DaMarcus Patterson, Dr. Julie Chalfant, Dr. Michael Triantafyllou
Sea Grant Design Laboratory, Massachusetts Institute of Technology

Automation of Component Placement in Early-Stage Ship Design

ABSTRACT

Modern ships, and naval ships in particular, feature a wide range of components. Devices for power generation, navigation, and defense are all interconnected in a complex, integrated system. Software is heavily used in the design phase when placing internal components; however, manually placing components can be time consuming, especially when designers are considering things like weight distribution and accessibility to components for maintenance. In this paper we present a methodology for automatically arranging components in a ship design in a manner that eliminates overlaps, provides spacing between components, and minimizes connection length. The algorithm is described, and several examples of varying complexity are presented.

INTRODUCTION

Modern weapon and sensor systems are increasingly power-intensive, placing great demands on a ship's electrical power generation and distribution systems; further, the losses incurred place great demands on a ship's thermal management systems. In addition, the increasingly integrated nature of a ship's control structure in which the systems traditionally known as support systems are now an integral part of the fighting success of a ship place even greater importance on the role of these systems. The increased importance and increased size of the support systems make it imperative that the systems be modeled earlier in the ship design process than was previously accomplished, that the models throughout the design cycle are more detailed, and that the models are able to simulate the performance of systems at a level of detail appropriate to the stage of design. These demands are precipitating a change in the design tools used.

In recent years, the increased capacity and capability of computational resources has fundamentally changed the ship design process from a spiral iteration of a few point designs to the automated exploration of a very broad design space reaching thousands or even hundreds of thousands of designs. Methodologies such as set-based design [1], technology identification, evaluation, and selection [2], and surrogate modeling [3] are becoming the new face of ship design. New design tools must function in such an environment.

This material is based upon research supported by, or in part by, the U.S. Office of Naval Research (ONR) under award number ONR N00014-16-1-2945 Incorporating Distributed Systems in Early-Stage Set-Based Design of Navy Ships; ONR N00014-16-1-2956 Electric Ship Research and Development Consortium; and by the National Oceanic and Atmospheric Administration (NOAA) under Grant Number NA14OAR4170077 - MIT Sea Grant College Program.

Distribution A. Approved for public release, distribution is unlimited. DCN # 43-6492-20.

In response to these needs, the Electric Ship Research and Development Consortium (ESRDC) under a grant from the Office of Naval Research and in consultation with the Naval Surface Warfare Center Carderock Division has created a software environment called S3D (Smart Ship Systems Design) [4]–[6]. S3D enables the creation, simulation and analysis of shipboard distribution systems in the electrical, thermal and mechanical energy domains. It is a component-centric, object-oriented software in which each component has a mathematical representation in each applicable domain. Further description of S3D is provided in the background section.

The initial effort in creating S3D concentrated on the ability to reliably construct and analyze a multi-disciplinary set of systems for an individual ship design, with the user fully integrated into each step of the process. The vision for S3D has always contained the concept of expanding the capability to include an automated, user-directed process in which multiple systems can be created, adapted to different hullforms, simulated, and analyzed. The approach to this concept has come to be known as Templating. In short, Templating involves combining pre-designed sections of systems into a full ship system architecture, tracing the resulting system to determine required capacity and thus dimensions of each component, then arranging the components in three-dimensional space. An overview of the Templating process is presented in the background section.

The final step in the Templating process is the physical arrangement of components in three-dimensional space. This paper addresses a methodology of arranging components using force-directed graph drawing, which is a method for achieving visually pleasing arrangements of graphs in two or three dimensions. By applying repulsive forces between all vertices of a graph and attractive forces between vertices connected by edges, then moving the vertices to minimize energy, the graphs tend to resolve themselves such that edges are of fairly equal length and edge crossings are minimized. Numerous algorithms exist; a survey is presented in [7]. We apply this concept

