Spatial Packaging and Routing Optimization of Complex Interacting Engineered Systems

Mohammad M. Behzadi[a], Peter Zaffetti[b], Jiangce Chen[c], Lawrence E. Zeidner[d], Horea T. Ilieş[a,*]

[a]*School of Mechanical Aerospace and Manufacturing Engineering University of Connecticut*
[b]*School of Computing University of Connecticut*
[c]*Department of Mechanical Engineering Carnegie Mellon University*
[d]*RTX Technology Research Center*

Abstract

Designing the 3D layout of interconnected systems (SPI2), which is a ubiquitous task in engineered systems, is of crucial importance. Intuitively, it can be thought of as the simultaneous placement of (typically rigid) components and subsystems, as well as the design of the routing of (typically deformable) interconnects between these components and subsystems. However, obtaining solutions that meet the design, manufacturing, and life-cycle constraints is extremely challenging due to highly complex and non-linear interactions between geometries, the multi-physics environment in which the systems participate, the intricate mix of rigid and deformable geometry as well as the difficult manufacturing and life-cycle constraints. Currently, this design task heavily relies on human interaction even though the complexity of searching the design space of most practical problems rapidly exceeds human abilities.

In this work, we take advantage of high-performance hierarchical geometric representations and automatic differentiation to simultaneously optimize the packing and routing of complex engineered systems, while completely relaxing the constraints on the complexity of the solid shapes that can be handled and enable intricate yet functionally meaningful objective functions. Moreover, we show that by simultaneously optimizing the packing volume as well as the routing lengths we produce tighter packing and routing designs than by focusing on the bounding volume alone. We show that our proposed approach has a number of significant advantages and offers a highly parallelizable, more integrated solution for complex SPI2 designs, leading to faster development cycles with fewer iterations, and better system complexity management. Moreover, we show that our formulation can handle complex cost functions in the optimization, such as manufacturing and life-cycle constraints, thus paving the way for significant advancements in engineering novel complex interconnected systems.

Keywords: 3D spatial packaging of Interconnected Systems with Physical Interactions, Maximal Disjoint Ball Decomposition

1. Introduction

The current practice in designing interconnected 3D systems operating in multi-physical environments follows a largely manual process that depends heavily on human experience. This traditional design methodology, which involves spatial placement of interconnected components and subsystems, interconnect routing, and physics-based performance evaluation are all complex and highly non-linear tasks that quickly overwhelm human cognitive capacities, especially as system complexity reaches moderate levels. Furthermore, effectively optimizing 3D Spatial Packaging of Interconnected Systems with Physics Interactions (SPI2) requires a comprehensive understanding of various system requirements, including manufacturing, assembly, testing, operation, and maintenance [1, 2, 3].

It is a common industrial practice for engineers to approach these problems by modifying existing system designs to fit new requirements. Nevertheless, while common, such a practice is not only labor-intensive but also frequently results in sub-optimal solutions in terms of size, weight, system performance, and reliability. In addition, this common process evaluates most of the downstream constraints, such as those related to manufacturing and product life-cycle, late in detailed design, which often results in significant increases in the number of design iterations, and consequently causes schedule delays and increases product development costs. Hence, novel automated design optimization methods that take into consideration a larger set of constraints up-front, early in conceptual design would make the design process more efficient and result in final designs that not only perform significantly better, but also have remarkably lower total costs. For example, simultaneously considering the optimal component placement, routing, and physical behavior of complex engineered systems that are subject to functional, manufacturing, and life-cycle constraints would have a significant effect not only on the quality of the designs but also on the associated product development costs and schedule.

Various algorithms have focused on the challenges of packing and routing separately [4, 5]. However, the unique demands of SPI2 designs call for a simultaneous solution to spatial packing/layout, routing, and functional evaluation. This concurrent approach has received relatively little attention in the field largely due to its inherent theoretical and computational complexity. A significant factor influencing this complexity in SPI2 designs is that both packing and

*SPI2 preprint: Manuscript published in ASME JMD, 2025.
*Corresponding author
Email address: horea.ilies@uconn.edu (Horea T. Ilieş)

routing problems are each NP-hard problems [6]. Consequently, as the scale and complexity of these systems grow, not only does the number of potential solutions increase exponentially, but the associated computational cost explodes. Additionally, the complex and non-linear characteristics of the packing and routing problem increase the likelihood of the optimizer encountering local optima, a challenge that is less prevalent when tackling the components of the packaging problem – packing and routing – as separate tasks. This complexity amplifies the challenge of effectively addressing the spatial packaging problem, highlighting the need for innovative approaches in this field.

Recently, some algorithms have been proposed to automate SPI2 designs for both 2D [7, 8, 9, 10] and 3D [11, 12] systems. The formulation of these algorithms, focused on minimizing the bounding box volume and on considering limited physical interactions within the system, overlooks the crucial aspect of minimizing the lengths of the interconnects and restrict the geometries of all shapes involved to cubic shapes. In turn, this simplification leads to suboptimal solutions, although these simplifications allow the use of analytical sensitivities in the optimization problem. At the same time, the existing algorithms use the smallest enclosing spheres of each component in their collision detection, which overestimates potential collisions and unnecessarily restricts the design space of potential solutions.

In this paper, we introduce a SPI2 design automation algorithm that leverages the strengths of a new high-performance hierarchical geometric representation, specifically the Maximal Disjoint Ball Decomposition (MDBD)[13], and the recent advances in automatic differentiation. As a result, we effectively remove the current limitations on geometric complexity, enabling the creation of complex yet functionally meaningful objective functions of practical importance. Moreover, by using automatic differentiators, we can optimize both the packing volume and the routing lengths of systems functioning within multiphysics environments. Our proposed approach offers several additional significant benefits, such as being highly parallelizable, providing an integrated and scalable solution for complex SPI2 designs, and leading to faster development cycles with reduced iterations and enhanced management of system complexities. Additionally, our integrated method for 3D packing and routing of complex geometries allows for the consideration of intricate cost functions in the optimization process, such as critical manufacturing and lifecycle constraints. In other words, our proposed approach lays the groundwork for substantial progress in the research and practice of complex engineered interconnected systems.

2. Background

The design problems of 3D packing and interconnect/pipe routing have been the subject of extensive but separate investigations, reflecting their significant relevance across multiple industries, including transportation[14, 15], manufacturing[16, 17, 5], and robotic path planning[18, 19, 20], among others[21, 22, 23, 24, 25]. While numerous studies have focused on addressing 3D packing and interconnect/pipe routing design as separate challenges [5, 26, 27, 28, 29, 30, 31, 32, 33], complex engineering systems, such as aircraft engines with their hundreds of interconnected components, necessitate an integrated approach to solve 3D packing and interconnect routing designs concurrently.

A few prior studies considered the integrated packing and routing design. For instance, the authors in [34] employed a simulated annealing-based algorithm for packing and routing design, optimizing a weighted objective function that includes component packaging density, total routing length, and overall system volume. To circumvent infeasible solutions, collision constraints were integrated into the main objective function as a penalty term. The effectiveness of this method was showcased in a case study involving a heat pump, where the algorithm achieved a reduction of approximately 40 % in interconnect lengths compared to the best existing methods, albeit with a slight compromise on volume optimization. However, the study's reliance on simplified component geometries, such as prisms and cylinders, to model the heat pump's parts limits its applicability to a broader range of real-world scenarios. In another study by [35], the authors proposed a mixed integer linear program tailored to packing problems of rectangular objects aligned with principal axes, allowing only 90-degree rotations. This study incorporates an innovative area estimation for routing flexible connections between objects, utilizing a branch & bound strategy combined with a linear relaxation for wiring area estimation to address the optimization problem. The algorithm's strength lies in its ability to generate multiple solutions for each scenario, providing flexibility to select an optimal solution based on criteria beyond those initially incorporated into the optimization process. However, this flexibility is offset by a marked increase in computational demand, especially in scenarios involving routing, leading to significantly higher computational costs. Additionally, [2] introduced a Constraint Programming (CP) based algorithm for 3D packing and interconnect design, utilizing basic geometric forms (like spheres, cylinders, or rectangular prisms) for components and uniform-diameter cylinder chains for interconnects. The algorithm focuses on optimizing component positions and orientations, as well as the shapes and placements of interconnect cylinders, aiming to minimize both packing volume and interconnect lengths. However, despite these advancements in simultaneous packing and interconnect design, these approaches generally cater to simpler geometries. More critically, they often overlook the physical interactions between components, a fundamental factor in the design of engineering systems [36].

Recently, various algorithms have been developed that not only address packing and routing but also integrate physical aspects into the optimization process. For instance, in [8], a gradient-based optimization algorithm was developed specifically for packaging electro-thermal systems. This algorithm aims to minimize the system's bounding volume, which contains 2D geometries, while also accounting for thermal conduction and fluid pressure loss. In another study, [9] introduced a two-stage design automation approach that methodically enumerates and describes feasible topological layouts for a 2D fluid-thermal system. It then optimizes each layout using a gradient-based design optimization procedure, but with a focus on the physics-based performance of the system. These studies have made notable progress in simultaneously considering packing and interconnect design, along with some physical aspects of the system. However, a major common limitation is the restriction to 2D simple geometries, which limits their utility in addressing real-world design problems.

Very recently, [12] presented a gradient-based algorithm for simultaneous packing and routing of 3D systems and uses analytical sensitivities. This algorithm not only minimizes the bounding box volume but also integrates multiphysics capabilities, such as thermal and hydraulic models, while considering non-interference geometric constraints. Notably, their results show a reduction of the bounding box volume by approximately 90% compared to the start configurations of the optimization cycles. It is important to note that in this approach, every component is assumed to be a 3D cube, and the objective function only contains terms associated with the design volume and the thermal/hydraulic physical phenomena, while ignoring the lengths of the interconnects. Clearly, these simplifying assumptions were made to be able to derive analytical sensitivities, but these same assumptions produce in turn suboptimal solutions. Additionally, the method overestimates the collision predicates because it approximates the distance between components using the distance between their smallest enclosing spheres.

3. Contributions

In this work, we introduce a pioneering SPI2 design automation algorithm that capitalizes on the strengths of high-performance hierarchical geometric representations, specifically of the Maximal Disjoint Ball Decomposition (MDBD) [13], complemented by the use of automatic differentiators. The key contributions of our work are as follows.

1. By employing MDBD, our algorithm can efficiently and accurately compute the collisions between the complex 3D geometries.
2. We leverage automatic differentiation, which allows us not only to relax the constraints on the geometric complexity, but also to consider sophisticated and functionally relevant terms in the objective function, such as those capturing manufacturing and life-cycle constraints. This, in turn, affords for the first time the formulation of SPI2 design problems of practical complexities within an optimization framework. We note that the inclusion of manufacturing and life-cycle constraints is outside the scope of this paper and will be detailed separately.
3. Unlike traditional methods that focus predominantly on minimizing the bounding volume, our algorithm optimizes both the packaging volume and the routing lengths. This dual-focus approach results in more efficient design solutions with tighter packaging and improved functionality.
4. Our proposed method inherits the parallelization and scalability proprieties of sphere-sphere distance computations [37, 38, 39, 40], automatic differentiation [41, 42, 43], and physics solvers [44, 45, 46], and therefore supports parallel and scalable implementations of our SPI2 approach. In turn, our formulation has the potential to accelerate the development process and improve the overall management of system complexities.

The remainder of this article is structured as follows: Section 4 summarizes the key properties and generation of MDBD, and provides a detailed discussion on the formulation of the simultaneous component packing and routing design optimization problem. Then, Section 5 showcases the results derived from various numerical case studies conducted using our proposed optimization methodology. The article concludes with Section 6, where we summarize our findings and outline potential avenues for future research.

