems was demonstrated through the packaging of a hypothetical nacelle, incorporating practical considerations of center of mass and rotational inertia.

The algorithm proposed in this paper currently considers a limited set of performance metrics. However, the modular structure and acceleration summation-based approach of the proposed packaging algorithm pave the way for future development efforts regarding the inclusion of additional physical effects (e.g., thermal, structural, vibrational). Newly introduced acceleration fields would simply be added to summation of all other existing acceleration fields simultaneously applied on system components. Additionally, integration with commercial analysis software (e.g., computational fluid dynamic solvers, thermal finite element analysis solvers) to provide analytical data throughout optimization for use in the formulation of new acceleration fields is also proposed for future work.

In summary, the novel algorithm in this paper demonstrates an effective methodology for solving generalized packaging optimization problems. Furthermore, this work provides a modular framework that can be expanded upon in the future to incorporate additional physical performance metrics, thereby enhancing the algorithm's capability to solve a broader range of practical packaging problems.

Abbreviations

STL	Stereolithography
GOR	Geometric overlap resolution
GJK	Gilbert-Johnson-Keerthi
EPA	Expanding polytope algorithm
CA	Component attraction
DE	Domain encapsulation
COMA	Center of mass alignment
RIR	Rotational inertia reduction
RA	Rotational alignment
UAM	Urban air mobility
DAU	Data acquisition unit

Acknowledgements
Not applicable

Authors' contributions
CD developed, programmed, and experimentally tested the packaging optimization methodology presented in the manuscript and was the primary contributor in writing the manuscript. JH, SJ, and IYK provided supervisory roles throughout the development of this research and reviewed and provided feedback to the work. All authors read and approved the final manuscript.

Funding
Funding for this research was provided by Queen's University.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 1 July 2023 Accepted: 8 April 2024
Published online: 16 April 2024

References

1. Papadimitriou CH (1994) Computational complexity. Addison-Wesley, Boston
2. Lawler EL (1985) The traveling salesman problem: a guided tour of combinatorial optimization. Wiley, Chichester
3. Martello S, Toth P (1990) Knapsack problems; algorithms and computer implementations. Wiley, Chichester
4. Garey MR, Johnson DS (1996) Approximation algorithms for bin-packing; a survey. Approximation Algorithms for NP-Hard Problems 266:147–172. https://doi.org/10.1007/978-3-7091-2748-3_8
5. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness
6. Lee C, Subbiah S (1991) Prediction of protein side-chain conformation by packing optimization. J Mol Biol 217(2):373–388. https://doi.org/10.1016/0022-2836(91)90550-p
7. Sanches CAA, Soma NY (1988) A polynomial-time DNA computing solution for the bin-packing problem. Comput Methods Appl Mech Eng 71(2):197–224. https://doi.org/10.1016/0045-7825(88)90086-2
8. Amirjanov A, Sobolev K (2008) Optimization of a computer simulation model for packing of concrete aggregates. Part Sci Technol 26(4):380–395. https://doi.org/10.1080/02726350802084580
9. Chen JJ et al (2021) Packing optimization of paste and aggregate phases for sustainability and performance improvement of concrete. Adv Powder Technol 26(4):987–997. https://doi.org/10.1016/j.apt.2021.02.008
10. Fadel GM, Wiecek MM (2015) Packing optimization of free-form objects in engineering design. Optimized Packings Appl 105:37–66. https://doi.org/10.1007/978-3-319-18899-7_3
11. Joung YK, Noh SD (2014) Intelligent 3D packing using a grouping algorithm for automotive container engineering. J Comput Des Eng 1(2):140–151. https://doi.org/10.7315/JCDE.2014.014
12. First H and Alpaslan N. An effective approach to the two-dimensional rectangular packing problem in the manufacturing industry. Comput Ind Eng. 2020;148. https://doi.org/10.1016/j.cie.2020.106687
13. Araújo LJP et al (2018) Analysis of irregular three-dimensional packing problems in additive manufacturing: a new taxonomy and dataset. Int J Prod Res 57(18):5920–5934. https://doi.org/10.1080/00207543.2018.1534016
14. Dósa G (2007) The tight bound of first fit decreasing bin-packing algorithm. Int Symp Comb Algorithms Probab Exp Methodol 4614:1–11. https://doi.org/10.1007/978-3-540-74450-4_1