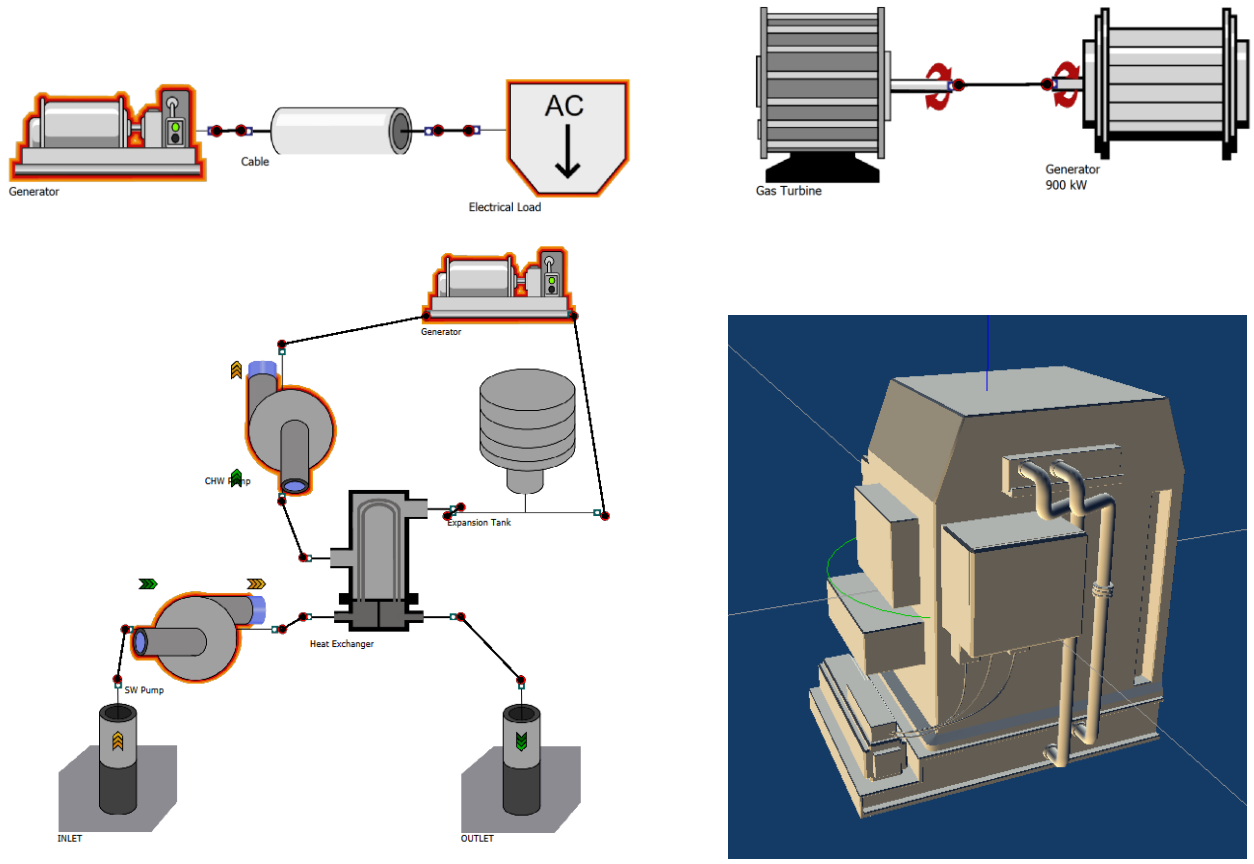

Figure 1: Views of a generator in four different disciplines: electrical (top left), mechanical (top right), thermal (bottom left) and CAD (bottom right) [8].

to the placement of components in a compartment to achieve an arrangement in which components (represented as graph vertices) are spaced apart while the length of connections such as cables or pipes (represented as graph edges) are minimized.

BACKGROUND

Smart Ship System Design (S3D)

S3D is a software tool used to design and analyze ship systems. It contains a rich equipment library of typical shipboard components that can be selected, parameterized, placed in a ship design, and connected to one another to form systems. Each component has a mathematical and visual representation in multiple domains including electrical, piping and mechanical.

A system design in S3D consists of components arranged and connected in a one-line diagram for a specific discipline. If a component acts in multiple domains, the same component will appear in each relevant domain, with the associated mathematical model and properties for that domain. For example, a generator will have a model in the electrical domain to represent the electrical power generated including properties such as voltage and frequency. The same component will have a representation in the mechanical domain to model the input mechanical power required from an engine, including such properties as rotational speed and torque. If it is a water-cooled generator, it will also have a representation in the piping domain, specifying properties such as cooling water temperature and specific heat. Physical properties such as dimensions and weight are also tabulated. A CAD representation is available for three-dimensional placement within a ship hull; the locations of equipment can then be used to determine the length of distribution components such as cabling, piping or shafts. Views of a generator in the electrical, mechanical and thermal disciplines of S3D can be seen in Figure 1.