4. Method

4.1. Structure of the System Configuration Space

In this section, we introduce a generalized mathematical framework for the configuration space of the interconnected system. As we can see below, this configuration space, which is the Cartesian product of the individual configuration spaces corresponding to the rigid components and the flexible interconnects, is itself a Lie group, and hence a differentiable manifold. This provides the necessary underlying differentiability framework required by gradient-based optimization methods.

Components are assumed to be n-dimensional r-sets [47], which are compact (i.e., bounded and closed) regular semi-analytic subsets of the Euclidean n-space R^n. Every r-set A within an n-dimensional workspace $\mathcal{W} = R^n$ is defined in a Cartesian frame F_A attached to A. Similarly, a Cartesian frame $F_{\mathcal{W}}$ is attached to workspace $\mathcal{W}$. Thus, $F_{\mathcal{W}}$ and F_A are the associated global and local coordinate frames, and therefore the configuration of A in $\mathcal{W}$ is the position and orientation of F_A relative to $F_{\mathcal{W}}$. The set of all possible configurations of A represents its configuration space $\mathcal{C}_A$, which can be described by a list of real parameters corresponding to the rotation $R \in \mathrm{SO}(n)$ and translation $\mathbf{t} \in R^n$. The configuration space $\mathcal{C}_A := \mathrm{SE}(n) = \mathrm{SO}(n) \times R^n$ inherits the Lie group structure of the motion group $\mathrm{SE}(n)$, since the special orthogonal group $\mathrm{SO}(n)$ and R^n are both Lie groups. Importantly, $\mathcal{C}_A := \mathrm{SE}(n)$ is a differentiable manifold since, for every open set of elements of $\mathrm{SE}(n)$, one can define a 1-1 map onto an open set of R^n. A common way to represent elements of the special[1] orthogonal group of rotations $\mathrm{SO}(n)$ is through $n \times n$ orthogonal rotation matrices[2], and a common parametrization of these elements in 3-dimensions is achieved via Euler angles [52], which have been generalized to higher dimensions.

For a given rotation $R \in \mathrm{SO}(n)$ and translation $\mathbf{t} \in R^n$, the relationship between the coordinates of a point $\mathbf{p} \in A$ between the two reference frames, F_A and $F_{\mathcal{W}}$, can be expressed as:

$$\mathbf{p}_{\mathcal{W}} = R\mathbf{p}_A + \mathbf{t}, \tag{1}$$

which treats translation differently (as an addition) from rotation (as a multiplication). As usual, the homogeneous coordinates can provide the unifying framework that allows any rigid body transformation as well as more general affine transformations to be treated as matrix multiplications.

Thus, a rigid body transformation $T \in \mathrm{SE}(n)$ is defined as:

$$T = \left\{ \begin{bmatrix} R & \mathbf{t} \\ 0_n & 1 \end{bmatrix} : R \in \mathrm{SO}(n), \mathbf{t} \in R^n \right\}. \tag{2}$$

and therefore equation (1) becomes:

$$\mathbf{p}'_{\mathcal{W}} = T\mathbf{p}'_A \tag{3}$$

[1] This adjective reflects the fact that we exclude the orthogonal matrices with negative determinants.

[2] Another common representation of rotations is provided by the quaternion algebra [48] discovered in 1866 by Hamilton [49]. In fact, rigid body transformations can be represented as unit dual quaternions [50], which were discovered by Clifford in 1882 [51].

where $\mathbf{p}'_{\mathcal{W}}$ and $\mathbf{p}'_A$ are the homogeneous versions of vectors $\mathbf{p}_{\mathcal{W}}$ and $\mathbf{p}_A$.

Our r-sets are 3-dimensional pointsets embedded in the 3-dimensional Euclidean space R^3, and SE(3) is a differentiable manifold and a Lie group. As a result, this Lie group structure supports the formulation of gradient-based optimizations involving this particular r-set. Thus, for a set of m rigid r-sets embedded in R^3, each such r-set will generate a separate 6-dimensional configuration space. Collectively, the system of m r-sets will have $6m$ degrees of freedom corresponding to a combined configuration space

$$\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \ldots \times \mathcal{C}_m, \tag{4}$$

where $\times$ denotes the Cartesian product and each C_i has the structure of SE(3). Based on Theorem 6.4, p82, from [53], the Cartesian product of 2 Lie groups is a Lie group. Hence, the combined configuration space $\mathcal{C}$ is a differentiable manifold and supports gradient-based optimizations.

While the rigid r-sets described above have exactly 6 degrees of freedom in 3D, the interconnects are flexible components with generally infinitely many degrees of freedom, which makes the problem intractable. However, by assuming some parametrization of these interconnects that bounds the corresponding number of degrees of freedom, we can formulate the combined configuration space of the rigid r-sets as well as the flexible but parameterized interconnects. For example, one such parametrization inspired by practical applications can be expressed in terms of generalized cylinders generated by d disks of potentially different radii placed along the curved center axis and perpendicular to the center axis, and some predefined interpolation between successive disks. For such an interconnect, the configuration space defined by the centers of the disks and their radii is $(R^3 \times R)^d$, which[3] is a Lie group.

With this parametrization in place, the combined configuration space of the interconnected system of m rigid components and p interconnects becomes:

$$\mathcal{C}_{\text{system}} = \mathcal{C}_1 \times \ldots \times \mathcal{C}_m \times (R^3 \times R)^{dp}. \tag{5}$$

which is also a Lie group and therefore a differentiable manifold. An element of $\mathcal{C}_{\text{system}}$ specifies the configurations of all rigid components of the system, all the disks of all the interconnects, as well as their radii relative to the reference coordinate frame.

4.2. Problem Formulation

The SPI2 optimization problem can generally be formulated in compact form as the constrained minimization:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{subject to:} \\ &\mathbf{g}_{collision} \leq 0, \\ &\mathbf{g}_{physics} \leq 0, \end{aligned} \tag{6}$$

where the objective $f(\mathbf{x})$ contains functional terms, and the constraints $\mathbf{g}$ are vector functions. In this paper, our objective function contains terms corresponding to the packing volume and the total length of the interconnects, and, therefore,

$$f(\mathbf{x}) := w_v f_v + w_r f_r \tag{7}$$

Here, w_v and w_r represent the weights attributed to the dual objectives of minimizing packing volume and routing length, respectively. Additionally, $\mathbf{g}_{collision}$ and $\mathbf{g}_{physics}$ are the constraints within the optimization framework ensuring that in the final solution the components and the interconnects do not collide and meet the physical requirements. In this section, we offer an in-depth discussion of the geometric representations and the formulation of the gradient-based optimization process, including the specific design variables, objective functions, and constraints that are critical for successfully navigating the challenges of simultaneous packing and routing in the SPI2 design domain.

4.2.1. Geometric Representation

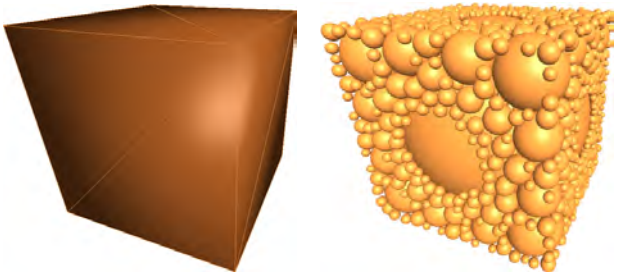
A crucial aspect of the computational approach must involve the selection of an appropriate geometric representation that allows efficient computations of the terms that form the objective function.

There have been a number of geometric representations used in existing approaches focused on 3D packing problem alone, including surface meshes [54, 55], voxels [5, 28], and simple primitives [56, 12]. However, all existing algorithms either are not gradient-based, such as genetic algorithms and simulated annealing, or use blocks and cylinders as geometries. Moreover, the existing approaches do not scale up well.

One recent approach to packing that does achieve impressive performance on a simplified version of the packing problem for additive manufacturing [5] uses indicator functions defined over voxels in conjunction with Fast Fourier Transforms (FFT), and employs a subset of the computational machinery discussed in [57, 58, 59]. The packing is defined as a search for optimal configurations and is accelerated by FFT, coupled with a translational disassembly based on a flood-fill algorithm followed by a repeated reassembly process. Indicator functions are not differentiable, which is why bump functions [58] could be used instead to obtain the required differentiability. Note that this formulation addresses the specific problem of packing 3D shapes inside a known 3D printing volume.

In this paper, we adopt the Maximal Disjoint Ball Decomposition (MDBD) [13], as illustrated in Figure 1, as a proxy geometric representation of 3D geometries involved in the SPI2 design problem. This choice was driven by several different factors. First, MDBD is a genuine proxy geometric representation that further allows the consideration of input geometries using any valid geometric representation. Second, as a union of hierarchically placed maximal disjoint balls, it supports efficient distance queries [60, 61] and massive parallelization on the GPU. Third, complex geometric components can be represented at various levels of details that can be established at run time [57]. Finally, complex geometric components can be easily represented with collections of simple and rotation-invariant spherical primitives whose definitions are very similar across spaces of different dimensions. A possible implication of the latter

[3] We note that this formulation of the configuration space of an interconnect defined this way requires additional assumptions of how one transitions between sections of the same interconnect that have different radii. One initial simplifying formulation can assume that each interconnect has a constant cross-section, in which case the corresponding configuration space would be $(R^3)^d \times R$.



(a) (b)

Figure 1: (a) The boundary representation of a cube component (r-set) and its MDBD representation (b).

property will be discussed in section 5.3. A detailed explanation of MDBD computation can be found in Section 6.1. It is important to note that, similar to other geometric representations, solving boundary value problems directly on the MDBD has not yet been formulated. However, an analysis workflow and other possible strategies are discussed below.

In what follows, a truncated MDBD model of a shape A consisting of t spheres is denoted by $\bar{\mathbf{A}} = \bigcup_{i=1}^{t} s_i$.

A suitable representation of the geometry of the interconnects is also essential. As described in section 4.1, one can conceptualize a large class of interconnects as generalized cylinders generated by circular cross-sections swept along a curve generator. This in turn would require the implementation of an efficient distance-to-curve algorithm - a curve that is controlled by a prescribed set of design variables depending on the curve representation, and a radius function defined along the curve. Without loss of conceptual generality, in this work, we assume that the circular cross-sections for a given interconnect have equal radii and that the generators are comprised of k piecewise linear curves connecting $d = k + 1$ nodes. Hence, each interconnect will be defined by $3d + 1$ degrees of freedom, where the additional degree of freedom corresponds to the value of the radius. Note that the first and the last nodes in the node sequence are connected to the respective components connected by the interconnect, and therefore the number of free degrees of freedom is $3(d-2)$ since the radius is assumed to be known.

4.2.2. Design Variables

In our optimization framework, the design variables consist of the parameters corresponding to the translations and rotations applied to the components, as well as translations applied to the routing nodes.

Given a specific component $\bar{\mathbf{A}}_i$, its MDBD spheres can be considered to be rigidly attached to each other. Hence the position and orientation of $\bar{\mathbf{A}}_i$ in $F_{\mathcal{W}}$ can be traced based on the position and orientation of the local frame $F_{\bar{\mathbf{A}}_i}$ attached to $\bar{\mathbf{A}}_i$. In this paper, we use the Euler angles to parameterize rotations and position vectors to trace the position of the origin of $F_{\bar{\mathbf{A}}_i}$. Consequently, equation (3) can be used to convert any point from $F_{\bar{\mathbf{A}}_i}$ to $F_{\mathcal{W}}$.