15. Halfin S (1989) Next-fit bin packing with random piece sizes. J Appl Probab 26(3):503–511. https://doi.org/10.2307/3214408
16. Dósa G and Sgall J. Optimal analysis of best fit bin packing. Autom Languages Program. 2014;8572. https://doi.org/10.1007/978-3-662-43948-7
17. Kim BI, Wy J (2010) Last two fit augmentation to the well-known construction heuristics for one-dimensional bin-packing problem: an empirical study. Int J Adv Manufact Technol 50:1145–1152. https://doi.org/10.1007/s00170-010-2572-z
18. Yuan B, Gallagher M (2005) A hybrid approach to parameter tuning in genetic algorithms. IEEE Congr Evol Comput. https://doi.org/10.1109/CEC.2005.1554813
19. Zhan S and Lin J. List-based simulated annealing algorithm for traveling salesman problem. Comput Intell Neurosci. 2016;2016. https://doi.org/10.1155/2016/1712630
20. Wodziak J, Fadel G (1999) Packing and optimizing the center of gravity location using a genetic algorithm. Clemson University, Design Methodology Group
21. Gonçalves J, Resende M (2013) A biased random key genetic algorithm for 2D and 3D bin packing problems. Int J Prod Econ 145(2):500–510. https://doi.org/10.1016/j.ijpe.2013.04.019
22. Feng X et al (2015) Hybrid genetic algorithms for the three-dimensional multiple container packing problem. Flex Serv Manuf J 27:451–477. https://doi.org/10.1007/s10696-013-9181-8
23. Liu C, et al. Optimizing two-dimensional irregular packing: a hybrid approach of genetic algorithm and linear programming. Appl Sci. 2023;22. https://doi.org/10.3390/app132212474
24. Dowsland K et al (2007) A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. Eur J Oper Res 179(3):759–774. https://doi.org/10.1016/j.ejor.2005.03.058
25. Cagan J et al (1998) A simulated annealing-based algorithm using hierarchical models for general three-dimensional component layout. Comput Aided Des 30(10):781–790. https://doi.org/10.1016/S0010-4485(98)00036-0
26. Gomes A, Oliveira J (2006) Solving irregular strip packing problems by hybridising simulated annealing and linear programming. Eur J Oper Res 171(3):811–829. https://doi.org/10.1016/j.ejor.2004.09.008
27. Torres et al. Convex polygon packing based meshing algorithm for modeling of rock and porous media. Int Conf Comput Sci. 2020:. 257–269. https://doi.org/10.1007/978-3-030-50426-7_20
28. Fernandez C et al (2022) Voxel-based solution approaches to the three-dimensional irregular packing problem. Oper Res. https://doi.org/10.1287/opre.2022.2260
29. Pankratov A, Romanova T (2020) Packing oblique 3D objects. Mathematics. https://doi.org/10.3390/math8071130
30. Demir I and Aliaga DG. Near-convex decomposition and layering for efficient 3D printing. Addit Manuf. 2018;;21. https://doi.org/10.1016/j.addma.2018.03.008
31. Fang J, et al. A deep reinforcement learning algorithm for the rectangular strip packing problem. PLoS One. 2023;18. https://doi.org/10.1371/journal.pone.0282598
32. Ren H, Zhong R (2024) An autonomous ore packing system through deep reinforcement learning. Adv Space Res. https://doi.org/10.1016/j.asr.2024.01.061
33. Tian R, et al, Learning to multi-vehicle cooperative bin packing problem via sequence-to-sequence policy network with deep reinforcement learning model. Comput Ind Eng 2023;177. https://doi.org/10.1016/j.cie.2023.108998
34. Miao Y et al (2003) Multi-objective configuration optimization with vehicle dynamics applied to midsize truck design. ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference. pp 1–9. https://doi.org/10.1115/DETC2003/DAC-48735
35. Wu S (2014) Multi-objective optimization of 3D packing problem in additive manufacturing. IIE Annual Conference and Expo. pp 1485–1494
36. Gao L et al. Multi-objective optimization of thermal performance of packed bed latent heat thermal storage system based on response surface method. Renew Energy. 153:669–680. https://doi.org/10.1016/j.renene.2020.01.157
37. Sridhar R. et al. Multi objective optimization of heterogeneous bin packing using adaptive genetic approach. Indian J Sci Technol. 9(48). https://doi.org/10.17485/ijst/2016/v9i48/108484
38. Bello W et al (2024) Multi-physics three-dimensional component placement and routing optimization using geometric projection. J Mech Des 146(8):2024. https://doi.org/10.1115/1.4064488
39. Carrick C, Kim IY (2019) Packaging optimization using the dynamic vector fields method. Int J Numer Meth Eng. https://doi.org/10.1002/nme.6161
40. Gilbert EG et al (1988) A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J Robot Automation 4(2):193–203. https://doi.org/10.1109/56.2083
41. Seelen LJH et al (2018) A granular discrete element method for arbitrary convex particle shapes: method and packing generation. Chem Eng Sci 189:84–101. https://doi.org/10.1016/j.ces.2018.05.034
42. Conway JH, Sloane NJA (1993) Sphere packings, lattices and groups. p 290
43. Song C et al (2008) A phase diagram for jammed matter. Nature 453(7195):629–632. https://doi.org/10.1038/nature06981

Publisher's Note