The S3D software co-simulates the design in all disciplines to capture the effects of one domain on another, using a power-flow-level simulation. Results include such data as voltages, currents, flow rates and temperatures throughout. Operational parameters such as valve position, power settings or breaker position can be varied to test the design against different operational missions. Metrics such as power required, fuel usage, losses, weight and volume can be parsed to compare and evaluate designs against one another.

Templating

The MIT Sea Grant Design Laboratory is conducting a body of work with the goal of achieving semi-automated design of ship systems, in which the systems are assembled automatically under the guidance of the engineer or designer, using a Templating process. Templates are pre-designed systems or sections of systems, and can be created using the S3D software. At the end of the Templating process, the templates have been assembled into fully connected, fully functioning ship systems with components properly parameterized, properly dimensioned, and placed in three-dimensional space in a ship hullform ready for system simulation and analysis.

The process for creating a fully functional ship system from templates requires several steps, described below.

The first step is to assemble the templates into a logically connected system by copying relevant templates into the ship design and connecting them appropriately to one another. This places the proper components in the proper configuration, yielding a logically appropriate one-line diagram with components placed in an approximate geographic position within the ship. At first blush this seems quite simple, but the complexity comes in the details of such things as deep copy of the templates and all subordinate components, connections, properties, views, and tools; connection of templates in the proper logical order without cross-connections or loose ends; and association of each item with the appropriate, non-duplicated systems and common groupings.

The second step is to determine the capacity required of each component. Since the Templating capability facilitates the creation of ship systems from an assembly of parts or system sub-sections, it is not possible to determine the required capacity of each element of a system until the system is fully assembled and placed in three-dimensional space. Variations in the sources and loads that make up the system and variations in the topology into which they are assembled will affect the capacity required of each component. Further, the location of components will affect the length of distribution components such as cables, pipes or shafts, which will extend or contract in length as components are placed farther apart or closer together. This in turn will affect the losses in the distribution components, as they are generally calculated on a per-unit-length basis.

To determine the capacity of each component, it is necessary to determine the maximum amount of power that can flow through each component given any possible alignment of the system. An algorithm to determine this power requirement using graph theory is described in [8]. This algorithm is applied to a fully connected system, and the rated power of each component is set to the value determined by the algorithm.

Once the capacity of each component is determined, the component is dimensioned appropriately for the capacity required. Some components in S3D contain automated sizing algorithms; the long-term goal is for all component types to provide this functionality.

The third step is to make final placement of the components in three-dimensional space, which is the topic of the remainder of this paper.

PROBLEM DESCRIPTION

Herein, a *component* is a piece of equipment such as a diesel engine, a pump, an electrical converter, or a valve.

A *zone* is a defined region of a ship. While we use the term zone in this paper, this functionality represents any region of a ship such as a compartment, a zone or an entire hull.

At the start of the component arrangement process, components have been logically connected into systems, but have not been precisely positioned in three-dimensional space. They may have no positioning information, they may all be lumped at the origin of the ship or at the centroid of the zone, or they may be roughly located close to their final position; however, components may overlap one another or intersect with the ship structure. Components may be assigned to zones, although they may or may not be positioned within the zone at the beginning of the arrangement process. Some components may be fixed in place.

The goal of this algorithm is to arrange all components within the designated zone. The components must be spread throughout the zone, separated so they do not overlap, and placed in a manner that considers dimensions, interactions and connections with other components. A few use cases of this functionality include:

1) Taking many components that were placed at the centroid of a zone and arranging them throughout the zone.
2) Arranging several components around other components with fixed locations. For example, arranging the tees, valves and piping of a cooling system around a few large fixed components that require cooling.
3) Resolving minor overlaps among components that have been placed in approximately the proper location, through small movements of the components.

The problem statement is summarized as follows.

Given a set of components including their dimensions, positions, and zone assignment; a list of connections between components; and indication of which, if any, components are fixed in three-dimensional space; arrange the components within the indicated zone and provide as an output the refined locations of the components.

Components must be placed such that the following constraints are met:

First, components that are connected to each other are placed close together. This reduces complexity of connections as well as the amount of material used for distribution components such as cabling and piping, consequently reducing the ship's weight.

Second, components are placed such that no component overlaps another component and there is as much space as possible between components. Space between components allows access for operation, maintenance and repair.