The location of each piecewise linear routing is determined in space by the location of its $d-2$ unconstrained nodes. Thus, for m components and p interconnects, the design variables for our problem can be written as

$$\mathbf{x} := \{\mathbf{t}_{\bar{\mathbf{A}}_1}, ..., \mathbf{t}_{\bar{\mathbf{A}}_m}, R_{\bar{\mathbf{A}}_1}, ..., R_{\bar{\mathbf{A}}_m}, \mathcal{T}_1, ..., \mathcal{T}_p\} \tag{8}$$

where each matrix $\mathcal{T}_i$ contains the translation vectors for $d-2$ nodes. Thus, the total number of variables is $6m + 3p(d-2)$.

4.2.3. Packing Volume

The system's packing volume, denoted by $f_v(\mathbf{x})$ in equation (7), is defined here as the smallest axis-aligned box enclosing all components and interconnects of the entire system determined by the design variables $\mathbf{x}$ from equation (8). Thus, assuming that each MDBD contains t spheres, we have

$$\begin{aligned} f_v(\mathbf{x}) = \prod_{\alpha=1}^{3} [\max(&\max_{i\in\{1,tm\}} (||\text{proj}_{x_\alpha}(\mathbf{c}_i)|| + r_i), \\ &\max_{i\in\{1,p\} j\in\{2,d-1\}} (||\text{proj}_{x_\alpha}(\mathbf{n}_{ji})|| + d_{ji})) \\ &- \min(\min_{i\in\{1,tm\}} (||\text{proj}_{x_\alpha}(\mathbf{c}_i)|| - r_i), \\ &\min_{\substack{i\in\{1,p\} \\ j\in\{2,d-1\}}} (||\text{proj}_{x_\alpha}(\mathbf{n}_{ji})|| - d_{ji}))] \end{aligned} \tag{9}$$

where $\mathbf{c}_i$ contains the coordinates of the center of sphere s_i of radius r_i; $\mathbf{n}_{ji}$ represents the coordinates of free node j of interconnect i having a cross-sectional radius d_{ji}; $\text{proj}_{x_\alpha}(\mathbf{a})$ is the projection of a vector $\mathbf{a}$ along the three coordinate axes x_1, x_2, and x_3 of the global frame; and $||\cdot||$ is the Euclidean L_2 norm.

4.2.4. Length of Interconnects

Additionally, our objective function incorporates the total length of the interconnects, denoted by f_r in equation (7), which is calculated as the aggregate length of all cylinders within the system. Thus, the total length of the interconnects is determined by:

$$f_r(\mathbf{x}) = \sum_{i=1}^{p} \sum_{j=1}^{k+1} ||\mathbf{n}_{(j+1)i} - \mathbf{n}_{ji}|| \tag{10}$$

4.2.5. Collision Detection

During optimization, collisions may occur between two components, between components and interconnects, or between two interconnects. The detection of collisions between any two parts within the system is typically achieved by calculating the minimum signed distance between them, where a negative distance indicates a collision.

A significant advantage of employing MDBD lies in its ability to facilitate both efficient and accurate distance calculations. This efficiency is due to the ability to determine the distance between MDBD models by calculating the distance between the spheres that represent these models. Thus, for a system composed of a set of m components and p interconnects , the overall minimum distance for the system becomes:

$$\begin{aligned} \mathbf{d} = [&\min_m \min_m d(s_i, s_j), \min_m \min_p d(s_i, \mathbf{R}_j), \\ &\min_p \min_p d(\mathbf{R}_i, \mathbf{R}_j)] \end{aligned} \tag{11}$$

where $d(s_i, s_j)$, $d(s_i, \mathbf{R}_j)$, and $d(\mathbf{R}_i, \mathbf{R}_j)$ are distances between two spheres, a sphere and an interconnect, or between two interconnects, respectively. It is important to highlight that for the distance calculations in equation (11) we assume that each interconnect is a collection of cylinders of constant thicknesses, as described above. Deviating from this assumption necessitates the adoption of suitable distance computation methods, such as those detailed in [62, 63, 64].

4.2.6. Physical Interactions

The physical interactions within SPI2 design as formulated here are not confined to any single type of physical interaction. In fact, one can include any physical interaction for which an appropriate cost term can be included in the formulation outlined in the equation (6). Furthermore, various solvers can be employed to model the system's physics, including finite element analysis (FEA), boundary element method (BEM) [65], or analytical approaches. In this work, we chose to model the thermal interactions between components as an example. However, one can similarly include other types of physical interactions or incorporate the interconnects as well.

The initial step in modeling thermal interactions through finite element analysis (FEA) involves projecting the MDBD components onto a spatial discretization. For a given sphere s centered at $\mathbf{c}$ and having a radius r, and a voxel element centered at $\mathbf{v}$, the signed distance function that quantifies the separation between the sphere and the mesh element is calculated as follows:

$$d(s, v) = ||\mathbf{c} - \mathbf{v}||_2 - r \tag{12}$$

The signed distance may be negative, zero, or positive, where a negative value indicates that the sphere intersects the voxel element. Given that the density of each element ranges from 0 to 1, we employ a Logistic function to map the signed distance onto the interval $[0, 1]$. This function approaches 1 as the overlap increases and approaches 0 with the positivity of $d_{SV}(s, v)$, signifying disjointness. The specific Logistic function utilized in our model is:

$$Y(s, v) = \frac{1}{1 + e^{20d(s,v)}} \tag{13}$$

We apply equations (12) and (13) iteratively across all spheres in the system and all voxel elements to determine the necessary density values for our FEA analysis.

We further assume that the system's thermal model is in a steady-state condition, with uniform thermal conductivity across all components. To derive the temperature distribution, we employ the weak form of the differential equations relevant to the FEA of the discretized domain, framing it as a linear problem:

$$K\mathbf{T} = \mathbf{P} \tag{14}$$

Here, $\mathbf{T}$ represents the discretized field of temperature solutions, and K is the global stiffness matrix, which is assembled from the stiffness matrices of individual elements, denoted as k_{el}. Additionally, $\mathbf{P}$ signifies the global load vector, comprised of the load vectors of individual elements, p_{el}. We use the following definitions for k_{el} and p_{el}:

$$k_{el} = (\rho_{min} + (1 - \rho_{min})\rho_e^{\alpha}) \left[\int_{\Omega_e} B^T \kappa B \, d\Omega_e - \int_{\partial\Omega_h} hNN^T d\partial\Omega_h \right]$$

$$p_{el} = \rho_A^{\alpha} \int_{\Omega_e} QN \, d\Omega_e + \int_{\partial\Omega_h} hT_{env} N \, d\partial\Omega_h$$

Here, ρ_e represents the elemental density after projecting the entire layout, with α serving as a penalization parameter designed to reduce intermediate densities between 1 and 0, thereby minimizing the gray area between solid and void regions; N and B correspond to the element shape function and its gradient, respectively; κ denotes the matrix of thermal conduction coefficients; Q is the heat flux per unit volume within the domain; T_{env} is the temperature of the convecting fluid along the $\partial\Omega_h$ boundary segment; h is the convection coefficient; ρ_A is the elemental density after projecting the A^{th} component, to account for its specific heat generation; ρ_{min} is employed within the stiffness matrix to prevent numerical instability that may arise from regions with zero density. Upon determining the thermal field, the temperature at the location of the largest sphere within each component is taken to represent the component's corresponding temperature.

4.2.7. Optimization Solver

Now that we have all the constituents required to solve our SPI2 design problem, we can proceed with the solution procedure. To solve the constraint optimization problem formulated in Equation 6, one can use traditional approaches such as the Augmented Lagrangian Methods implemented in known large-scale solvers. However, in our prototype implementation, we transformed the constrained optimization into an unconstrained formulation to simplify the integration with automatic differentiators by adding the constraints as weighted terms of the objective function:

$$\min_{\mathbf{x}} f := w_v f_v + w_r f_r + w_A g_A + w_{AR} g_{AR} + w_R g_R + w_T g_T \tag{15}$$

where ws are the weights assigned to each specific term and

$$\begin{aligned} g_A(x) &= \frac{1}{\min_m \min_m d(s_i, s_j)} \\ g_{AR}(x) &= \frac{1}{\min_m \min_p d(s_i, \mathbf{R}_j)} \\ g_R(x) &= \frac{1}{\min_p \min_p d(\mathbf{R}_i, \mathbf{R}_j)} \\ g_T(x) &= \frac{1}{T_{max} - T_{components}} \end{aligned} \tag{16}$$

where T_{max} is the maximum allowable temperature for each component, while $T_{components}$ is the average temperature of each component as determined through Finite Element Analysis (FEA).

A key aspect of our gradient-based optimization approach is the use of a gradient-based optimizer known as Adam

Table 1: The default setting for each epoch in the optimization process.

	w_v	w_r	w_A	w_{AR}	w_R	w_T	Step size
Epoch 1	1	1	1	1	1	200	0.005
Epoch 2	2	2	1	1	1	200	0.005
Epoch 3	4	1	2	2	2	100	0.001
Epoch 4	6	1	2	2	2	100	0.0005
Epoch 5	6	2	2	2	2	100	0.0005
Epoch 6	8	2	2	2	2	100	0.0001

[66], which is extensively utilized in the training of deep learning models. Note that we compute the gradient of the objective function relative to the design variables by using established automatic differentiation commonly used in the training processes of deep learning. This is a key attribute of our approach that allows us to handle highly complex geometries as well as practically any other physical terms, such as manufacturing and life-cycle constraints, into the objective function. This, however, is outside the scope of this paper and is the subject of ongoing research. Finally, our optimization algorithm is implemented using PyTorch, which also allows us to conveniently leverage GPUs to efficiently solve the optimization problem.

5. Results and Discussion

5.1. Overall Results

We designed four distinct scenarios to test the efficacy of our optimization algorithm. The first scenario encompasses three benchmarks with varying numbers of components and interconnects, for which optimal solutions are known. Here, we compare our algorithm's outcomes with these optimal solutions. The second scenario illustrates the algorithm's performance on examples that range from relatively simple to complex. In the third scenario, we exclude the interconnect length term from the objective function and assess how this modification impacts the results compared to the second scenario. Lastly, in the fourth scenario, we incorporate physical interactions into the objective function of the test cases from the second scenario, examining their influence on the solutions.

For each example, we compute a truncated MDBD with 100 spheres and the interconnects consist of 6 cylindrical segments. Moreover, the optimization is conducted over five epochs, with a cap of 20,000 iterations per epoch. The weights assigned to each term in the objective function vary, and an epoch concludes upon reaching the maximum iteration limit or the value of the objective function does not decrease or if a collision occurs. In the event of a collision, the system reverts to its pre-collision state and proceeds to the next epoch. Table 1 outlines the default settings for each epoch. All the experiments are done with the default setting unless otherwise indicated.