Third, components remain within the assigned zone and do not intersect the boundaries of that zone, such as the ship hull or the bulkheads or decks.

APPROACH

We implement force-directed graph-drawing theory and gradient descent methodologies for component placement. The resulting arrangements are conflict-free while respecting connections and interactions of components and the boundaries of the zone.

In simple terms, the force-directed graph-drawing algorithm assigns repelling forces to all components and attracting forces just to connected components, with a goal of equal separation between components.

The system of components is depicted using an undirected graph in which the vertices represent components and the edges represent physical connections between components such as cables, piping or shafts. It is possible that some components may be connected to no other components, or that there may be islanded sections of the system in which a few components are connected to one another but not to other portions of the set of components.

A. Equations

The attractive and repulsive forces are functions of the distance between components, r_{ij}, which we define as the distance between the edges of the bounding boxes of components *i* and *j* as shown in Figure 2. Note that the distance between intersecting components is defined to be zero; thus, r_{ij} is never less than zero.

1) Forces: The repulsive force, F_R, is modeled after electrostatic forces using Coulomb's law, and is thus inversely proportional to the square of the distance between components such that

$$F_R(i,j) = \frac{k_R}{r_{ij}^2} \tag{1}$$

where k_R is the weighting of the repulsive force. For our application, we assume the point charges of all components are equal.

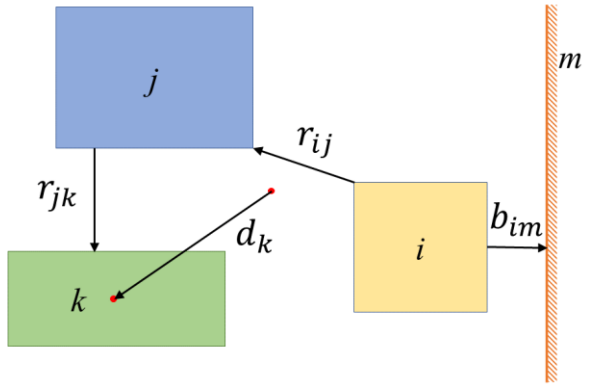

Figure 2: Various distance measurements

The attractive force, F_A, is modeled on the force of a spring as described by Hooke's law, and is thus proportional to r_{ij} such that

$$F_A(i,j) = -k_A r_{ij} \tag{2}$$

where k_A is the weighting of the attractive force.

At each iteration, computing the pairwise forces on each vertex allows the 3D locations of the vertices to be adjusted according to the sum of the forces.

To improve speed, we converted the iterative pairwise comparison to a gradient descent formulation. We compute an overall loss function for the entire system of equations and minimize that loss function with respect to the vertex positions for every vertex. This representation requires the anti-derivative of each force equation. Thus, repulsive force in (1) is replaced with repulsive energy

$$U_R(i,j) = k_R \frac{1}{r_{ij}},$$

and the attractive force in (2) is replaced with attractive energy

$$U_A(i,j) = \frac{1}{2} k_A r_{ij}^2.$$

So that larger components are placed lower, we added a gravitational energy to each component, U_G, such that

$$U_G(i) = z_i v_i \tag{3}$$

where z_i is the vertical component of the distance of the *i*th component's centroid from the center of the zone as shown in Figure 2, and v_i is the volume of the *i*th component's bounding box minus the mean of all components' bounding boxes.

The k-factors k_R and k_A are constant and the same for each component; adjusting these k-factors changes the relative strength of the repulsive and attractive forces.

2) Bounds: When placing the components in the ship, one goal is that the components do not intersect the sides of the zones; we do not want intersections between equipment and the bulkheads, decks or hullform. We also

want components to be spread out within the zone. This is accomplished by placing attractive and repulsive forces on the boundaries of the zone,

$$U_B(i,k) = k_{BR}\frac{1}{b_{ik}} + k_{BA}b_{ik},$$

where b_{ik} is the distance from bounding box of component i to boundary k as shown in Figure 2, k_{BR} is the weighting of the repulsive component of the boundary force, and k_{BA} is the weighting of the attractive component of the boundary force. The bounds are planar and are currently axis-aligned.