5.1.1. Benchmarks Cases: Equal Sized Cubes

We defined three benchmarks to evaluate if our proposed algorithm could achieve optimal packaging results. The first benchmark involves three equal-sized and interconnected cubes, as illustrated in Figure 2a, whose minimal total volume is obtained when the three cubes are aligned with mutually parallel and coplanar faces. Note that in this configuration the lengths of the interconnects may not be minimal so we expect to converge to this solution when the bounding volume is given a much larger priority than that of the other length-related terms in the loss function. Initially, the bounding box measures 11.56 x 6.56 x 5.54 units, totaling a volume of 420.11 cubic units, with the interconnects spanning 20.24 units in length. The solution that the optimizer converges to with the parameters detailed in Table 1 is depicted in Figure 2b, and shows the capability of the algorithm to converge to the optimal solution. The bounding box of the optimal configuration measures 5.10 x 1.60 x 1.50 units, or a drastic 97.1% decrease compared to the initial volume of 12.24 cubic units. It's important to note the minor gap between components in Figure 2b is an artifact of our interconnect placement and the necessity to prevent collisions between components and interconnects. Furthermore, the total interconnect length of the optimal configuration is reduced to 7.06 units, or a 65.11% reduction. By changing the weights of the terms in the objective function, the optimizer will converge to different solutions. For example, by weighing the length term the most ($w_r = 20$) in the first epoch, we converge to the result shown in Figure 2c. In this case, the bounding box dimensions become 3.26 x 3.26 x 1.50 units, yielding a total volume of 15.96 cubic units, and an interconnect length of 2.36 units. This alternate arrangement of Figure 2c exhibits a larger volume but significantly shorter total length of the interconnects compared to the configuration in Figure 2b.

In our second benchmark, we examined a setup involving four cubic components connected by four interconnects, as shown in Figure 2d. The initial bounding box dimensions were 14.07 x 9.56 x 6.55 units, with a total volume of 881.03 cubic units and the interconnects measuring 30.88 units in length. For this benchmark, we set $w_r = 4$ in the first epoch and $w_r = 3$ in the second epoch. The optimal configuration, depicted in Figure 2e, illustrates the optimizer's ability to converge to the global optimum of the bounding box volume. The optimized dimensions of the bounding box are 3.27 x 3.27 x 1.50 units, drastically reducing the volume to 16.4 cubic units—a 98.13 % decrease. Furthermore, the optimizer reduced the interconnect length to 6.16 units, marking an 80.05 % decrease. In a similar manner with the first scenario, the final configuration displays minor gaps between components resulting from the deliberate choice to avoid collisions. Again, by changing the weights of the terms in the objective function we reach a different optimal configuration illustrated in Figure 2f. This alternative setup, with bounding box dimensions of 6.91 x 1.59 x 1.50 units and a volume of 16.48 cubic units, closely matches the volume seen in Figure 2e. However, the interconnect length in Figure 2f is 11.96 units, which is nearly double that of the configuration shown in Figure 2e.

The third benchmark has a higher complexity than that of the prior ones, with six cubic components that initially rest on a plane and are linked by eight interconnects, as shown in Figure 2g. The bounding box enclosing these 6 objects and interconnects has 21.55 x 21.56 x 1.80 units, equating to a volume of 836.31 cubic units, and an interconnect length totaling 90.21 units. In this case, we adjusted $w_v = 25$ for the initial epoch, ensuring the model initially focuses on reducing the system's volume rather than the length of the interconnects. The resulting configuration, illustrated in Figure 2h, features bounding box dimensions of 4.88 x 4.61 x 1.50 units, decreasing the volume to 25.66 cubic units or a 96.93 % reduction. The total interconnect length also saw

a substantial decrease to 13.31 units, which amounts to an 85 % decrease. This configuration appears to be the global optimum for the bounding box volume, given that it tightly wraps around the objects. The other alternative solution possessing a tightly wrapped bounding box, i.e., when the 6 objects are aligned linearly, theoretically has the same volume[4], so this particular problem has at least two minima. Running the optimization with default settings for this benchmark yields a different configuration shown in Figure 2i, where minimizing interconnect length takes precedence over volume reduction. The corresponding bounding box is 4.83 x 6.50 x 1.50 units, resulting in a volume of 47.09 cubic units –nearly double that of Figure 2h. However, the total length of the interconnects is reduced to 9.02 units, which is significantly less than that of the previous configuration.

Table 2 presents a summary of the bounding box volumes and interconnect lengths for the three benchmark configurations. Note that by adjusting the weights of the different terms of the objective function, as specified in Table 1, the algorithm can be tailored to prioritize different optimization goals.

5.1.2. Volume and Length of Interconnect Minimization for Various Geometries

The current leading algorithm for SPI2 design, as detailed in [12] utilizes cubic geometries for system components, which limits its applicability to real-world scenarios. However, with the increase in geometric complexity come challenges that must be handled differently than the formulation presented in [12], including efficient collision detection between complex models and the lack of analytical sensitivities. We demonstrate here the performance of our framework in handling systems with varied complex geometries and for this purpose, we introduce three systems of growing complexities.

The first system, shown in 3a, consists of six cubes and eight interconnects. To illustrate how the interconnect thickness affects the solution, we considered both thin as well as "fat" interconnects, as shown in Figure 3b & 3c, where the thicknesses in Figure 3c are 80 times greater than in Figure 3b. Both systems start from the same initial configuration from Figure 3a, and the optimization was conducted using the parameters detailed in Table 1. Initially, the system's bounding box had a volume of 2797.66 cubic units, and the total interconnect length was 118.11 units. For the system with thin interconnects, the final configuration in Figure 3b had a volume of 27.50 cubic units and a total interconnect length of 14.30 units, showing reductions of approximately 99% in volume and 88% in interconnect length. In contrast, the system with thick interconnects, shown in Figure 3c, had a final volume of 87.7 cubic units and a total interconnect length of 25.8 units, representing reductions of 96.9% in volume and 79% in interconnect length. The larger volume in the thick interconnect system resulted from the collision constraints that limit further reductions. These examples demonstrate that our framework can effectively handle interconnects with varying thicknesses.

The second system, depicted in Figure 4a, includes six components and nine interconnects, utilizing a variety of shapes such as prisms, cylinders, and plates. Initially, this system's bounding box measured 20.86 x 21.07 x 3.34 units, yielding a total volume of 1467.99 cubic units and an initial interconnect length of 107.73 units. Optimization, conducted with parameters detailed in Table 1, resulted in the final configuration shown in Figure 4b. The bounding box of the resulting configuration was reduced to 3.50 x 3.30 x 1.99 units, with a volume of 22.98 cubic units, and the total length of the interconnects shortened to 11.20 units, achieving a volume reduction of 98.43% and a 90% reduction in the length of the interconnects.

The third system, featuring ten components and fifteen interconnects, is presented in Figure 4c. It incorporates additional more complex geometries, including L-shaped and X-shaped components. The initial bounding box dimensions were 20.66 x 20.09 x 3.34 units, with a total volume of 1386.30 cubic units and an initial interconnect length of 116.27 units. The optimization converged to the solution shown in Figure 4d, whose bounding box measured 5.43 x 5.32 x 0.99 units, resulting in a volume of 28.59 cubic units and an interconnect length of 29.58 units. This equates to a volume reduction of approximately 98% and a 75.4% reduction in interconnect length.

These outcomes for these systems demonstrate that utilizing MDBD alongside automatic differentiation enables the optimizer to efficiently evaluate various objective function terms, significantly reducing both the bounding box volume and interconnect lengths for systems with complex geometries. We note that these attributes of our proposed algorithm render it as the first practical approach for optimizing complex interconnected systems of industrial relevance.

5.1.3. Only Volume Minimization Vs Volume & Length Minimization

A key distinction between our proposed algorithm and the current state-of-the-art is the fact that we can include in the objective function terms representing the interconnect lengths. To illustrate the impact of incorporating terms corresponding to the lengths of the interconnects on achieving configurations that have better overall functional attributes, we revisited the examples from Section 5.1.2 and excluded the length term from the objective function. Figure 5 provides a comparative analysis, showcasing the outcomes with and without the interconnect length consideration.

In the case of the six-component system, incorporating both volume and interconnect length into the objective function yielded a configuration with a total volume of 22.98 cubic units and interconnect length of 11.20 units, as depicted in Figure 5a. In contrast, omitting the interconnect length term resulted in a configuration with significantly longer interconnects, as shown in Figure 5b, where the total volume increased slightly to 23.42 cubic units, and the interconnect length surged to 35.51 units. This comparison underscores the observation that the exclusion of the interconnect length from the objective function not only marginally increases the volume but also significantly expands the interconnect length.

The same effect can be observed for the ten-component system for which the combined volume and length terms in the objective function led to a final volume of 28.59 cubic units and an interconnect length of 29.58 units. This system is illustrated in Figure 5c. However, by excluding the length term we obtain the configuration shown in Figure 5d, hav-

[4]For 6 unit cubes, the 3×2 alignment has a volume of 6 cubic units, and the 6×1 alignment has the same volume, so both configurations are local minima.

Table 2: The initial and final volume and length of interconnects for each benchmark shown in Figure 2.

	Benchmark 1			Benchmark 2			Benchmark 3		
	Initial (Figure 2a)	Figure 2b	Figure 2c	Initial (Figure 2d)	Figure 2e	Figure 2f	Initial (Figure 2g)	Figure 2h	Figure 2i
Volume	420.11	12.24	15.94	881.03	16.4	16.48	836.31	25.66	47.09
Length	20.24	7.06	2.36	30.88	6.16	11.96	90.21	13.31	9.02

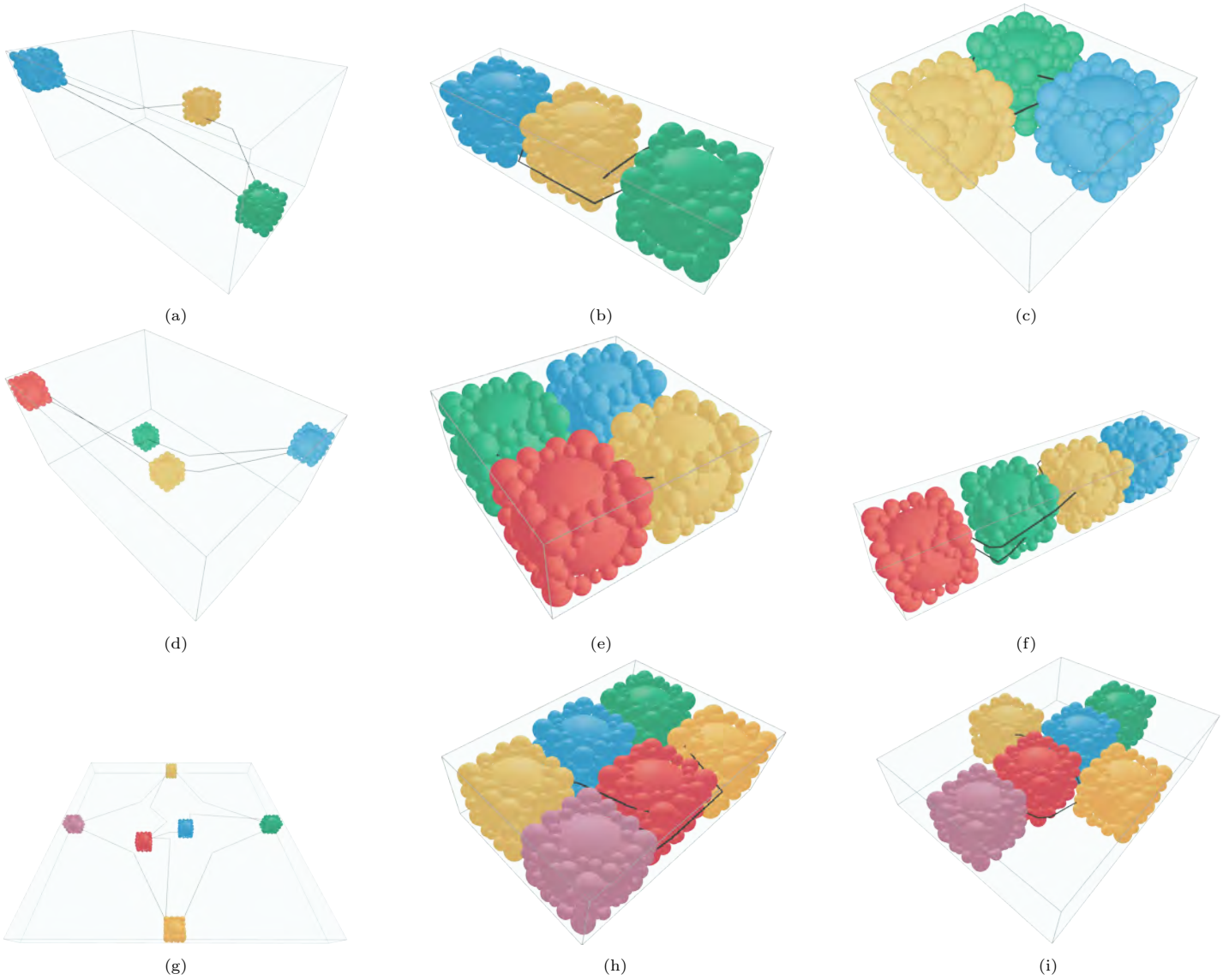


Figure 2: Initial and optimized configurations obtained by different optimization parameters for three benchmarks: (a), (b), and (c) show a system with three components and interconnects, (d), (e), and (f) with four components and interconnects, and (g), (h) and (i) with six components and eight interconnects.

ing a total volume of 36.20 cubic units and an interconnect length of 83.92 units.

A possible explanation of this behavior is that the absence of the length term does not stimulate a repositioning of the interconnects unless directly affected by the bounding box volume.

5.1.4. System with Physical Interactions

One of the main objectives in any SPI2 design is to incorporate relevant physical interactions in the optimization formulation. As argued above, our formulation allows the inclusion of practically any physical interaction whose cost can be quantified and included in the objective function. To demonstrate this capability, we introduce a new thermal-related constraint into the objective function. In this scenario, components are positioned within a prescribed thermal field while exhibiting a unique heat dissipation rate, and each component has a maximum allowable temperature that cannot be exceeded.

As discussed in section 4.2.6, a number of solution methods can be used to simulate the physical behavior, including traditional FEA – the avenue pursued here, and mesh-free methods such as [67].

Initial configurations for systems with six and ten components are illustrated in Figures 6a and 6c, respectively. We simulated heat convection on the cooler side at ambient

(a) (b) (c)

Figure 3: Initial and final configurations for a system with varying interconnect thicknesses: (a) initial configuration of the system, (b) final configuration with thin interconnects, and (c) final configuration with thick interconnects.

Table 3: The heat dissipation rates, critical temperature, and final temperatures for each component in the system shown in Figure 6a.

Component	Heat dissipation rates ($\frac{W}{m^3}$)	Critical temperature (°C)	Final temperature (°C)
Cube (yellow)	5000	20	4
Prism (blue)	4000	60	17
Flat plate (dark green)	3000	50	11
Prism (red)	3000	70	25
Prism (orange)	2000	70	24
Disk (dark pink)	5000	95	18

Table 4: The heat dissipation rates, critical temperature, and final temperatures for each component in the system shown in Figure 6c.

Component	Heat dissipation rates ($\frac{W}{m^3}$)	Critical temperature (°C)	Final temperature (°C)
Cube (yellow)	5000	50	11
Prism (blue)	4000	70	17
Flat plate (dark green)	3000	70	33
Prism (red)	3000	90	37
Prism (orange)	2000	80	22
Disk (dark pink)	5000	60	16
Cylinder (brown)	1000	60	15
L-shaped (green)	3000	70	24
X-shaped (purple)	2000	60	26
Cube (blue)	2000	65	21

temperature, 0 °C, and a warmer face at a steady temperature of 100 °C, with the remaining boundaries of the enclosure assumed to be thermally insulated. The heat transfer coefficient on the cooler face boundary was set to 35.80 $\frac{W}{m^3}$.

Tables 3 and 4 detail the heat dissipation rates, critical temperatures, and final temperatures of each component for the six- and ten-component systems. The resulting configurations are depicted in Figures 6b and 6d. For the six-component system, the optimizer achieved a volume reduction of approximately 95% and a 70% decrease in interconnect length. Similarly, for the ten-component system, volume and interconnect length were reduced by 96% and 71%, respectively. Analyzing the final versus initial configurations for both setups reveals that components with lower critical temperatures and higher dissipation rates moved closer to the cooler face. These outcomes affirm that our algorithm can effectively integrate physical dynamics into the optimization process, with the potential for extending to other physical interactions, such as pressure head loss in interconnects.

Table 5: Computational times for evaluating each term in the optimization process for systems with six and ten components, expressed in $\frac{Second}{Iteration}$.

Systems	Volume Calculation	Collision Detection	FE Analysis	Gradient Calculation
System 1 (Figure 6a)	0.0012	0.021	0.35	0.2
System 2 (Figure 6c)	0.0014	0.031	0.36	0.25

5.2. Time Complexity

Table 5 shows the time per iteration for each calculation within the optimization process, conducted on a PC equipped with an Intel Core i7-10750H CPU, 32 GB of RAM, and an NVIDIA GeForce RTX 2070 Super GPU. The data highlights the efficiency of volume and collision calculations, underscoring the advantages of using MDBD. Notably, MDBD's hierarchical representation facilitates rapid adjustments in detail, further accelerating collision calculations. Moreover, the current implementation of collision detection uses brute force to calculate the distance between the parts in the system. This can be further improved by employing methods that heavily take advantage of multicore GPU units, such as FFT [57, 5]. As can be seen in Table 5, the computational time increases with the addition of more components due to a rise in the number of distance calculations, affecting the time required for collision term calculations. The time required by FEA is influenced by the voxel size and in this work we used a uniform voxelization of the same size for both examples.

Furthermore, the use of automatic differentiation in our optimization problem necessitates the maintenance of a "computational graph" for each design variable, with gradient calculations requiring traversals of this graph. While potentially more costly than analytical gradient calculations partly due to the complexity of the graph, automatic differentiation enables the incorporation of complex terms in the objective function and the handling of complex geometries. It is important to observe in Table 5 that the computational time required by automatic differentiation is closely tied to the number of design variables, with more variables leading to larger computational graphs. Unsurprisingly, the gradient calculation time increases with the system size, as more components contribute to a more complex computational graph.

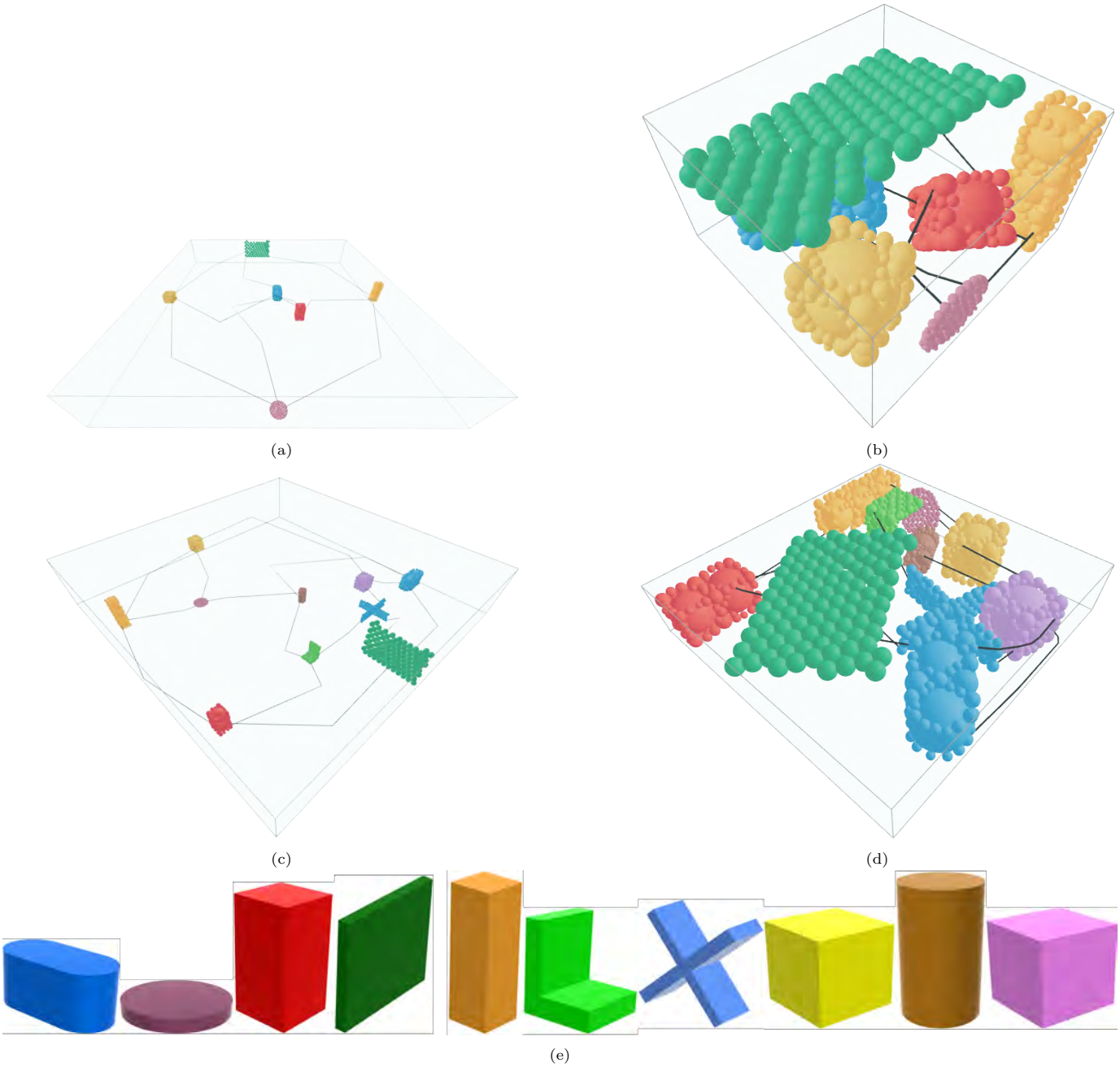


Figure 4: Initial and optimized configurations for two systems: (a)&(b) a simpler one with six components and nine interconnects, and (c)&(d) a more complex one with ten components and fifteen interconnects. Figure (e) shows the components used in the systems.

5.3. Entanglements, Dimensional Elevation, and MDBD

Given the interconnected nature of the systems under consideration, the optimizer may face entanglements generated by the imposed non-collision constraints, which can "lock" the optimizer into a suboptimal configuration. For instance, beginning with the setup in Figure 7a, the optimizer might converge to the state shown in Figure 7b, where it cannot further reduce the volume or length of the interconnects without causing a collision between an interconnect and the yellow component.

To circumvent such entanglements, we suggest elevating the dimension of the space in which optimization takes place. By adding a new dimension to the 3D Euclidean space of the system, which would turn every 3D point, for example, into a 4D point, we can elevate the dimension of the functional space to a 4D space. By performing the optimization in the augmented 4D space rather than the original 3D space, we can avoid entanglements during the optimization. The last step upon convergence to an optimal solution, is to project the optimal configuration from 4D back to the original 3D space.