If this constraint is strictly applied during the early time steps, the problem can become too constrained and the components will not have space to reposition themselves. Therefore, we apply the zone constraints in a gradual manner. At first, no boundary constraints are applied and components are allowed to reposition freely. After this initial phase, boundaries are imposed at a large distance and gradually brought into position. The initial (large) distance is twice the greater of the distance from the centroid of the zone to the boundary or to the farthest component in each of six directions ($\pm x, \pm y, \pm z$). The boundaries are then moved linearly into their final position over all remaining iterations. If at any time step a component is located outside the boundary, its position is forced to fall within the boundary.

3) Loss Function: The loss function includes the sum of the attractive and repulsive energy for every pair of vertices, plus the gravity and bound energy for each vertex. In total, the loss function is

$$L = \frac{2}{n(n-1)}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\left[U_A(i,j) + U_R(i,j)\right] + \sum_{i=1}^{n}U_G(i) + \sum_{i=1}^{n}\sum_{k=1}^{m}U_B(i,k)\right] \quad (4)$$

where n is the number of vertices in the graph, m is the number of boundaries, and $i \neq j$.

Input Data

When loading the components, we store a $3\times n$ position matrix V with each component's position, a matrix F where the entry F_{ii} is 0 if the component is fixed and otherwise 1, and a symmetric adjacency matrix A indicating whether components are connected or not.

Code Outline

At each of T iterations:

- An $n \times n$ distance matrix R is calculated with entry r_{ij} being the closest distance between component i and component j's bounding boxes. If the components intersect, the distance is 0.
- The loss function shown in (4) is calculated. The repulsive term involves every component and can be computed as half the sum of the elements of $1/R$. Attractive energy only exists between connected components, so the attractive term is half the sum of the elements of the element-wise multiplication of $\frac{1}{2}R^2$ with the adjacency matrix A. If the current iteration is in the initial unbounded phase, bound energy U_B is not included.
- The norm of the matrix of gradients is limited so that it does not exceed a constant value, C_{lim}. This restricts the movement of components at each step and prevents wild oscillating swings of component position in the early steps of the algorithm. For our application, we use $C_{lim} = 1$.
- The gradients are multiplied by a matrix F_{ij} that equals 0 in the diagonal position ii if the ith component is fixed. This eliminates any movement of fixed components.

$$G = GF$$

- The gradients are added to the position matrix and the bounds contract. If the bounds are already in their final position, they do not change.

Local Minima

One of the recognized problems with a force-directed graph algorithm is that local minima may cause an undesirable solution. For example, several components in a perfect line may have repelling and attracting forces that are exactly balanced, so there is no force in a direction that causes the components to move to a more desirable arrangement. A small perturbation of the position will change the direction of the forces sufficiently to escape the local minimum and arrive at a potentially better overall solution. For this purpose, we interject a small random perturbation in the position of all components once every 5 iterations up to iteration 50, then once every 10 iterations through iteration 100.

Metrics

When investigating the examples, we present several metrics for review. The sum of the distances between components provides an indication of total length of connecting materials such as cabling or piping. The sum of the repulsive energy, the sum of the attractive energy and the value of the loss function are indicators of the balance of forces on components and the spacing of the components. The number of collisions at the end of the process provides a check that the problem is resolved. We also present the number of components, number of iterations, and run time.

EXAMPLES

We begin with some simple examples to demonstrate specific functionality of the code, then present a medium-sized example with multiple components and systems, and finish with a large example representative of a more complex ship system design.

Simple Examples

For each simple example, 150 iterations of gradient descent were run with occasional random perturbations of the components' positions. Allowing additional iterations would result in further movement of components. All of these examples consist only of components and connections; no zonal boundaries are invoked.

The first example is composed of two unconnected components as shown in Figure 3. The two components rearrange around the origin. They are spaced apart due to the repelling force, but the gravity force keeps them from being expelled from the zone.

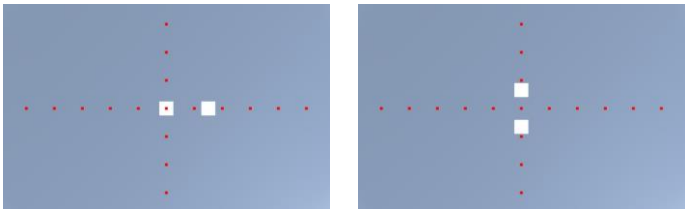

Figure 3: Simple example: two unconnected components

The second example consists of four components placed in a line as shown in Figure 4. Each component is connected to every other component. The components rearrange themselves in a tetrahedral shape centered on the origin. Note that in the starting position, the forces are perfectly aligned so no movement will occur until the random perturbation pushes the components slightly out of alignment. Once out of alignment, the applied forces move the components into position.

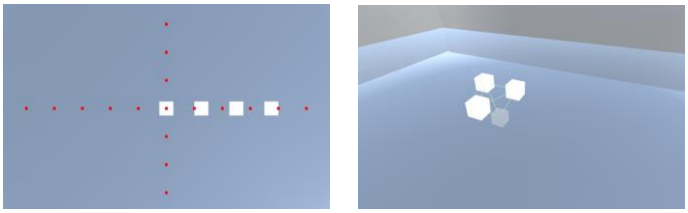

Figure 4: Simple example: connected components in a line

Next, we investigate four components placed in a line as shown in Figure 5. The two blue components are fixed. The left-hand blue component is connected to the unfixed white component on the far left, which is connected to the unfixed white component on the far right, which is connected to the fixed component on the right. In the final image on the right-hand side of Figure 5, the two fixed components remain fixed and the white components rearrange around them.

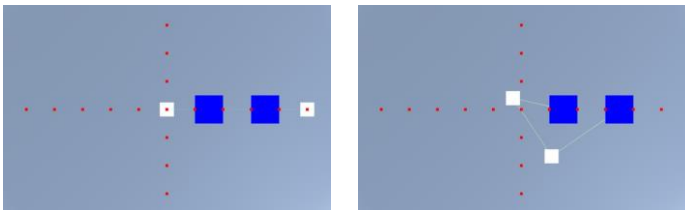

Figure 5: Simple example: connected components in a line with two fixed components

The final simple example consists of two pairs of connected components, with all components co-located at the origin as shown in Figure 6. The two pairs rearrange such that each pair is centered on the origin, but separated from the other pair.

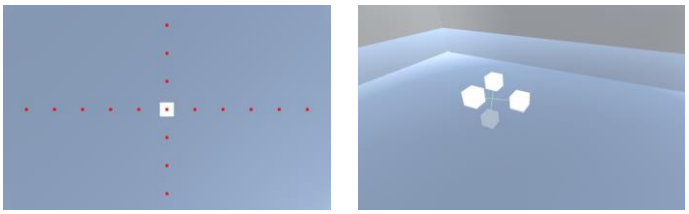

Figure 6: Simple example: co-located, connected components

Medium-Sized Example

Our next example is a set of sixteen components placed in a single compartment. The electrical and piping one-line diagrams showing connections between components appear in Figure 7, screenshot from the S3D software. This example was generated purely for testing the component location algorithm and is not intended to represent an actual ship system.

Note that this example consists of two separate sets of components with no connections between the sets. The generator, chiller and load in the top electrical one-line diagram in Figure 7 all have both electrical and piping representations, so they also appear in the piping diagram. The switchboard from the top electrical diagram and the generator, rectifier and dc load in the bottom diagram of Figure 7 are all air-cooled, so they do not appear in the piping diagram. Therefore, the three components in the lower portion of the electrical one-line diagram are islanded and do not have any physical connection to the other components in the system.

Prior to running this algorithm, the components had been logically connected in the one-line diagrams, but they had not been given a physical location in three-dimensional space; therefore, at the beginning of the application of the algorithm, all components were co-located with their centroids placed at the center of the zone. One of the components was designated as fixed, and all components were assigned to the same zone.

The algorithm was run on this system for 1000 iterations. For a system of this size, total run time was 72.3