General practical dimensional elevations and reductions for complex 3D/4D shapes are not known. Here is where MDBD can provide the critical and missing capability for dimensional elevation because 3D spheres that are rotation invariant as well as the nodes defining the piecewise linear center curve of the interconnects can be easily elevated to 4D and then projected back to 3D – essentially through

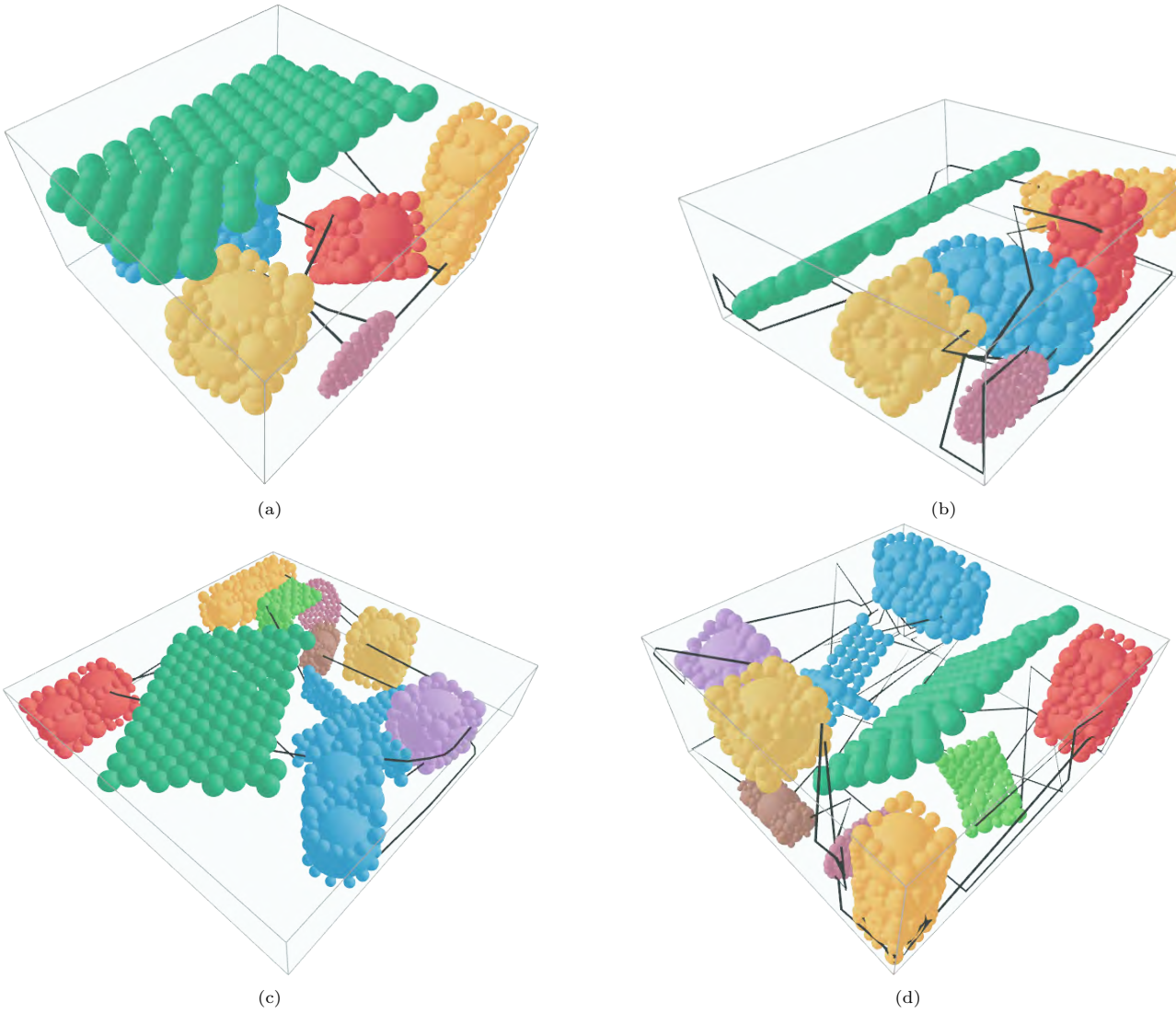


Figure 5: Side-by-side comparison showcasing the impact of including both volume and length terms versus only the volume term in the objective function, across two examples: (a) and (b) depict outcomes with both terms versus only the volume term for the six-component system, respectively; (c) and (d) illustrate results with both terms versus only the volume term for the ten-component system, respectively.

syntactic operations.

For example, let's assume an optimization problem for the system from Figure 7a in which the objective function comprises both volume and interconnect length minimization. The optimization carried out in the 3D space with collision constraints leads, as expected, to the solution depicted in Figure 7a. However, by employing the dimensional elevation proposed above, we are able to "insert" the interconnect through the toroidal shape illustrated in Figure 7a, without violating the collision constraints. We note that the two solutions are visually different and that through dimensional elevation the optimizer converges to a better solution.

There are, of course, many open questions raised by the proposed dimensional elevation that are outside the scope of this paper. However, we note that the proposed dimensional elevation technique, which takes full advantage of the computational properties of MDBD, expands the valid search space in which the optimization is carried out, which can potentially lead to significantly better optimal solutions that are not achievable through the state of the art gradient-based optimization approaches.

6. Conclusions

In this paper, we introduced a SPI2 design automation approach that utilizes the hierarchical Maximal Disjoint Ball Decomposition (MDBD) along with automatic differentiation to generalize the domain of the problems that can be handled to a domain that can include practical functional considerations. Our approach can handle complex interconnected systems with complex geometries, multi-physics interactions, and a variety of product development constraints, such as manufacturing and life-cycle constraints.

We verified the algorithm's ability to reach known global optima for three distinct benchmark problems and demonstrated the framework's capacity for optimizing systems with complex geometries. Additionally, we provided evidence of the benefits of integrating the lengths of the interconnects into the objective function to improve the performance of the system. Furthermore, we formulated and

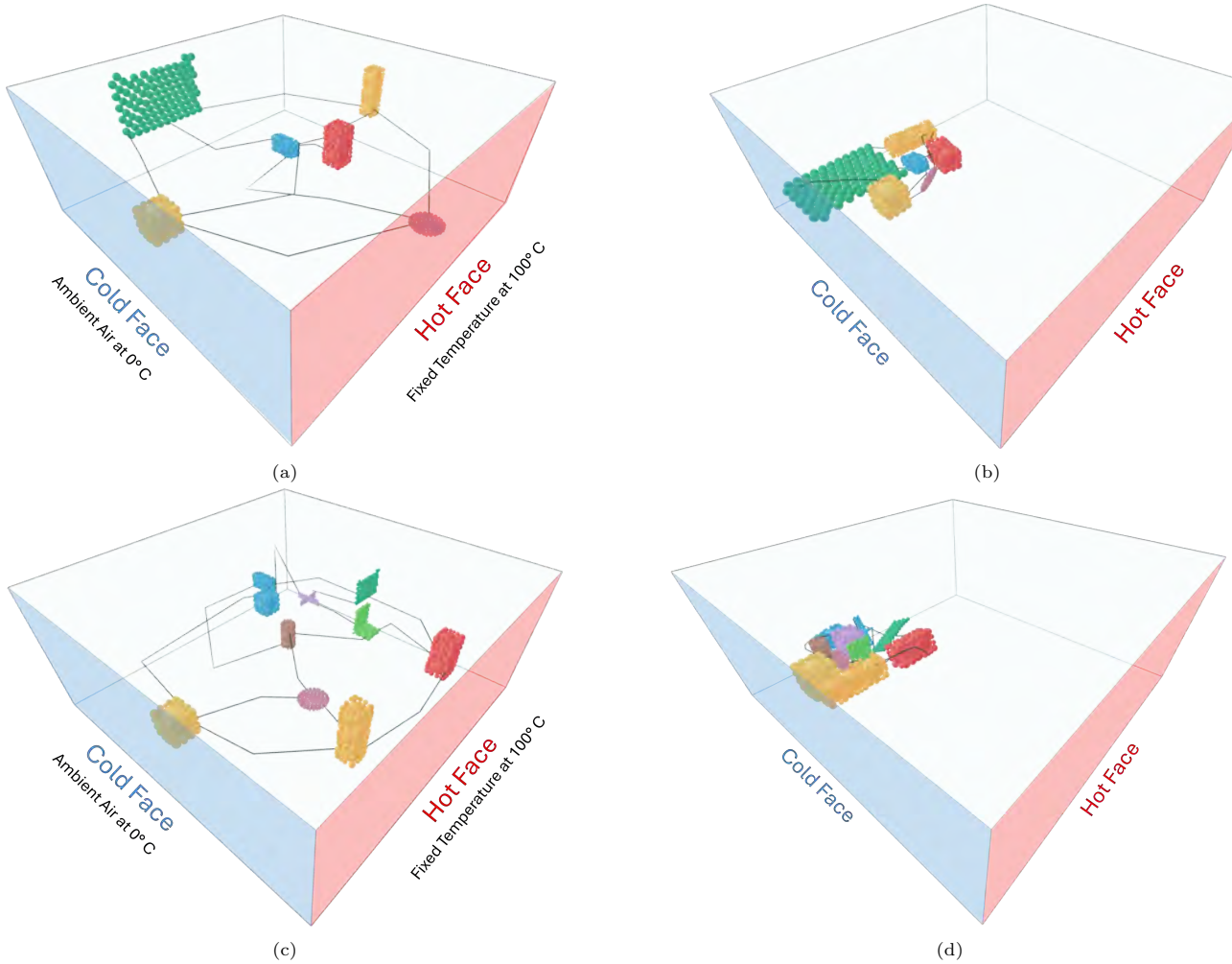


Figure 6: Thermal boundary conditions and optimized configurations for two setups: (a) and (b) showcase the initial and the final configuration for the system with six components and nine interconnects; (c) and (d) display the initial and the final configuration for the ten-component system with fifteen interconnects. The hot face maintains a fixed temperature of 100 °C, while the cold face experiences heat convection with ambient air at 0 °C. All other boundaries within the design are treated as thermally insulated.

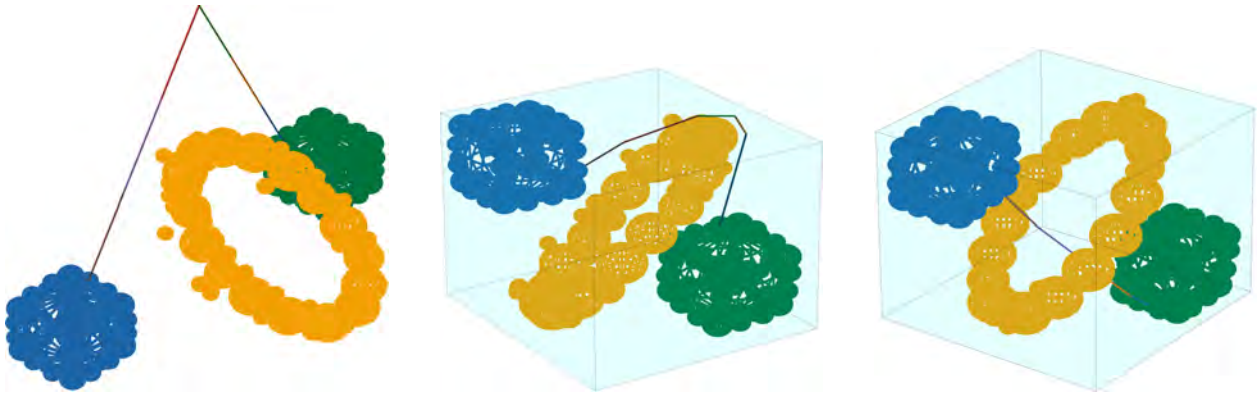


Figure 7: Side-by-side comparison showcasing original outcomes versus results achieved through dimension elevation. Figure (a) displays the initial configuration, (b) presents the original results without dimension elevation, and (c) illustrates the outcomes after applying dimension elevation.

validated the algorithm's abilities in handling physical interactions between the components of the system as well as those with the environment.