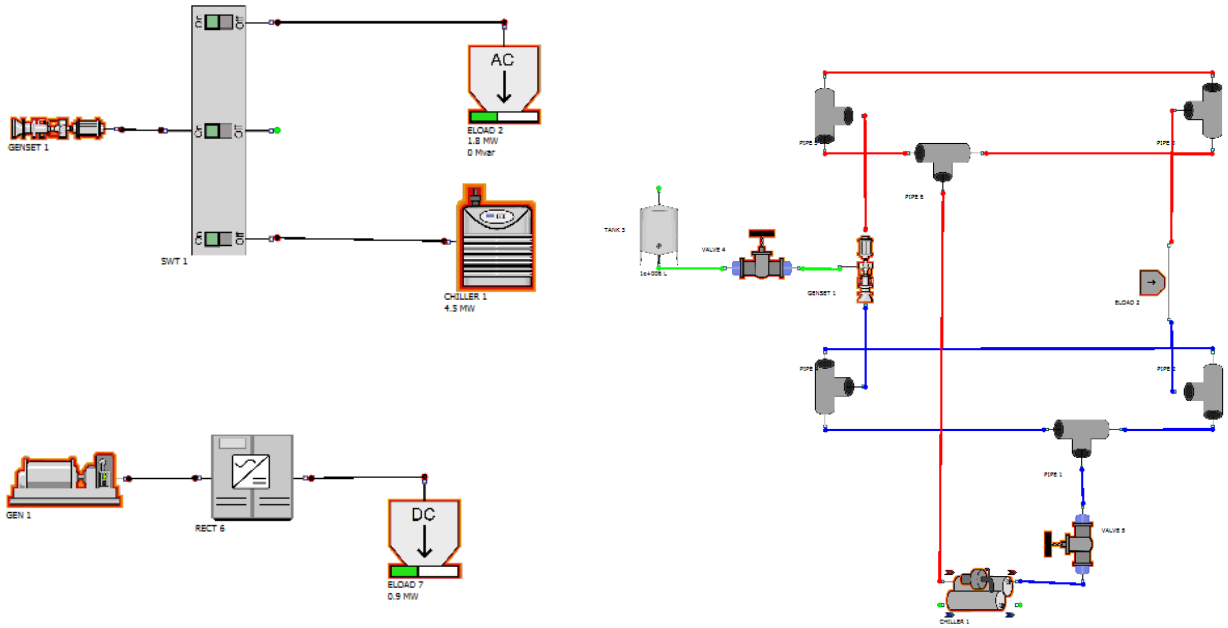

Figure 7: Medium example: electrical one-line diagram (left) and piping one-line diagram (right)

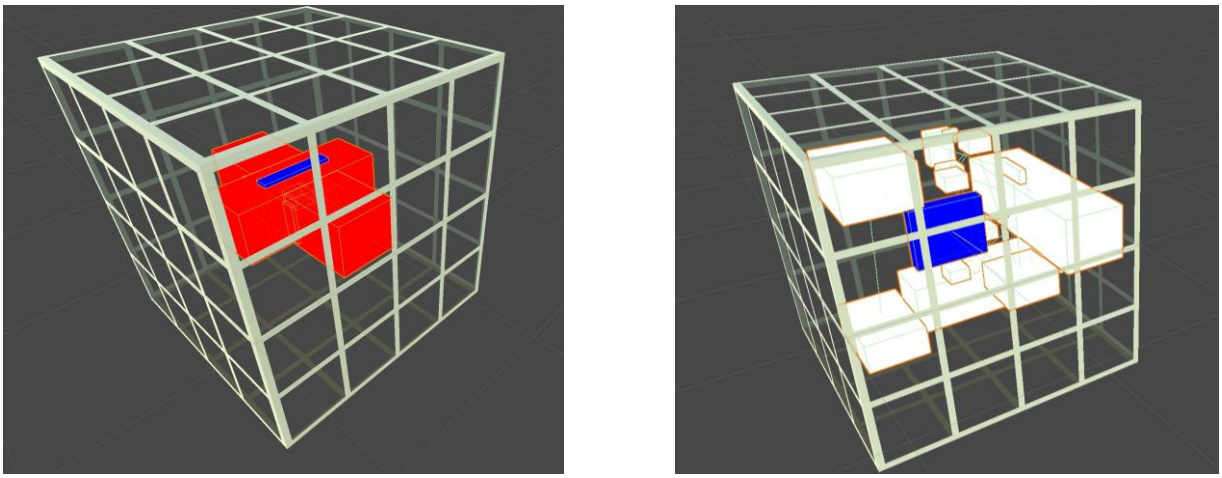

Figure 8: Medium example: starting position (left) and final position (right). Blue components are fixed, red components overlap, and white components are neither fixed nor overlapping.

seconds on a Surface Pro 4 with 8GB RAM, Intel Core i7-6650 CPU (2 cores, 2.2 GHz base frequency).

Metrics are provided in Table I for the system in its original position and after running the algorithm. Note that all components were centered in the zone at the start of the algorithm, so the sum of the distance between components and the sum of the attractive energies are essentially zero. The repulsive energy, on the other hand, is quite large. At the end of the run, the energies are fairly well balanced and the loss function value is significantly smaller than at the start. All collisions were resolved. Screen shots of the start and end positions are shown in Figure 8.