Our findings indicate that by employing MDBD and automatic differentiation, we can remove the existing constraints on geometric complexity, and provide the ability to handle realistic 3D shapes. This approach enables the incorporation of sophisticated terms in the objective function and

facilitates precise and efficient distance measurements essential for collision detection.

In this work, we assigned weights to the objective terms of the optimization problem to balance competing design goals, such as minimizing system volume and interconnect length. While the weights we used were integers for simplicity, the results of the optimization remain unaffected by introducing a weight normalization factor. Specifically, such a constant normalization factor multiplying the entire objective function would preserve the relative importance of each term and leads to the same optimization outcome. Moreover, the weights were chosen based on practical considerations for each case study, ensuring that the optimization process could effectively balance between the design objectives to achieve the most suitable solution for each scenario. While this approach provided effective solutions, future work may incorporate a Pareto front-based method to further refine the weight selection process and improve the balance between competing objectives.

Our investigations of time complexity detailed in section 5.2 indicate that the two most costly steps are the physical simulations and the automatic differentiation. Even though we used FEA augmented by uniform voxelization to simulate the physical internal and external interactions of our system, the proposed approach in principle supports a variety of other solution methods that would not require the computation of the voxel occupancy at every iteration. Nevertheless, exploring alternative solution methods is outside the scope of this paper.

In its current form, our implementation leverages GPU acceleration for evaluating objective functions and computing gradients. Notably, many components of the objective function exhibit a high degree of parallelizability. For instance, tasks such as calculating the length of interconnects and performing collision detection involve a series of Euclidean distance calculations that can be executed concurrently [68]. Additionally, various high-performance PDE solvers have been proposed, and see [69, 70] for some recent developments. Without a doubt, more efficient automatic differentiation approaches, distance computations, and more careful implementations that take full advantage of the available computational power are needed to scale up the number of components and interconnects by one or two orders of magnitude. Some specific suggestions were made in the respective sections of the paper and are not repeated here.

It is important to note that, even though the formulation presented in this paper allows the inclusion of complex geometries, objective functions, and constraints, further developments are needed in order to better support the industrial needs. For example, including the ability to co-design the components alongside the packing and routing would allow component geometries to adapt as the system configuration changes. In our current formulation, any geometric changes require a re-computation of the MDBD for the updated geometries, which introduces a significant computational burden. Enhancing the MDBD computational paradigm to address this component co-design represents a significant direction of future work.

Last but not least, in order to avoid the entanglements between interconnects/components during the optimization, we proposed a unique dimensional elevation strategy that can be implemented by using the computational properties of MDBD derived from the spherical decomposition. Specifically, by elevating the dimension of the space in which the optimization is carried out, and then by projecting the optimal configuration back onto the original space allows us to avoid the known and difficult numerical challenges produced by the entanglements produced by the colliding components and interconnects during the optimization.

In summary, this work marks a pioneering effort in leveraging MDBD and automatic differentiation for SPI2 system design. To the best of our knowledge, this is the first approach that is capable of handling design situations of practical complexity that involve complex geometries and complex functional objectives. These advancements lay the groundwork for future developments aimed at enhancing the scalability and efficiency of the algorithm.

Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under the "Multi-Disciplinary Optimization for Packaging (MDOP)" program, grant number FA8750-23-C-0501. Mohammad Behzadi and Horea Ilies would also like to acknowledge the support from the National Science Foundation grants CMMI-2232612, CMMI-2312175 and Peter Zaffetti was partially supported through a GAANN grant P200A210089 from the US Department of Education.

References

[1] S. R. Peddada, L. E. Zeidner, K. A. James, J. T. Allison, An introduction to 3D SPI2 (spatial packaging of interconnected systems with physics interactions) design problems: A review of related work, existing gaps, challenges, and opportunities, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 85390, American Society of Mechanical Engineers, 2021, p. V03BT03A034.

[2] A. Sakti, L. Zeidner, T. Hadzic, B. S. Rock, G. Quartarone, Constraint programming approach for spatial packaging problem, in: Integration of AI and OR Techniques in Constraint Programming: 13th International Conference, CPAIOR 2016, Banff, AB, Canada, May 29-June 1, 2016, Proceedings 13, Springer, 2016, pp. 319–328.

[3] S. R. Peddada, L. E. Zeidner, H. T. Ilies, K. A. James, J. T. Allison, Toward holistic design of spatial packaging of interconnected systems with physical interactions (SPI2), Journal of Mechanical Design 144 (12) (2022) 120801.

[4] Q. Liu, C. Wang, A discrete particle swarm optimization algorithm for rectilinear branch pipe routing, Assembly Automation 31 (4) (2011) 363–368.

[5] Q. Cui, V. Rong, D. Chen, W. Matusik, Dense, Interlocking-Free and Scalable Spectral Packing of Generic 3D Objects, ACM Trans. Graph 42 (4) (2023).

[6] D. E. Knuth, Postscript about NP-hard problems, ACM SIGACT News 6 (2) (1974) 15–16.

[7] S. R. Peddada, S. B. Rodriguez, K. A. James, J. T. Allison, Automated layout generation methods for 2D spatial packing, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 84010, American Society of Mechanical Engineers, 2020, p. V11BT11A013.

[8] A. Jessee, S. R. Peddada, D. J. Lohan, J. T. Allison, K. A. James, Simultaneous packing and routing optimization using geometric projection, Journal of Mechanical Design 142 (11) (2020) 111702.

[9] S. R. Peddada, K. A. James, J. T. Allison, A novel two-stage design framework for two-dimensional spatial packing of interconnected components, Journal of Mechanical Design 143 (3) (2021) 031706.

[10] C. Parrott, S. Peddada, J. T. Allison, K. James, Machine Learning Surrogates for Optimal 2D Spatial Packaging of Interconnected Systems with Physics Interactions (SPI2), in: AIAA AVIATION 2023 Forum, 2023, p. 4375.

[11] A. Bhattacharyya, S. R. Peddada, W. B. Bello, L. E. Zeidner, J. T. Allison, K. A. James, Simultaneous 3D component packing and routing optimization using geometric projection, in: AIAA SCITECH 2022 Forum, 2022, p. 2096.

[12] W. Bello, S. R. Peddada, A. Bhattacharyya, L. Zeidner, J. T. Allison, K. James, Multi-Physics 3D Component Placement and Routing Optimization Using Geometric Projection, Journal of Mechanical Design (2024) 1–40.

[13] J. Chen, H. T. Ilieş, Maximal Disjoint Ball Decompositions for shape modeling and analysis, Computer-Aided Design 126 (2020) 102850.

[14] H. Dong, P. Guarneri, G. Fadel, Bi-level approach to vehicle component layout with shape morphing (2011).

[15] A. Agafonov, V. Myasnikov, Vehicle routing algorithms based on a route reservation approach, in: Journal of Physics: Conference Series, Vol. 1096, IOP Publishing, 2018, p. 012029.

[16] S. R. Peddada, P. J. Tannous, A. G. Alleyne, J. T. Allison, Optimal sensor placement methods in active high power density electronic systems with experimental validation, Journal of Mechanical Design 142 (2) (2020) 023501.

[17] A. Panesar, D. Brackett, I. Ashcroft, R. Wildman, R. Hague, Design framework for multifunctional additive manufacturing: placement and routing of three-dimensional printed circuit volumes, Journal of Mechanical Design 137 (11) (2015) 111414.

[18] J. Tisdale, Z. Kim, J. K. Hedrick, Autonomous UAV path planning and estimation, IEEE Robotics & Automation Magazine 16 (2) (2009) 35–42.

[19] G. E. Jan, K. Y. Chang, I. Parberry, Optimal path planning for mobile robot navigation, IEEE/ASME Transactions on mechatronics 13 (4) (2008) 451–460.

[20] A. Cetera, A. Rabiee, S. Ghafoori, R. Abiri, Classification of emerging neural activity from planning to grasp execution using a novel eeg-based bci platform, arXiv preprint arXiv:2402.03493 (2024).

[21] P. Suchorab, D. Kowalski, Methods of routing and sizing of water supply networks, in: E3S Web of Conferences, Vol. 59, EDP Sciences, 2018, p. 00024.

[22] R. Araneo, S. Celozzi, C. Vergine, Eco-sustainable routing of power lines for the connection of renewable energy plants to the Italian high-voltage grid, International Journal of Energy and Environmental Engineering 6 (2015) 9–19.

[23] K. Abdel-Malek, H. Yeh, N. Maropis, Determining interference between pairs of solids defined constructively in computer animation, Engineering with Computers 14 (1998) 48–58.

[24] S. Yin, J. Cagan, P. Hodges, Layout optimization of shapeable components with Extended Pattern Search applied to transmission design, J. Mech. Des. 126 (1) (2004) 188–191.

[25] A. Rabiee, S. Ghafoori, A. Cetera, W. Besio, R. Abiri, A comparative study of conventional and tripolar eeg for high-performance reach-to-grasp bci systems, arXiv preprint arXiv:2402.09448 (2024).

[26] T. Xue, M. Wu, L. Lu, H. Wang, H. Dong, B. Chen, Learning Gradient Fields for Scalable and Generalizable Irregular Packing, in: SIGGRAPH Asia 2023 Conference Papers, 2023, pp. 1–11.

[27] S. Yang, S. Song, S. Chu, R. Song, J. Cheng, Y. Li, W. Zhang, Heuristics Integrated Deep Reinforcement Learning for Online 3D Bin Packing, IEEE Transactions on Automation Science and Engineering (2023).

[28] C. Lamas-Fernandez, J. A. Bennell, A. Martinez-Sykora, Voxel-based solution approaches to the three-dimensional irregular packing problem, Operations Research 71 (4) (2023) 1298–1317.

[29] J.-H. Pan, K.-H. Hui, X. Gao, S. Zhu, Y.-H. Liu, P.-A. Heng, C.-W. Fu, SDF-Pack: Towards Compact Bin Packing with Signed-Distance-Field Minimization, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2023, pp. 10612–10619.

[30] X. Wang, Z. Liu, J. Liu, Mobile robot path planning based on an improved a* algorithm, in: International Conference on Computer Graphics, Artificial Intelligence, and Data Processing (ICCAID 2022), Vol. 12604, SPIE, 2023, pp. 1093–1098.

[31] A. Sonny, S. R. Yeduri, L. R. Cenkeramaddi, Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance, Applied Soft Computing 147 (2023) 110773.

[32] Y. Zhou, X. Kong, K.-P. Lin, L. Liu, Novel task decomposed multi-agent twin delayed deep deterministic policy gradient algorithm for multi-UAV autonomous path planning, Knowledge-Based Systems 287 (2024) 111462.

[33] J. Huh, S. Jun, I. Y. Kim, Thermally Driven Multi-Objective Packing Optimization Using Acceleration Fields, Journal of Mechanical Design 146 (2024) 081703–1.

[34] S. Szykman, J. Cagan, P. Weisser, An integrated approach to optimal three dimensional layout and routing (1998).

[35] M. Scha¨ fer, T. Lengauer, Automated layout generation and wiring area estimation for 3D electronic modules, J. Mech. Des. 123 (3) (2001) 330–336.

[36] J. Cagan, K. Shimada, S. Yin, A survey of computational approaches to three-dimensional layout problems, Computer-Aided Design 34 (8) (2002) 597–611.

[37] M. Kaluschke, U. Zimmermann, M. Danzer, G. Zachmann, R. Weller, Massively-parallel proximity queries for point clouds, in: Virtual Reality Interactions and Physical Simulations (VRIPhys), Eurographics Association, Bremen, Germany, 2014.

[38] C. Lauterbach, Q. Mo, D. Manocha, gproximity: hierarchical gpu-based operations for collision and distance queries, in: Computer Graphics Forum, Vol. 29, Wiley Online Library, 2010, pp. 419–428.

[39] M. Behandish, H. T. Ilieş, Haptic Assembly Using Skeletal Densities and Fourier Transforms1, Journal of Computing and Information Science in Engineering 16 (2) (2016) 021002.

[40] M. Behandish, H. T. Ilieş, Analytic methods for geometric modeling via spherical decomposition, Computer-Aided Design 70 (2016) 100–115, sPM 2015.

[41] W. S. Moses, S. H. K. Narayanan, L. Paehler, V. Churavy, M. Schanen, J. Hückelheim, J. Doerfert, P. Hovland, Scalable automatic differentiation of multiple parallel paradigms through compiler augmentation, in: SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, 2022, pp. 1–18.

[42] J. Hückelheim, L. Hascoët, Source-to-source automatic differentiation of openmp parallel loops, ACM Trans. Math. Softw. 48 (1) (feb 2022).

[43] I. Ifrim, V. Vassilev, D. J. Lange, Gpu accelerated automatic differentiation with clad, Journal of Physics: Conference Series 2438 (1) (2023) 012043.

[44] M. Zubair, D. Ranjan, A. Walden, G. Nastac, E. Nielsen, B. Diskin, M. Paterno, S. Jung, J. H. Davis, Efficient gpu implementation of automatic differentiation for computational fluid dynamics, in: 2023 IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC), 2023, pp. 377–386.

[45] S. Fialko, Parallel finite element solver for multi-core computers with shared memory, Computers & Mathematics with Applications 94 (2021) 1–14.

[46] G. E. Karniadakis, S. A. Orszag, Parallel spectral computations of complex engineering flows, Supercomputing in Engineering Analysis (2020) 289–324.

[47] A. Requicha, Representations for Rigid Solids: Theory, Methods, and Systems, ACM Computing Surveys 12 (4) (1980).

[48] R. Goldman, Understanding quaternions, Graphical models 73 (2) (2011) 21–49.

[49] W. R. Hamilton, Elements of quaternions, Longmans, Green, & Company, 1866.

[50] J. Dooley, J. McCarthy, Spatial rigid body dynamics using dual quaternion components, in: Proceedings. 1991 IEEE International Conference on Robotics and Automation, 1991, pp. 90–95 vol.1.

[51] W. K. Clifford, Mathematical Papers by William Kingdon Clifford: Edited by Robert Tucker, with an introduction by HJ Stephen Smith, Macmillan and Company, 1882.

[52] M. A. O'Connor, V. Srinivasan, A. Jones, Connected Lie and symmetry subgroups of the rigid motions: Foundations and classification, IBM TJ Watson Research Center, 1996.

[53] W. M. Boothby, An introduction to differentiable manifolds and Riemannian geometry, Academic press, 1986.

[54] S. Tiwari, G. Fadel, P. Fenyes, A Fast and Efficient Compact Packing Algorithm for SAE and ISO Luggage Packing Problems, Journal of Computing and Information Science in Engineering 10 (2) (2010) 021010.
[55] H. Zhao, Z. Pan, Y. Yu, K. Xu, Learning physically realizable skills for online packing of general 3D shapes, ACM Transactions on Graphics 42 (5) (2023) 1–21.
[56] S. Tiwari, G. Fadel, P. Fenyes, A Fast and Efficient Compact Packing Algorithm for SAE and ISO Luggage Packing Problems, Journal of Computing and Information Science in Engineering 10 (2) (2010) 021010.
[57] M. Behandish, H. T. Ilieş, Analytic methods for geometric modeling via spherical decomposition, Computer-Aided Design 70 (2016) 100–115.
[58] M. Lysenko, Fourier collision detection, The International Journal of Robotics Research 32 (4) (2013) 483–503.
[59] L. Kavraki, Computation of configuration-space obstacles using the fast Fourier transform, IEEE Transactions on Robotics and Automation 11 (3) (1995) 408–413.
[60] D. L. James, D. K. Pai, BD-Tree: Output-sensitive collision detection for reduced deformable models, in: ACM SIGGRAPH 2004 Papers, 2004, pp. 393–398.
[61] R. Weller, Inner Sphere Trees, Springer International Publishing, Heidelberg, 2013, pp. 113–144.
[62] A. Barbier, E. Galin, Fast distance computation between a point and cylinders, cones, line-swept spheres and cone-spheres, Journal of Graphics tools 9 (2) (2004) 11–19.
[63] S. Agarwal, R. A. Srivatsan, S. Bandyopadhyay, Analytical determination of the proximity of two right-circular cylinders in space, Journal of Mechanisms and Robotics 8 (4) (2016) 041010.
[64] P. Zsombor-Murray, Intrusion, proximity and stationary distance, in: Computational Kinematics: Proceedings of the 7th International Workshop on Computational Kinematics that was held at Futuroscope-Poitiers, France, in May 2017, Springer, 2018, pp. 475–482.
[65] S. Kirkup, The boundary element method in acoustics: A survey, Applied Sciences 9 (8) (2019) 1642.
[66] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[67] S. Kambampati, A. Taber, G. Kumar, H. A. Kim, A CAD-aware plug-and-play topology optimization framework using moments, Structural and Multidisciplinary Optimization 66 (3) (2023) 63.
[68] D. Man, K. Uda, H. Ueyama, Y. Ito, K. Nakano, Implementations of parallel computation of euclidean distance map in multicore processors and gpus, in: 2010 First International Conference on Networking and Computing, IEEE, 2010, pp. 120–127.
[69] J. Andrej, N. Atallah, J.-P. Bäcker, J. Camier, D. Copeland, V. Dobrev, Y. Dudouit, T. Duswald, B. Keith, D. Kim, T. Kolev, B. Lazarov, K. Mittal, W. Pazner, S. Petrides, S. Shiraiwa, M. Stowell, V. Tomov, High-performance finite elements with mfem (2024). `arXiv:2402.15940`.
[70] P. Fischer, M. Min, T. Rathnayake, S. Dutta, T. Kolev, V. Dobrev, J.-S. Camier, M. Kronbichler, T. Warburton, K. Świrydowicz, J. Brown, Scalability of high-performance pde solvers, The International Journal of High Performance Computing Applications 34 (5) (2020) 562–586.

Appendix

6.1. MDBD Computation

Given a component r-set, O, the signed distance function (SDF) is generated for the encompassing domain and then searched over to select the maximal radii corresponding to the MDBD representation spheres. For an efficient computation of the signed distance function, five parallelizable steps are performed in sequence. In our work, the component r-sets are defined using triangular meshes. Computational speed up is derived from heavy use of hardware raytracing and using triangles' normals to localize and reduce the number of sample point to triangle distance calculations. Steps 1-3 use a ray tracing acceleration structure built from the triangles of the defining meshes.

1. For each sample point of the encompassing domain of O, perform an inclusion test by ray casting a vector and counting the number of intersections. Denote I as the set of all interior sample points of O.
2. For each mesh vertex v, average the interior-facing normals of its adjacent faces and ray trace the result to find the point, v_o, colliding with O opposite v.
3. For each triangle t, define a bounding box b_t as the minimum and maximum x, y, and z coordinates from the set $\{v_1, v_2, v_3, v_{o1}, v_{o2}, v_{o3}\}$, where vertices $v_1, v_2, v_3 \in t$ and v_{o1}, v_{o2}, and v_{o3} are the respective collision points. Denote B as the set of all bounding boxes.
4. Using an acceleration structure built from the bounding boxes of B, for each interior sample point $i \in I$, ray trace a vector in any direction but with zero length. This produces a set of bounding boxes, B_i, which intersect i.
5. For each point i and associated bounding boxes $b \in B_i$, compute the minimum distance between i and t_b, where t_b is the triangle which produced bounding box b. Let M_i be the set of minimum distances between point i and triangles t_b. Assign the minimum value of M_i as the SDF value for i.

Steps 4 and 5 MDBD generation is completed by iteratively:

1. Searching over the SDF for the maximally interior distance, r, and location, c.
2. Placing a sphere s at c with radius r.
3. Updating SDF values due to inserting s

A key observation is that for an inserted sphere s changed regions in the SDF only extend to $2 * r$ from the sphere center, c. Searching over the signed distance function can be performed quicker by caching maximal interior SDF value and locations of unchanged regions of the SDF.

Algorithms 1-3 provide high-level pseudo-code for generating the MDBD using the signed distance function regardless of the geometric representation of the r-set.

Algorithm 1 Maximal Disjoint Ball Generation

function MDBDGeneration(r, n_{sph})
  Input
    r The r-set which will be decomposed
    n_{sph} The number of spheres to place in the decomposition
    n_{grid} The number of grid cells in each coordinate direction
    s The grid cell size
  Output
    sph An array of spheres. Each sphere is represented by 4 values: x, y, z, radius where x, y, z correspond to the sphere's center location.
  $sdf \leftarrow$ SDFCompute(r, n_{grid}, s)
  $i \leftarrow 0$
  $sph \leftarrow Array\left[n_{sph}\right]$ ▷ Initialize the array of spheres where each element will hold 4 values: x, y, z, radius
  while $i < n_{sphere}$ **do**
    $sph\left[i\right], sphRadius, sphIdx \leftarrow$ FindSphere(sdf)
    UpdateSDF$(sdf, sphRadius, sphIdx)$
  end while
  return spheres

Algorithm 2 Signed Distance Function Computation

function SDFCompute(b, n, s)
  Input
    b The boundary of the r-set for which the SDF will be generated
    n The number of SDF sample points in each coordinate direction. If $n = 128$ then there will be 128^3 sample points total.
    s The SDF grid cell size
  Output
    sdf An array of length n^3 containing the distance values at each sample point.
  $x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max} \leftarrow$ BoundingBox(b)
  $cell_{center} \leftarrow s/2$
  $sdf \leftarrow Array\left[n^3\right]$ ▷ Initialize sdf as an array where each element contains 4 empty fields: x, y, z, and sdf value
  for $i \in n^3$ **do** ▷ Generate and evaluate discrete sample points
    $idx_z \leftarrow i/(n * n)$
    $idx_y \leftarrow (i - idx_z * n * n)/x$
    $idx_x \leftarrow i - (idx_z * n * n) - idx_y * n$
    $x \leftarrow x_{min} + idx_x * s + cell_{center}$
    $y \leftarrow y_{min} + idx_y * s + cell_{center}$
    $z \leftarrow z_{min} + idx_z * s + cell_{center}$
    $interior \leftarrow$ PointInclusionRaycast(b, x, y, z)
    $minDistance \leftarrow$ MinimumDistanceToBoundary(b, x, y, z)
    $sdf\left[i\right] \leftarrow [x, y, z, interior * minDistance]$
  end for

Algorithm 3 Algorithm for determining whether a point is in the interior of a boundary set using raycasting.

function PointInclusionRaycast(b, x, y, z)
  Input
    b The boundary of the r-set
    x The x-value of the point to evaluate
    y The y-value of the point to evaluate
    z The z-value of the point to evaluate
  Output
    $sign$ If the point is inside or on the boundary then $sign$ will be +1, otherwise it will be -1.
  $sign \leftarrow -1$
  $rayEndpoint \leftarrow [0, 0, \infty]$ ▷ Set the endpoint of the ray to be infinitely far away
  $intersectionVector \leftarrow rayEndpoint - [x, y, z]$ ▷ Subtract the two endpoints to create the intersection vector.
  for $e \in b$ **do**
    if Intersects$(e, intersectionVector)$ **then**
      $sign \leftarrow -1 * sign$
    end if
  end for
  return $sign$