TABLE I: Medium-sized example: metrics before and after running code

	Before	After
Sum distance	2e-15	50
Repulsive energy $\times 10^3$	18,973,670	0.166
Attractive energy $\times 10^3$	4e-18	0.139
Loss function	158,113,940	16.7
Number of Collisions	225	0
Number of Components	16	
Number of Iterations	1000	
Run Time (seconds)	72.3	

Large Example

Our final example represents the complexity of a full ship system in the early stages of design. The one-line diagram of the electrical system, containing 282 components, is shown in Figure 9.

In this application of the algorithm, components were placed in approximate positions at the beginning, but numerous overlaps existed. Some components were not placed and thus had a location of (0,0,0). Eight of the larger components, e.g. the generators, were fixed in position. Each component was assigned to one of four zones; these zones are distributed along the longitudinal axis of the ship.

Time to run this example was 207.9 seconds for 1000 iterations. As an indication of the improved speed of the tensor-formulated gradient-descent algorithm, running this example using pairwise comparison instead of gradient descent took 3694 seconds.

Metrics are provided in Table II for the system in its original position and after running the algorithm. Note that components had been placed in approximately correct positions prior to running the algorithm, so there is not a large change in the sum of distances. All overlaps were, however, resolved. There is an imbalance in the final energies, and the loss function, although greatly reduced, is much higher than the loss function value in the medium example. This is mainly due to the fixed components which, since they do not move, do not change the forces between them. Figure 10 shows the full set of components in their final position.

TABLE II: Large example: metrics before and after running code

	Before	After
Sum distance	809.7	876.4
Repulsive energy $\times 10^3$	236,854,640	1,106,805
Attractive energy $\times 10^3$	6.13	5.63
Loss function	infinity	49,027
Number of Collisions	1,133	0
Number of Components	248	
Number of Iterations	1000	
Run Time (seconds)	207.9	

CONCLUSIONS AND FUTURE WORK

We have developed and demonstrated an algorithm for the placement of components in a shipboard environment with the ability to detect and resolve overlaps, contain components within a specified region, and consider the interactions between components. The algorithm and several examples have been presented within this paper.

There are a number of areas for further exploration and future work, some of which are described below.

Nuances to the placement of components are not yet considered. For example, components should be placed with enough space to be accessed for maintenance, while allowing maintenance space to be shared between components to make designs that are more compact. Similarly, piping and liquid-bearing components tend to be low in a space, and electrical components tend to be higher. For survivability purposes, components with similar functions are separated. Components linked by a rotating shaft must be axially aligned. Components linked by cables must allow space for bending radius considerations. None of these factors are currently addressed, and all of them would improve the placement algorithm.

Very large sets of components, especially those with numerous zonal assignments, are somewhat slow in execution: on the order of a few minutes. When creating many thousands of ship designs, even using high-performance computers, a faster algorithm would be better. Methods for speeding up the calculations should be investigated. For example, it may be possible to place subsets of the component list, then combine results.

Further investigation into various weightings of the different forces would yield a recommended set of weightings, possibly tailored to the project in hand. In a related manner, the component locations could be normalized by the size of the compartment in order to spread items through the compartment more evenly. This investigation should also include consideration of situations in which the components cannot fit into the space available.

This paper describes a robust first step in the arrangement of components for the automation of early-stage design of ship systems. We look forward to continued work to further improve the algorithm.

REFERENCES

[1] D. J. Singer, N. Doerry, and M. E. Buckley, "What is set-based design?" *Naval Engineers Journal*, vol. 121, no. 4, pp. 31–43, 2009.

[2] D. N. Mavris and M. R. Kirby, "Technology identification, evaluation, and selection for commercial transport aircraft," in *Annual Conference of the Society of Allied Weight Engineers, Inc. (SAWE)*. SAWE, May 24-26, 1999.

[3] L. Bonfiglio, P. Perdikaris, J. del Aguila, and G. E. Karniadakis, "A´ probabilistic framework for multidisciplinary design: Application to the hydrostructural optimization of supercavitating hydrofoils," *International Journal for Numerical Methods in Engineering*, vol. 116, no. 4, pp. 246–269, 2018.

[4] E. Broughton, R. Smart, R. Leonard, R. Dougal, J. Chalfant, I. Leonard, and N. A. Robertson, "High temperature superconducting cable study driving M&S tool development," in *2019 IEEE Electric Ship Technologies Symposium (ESTS)*, August 2019.

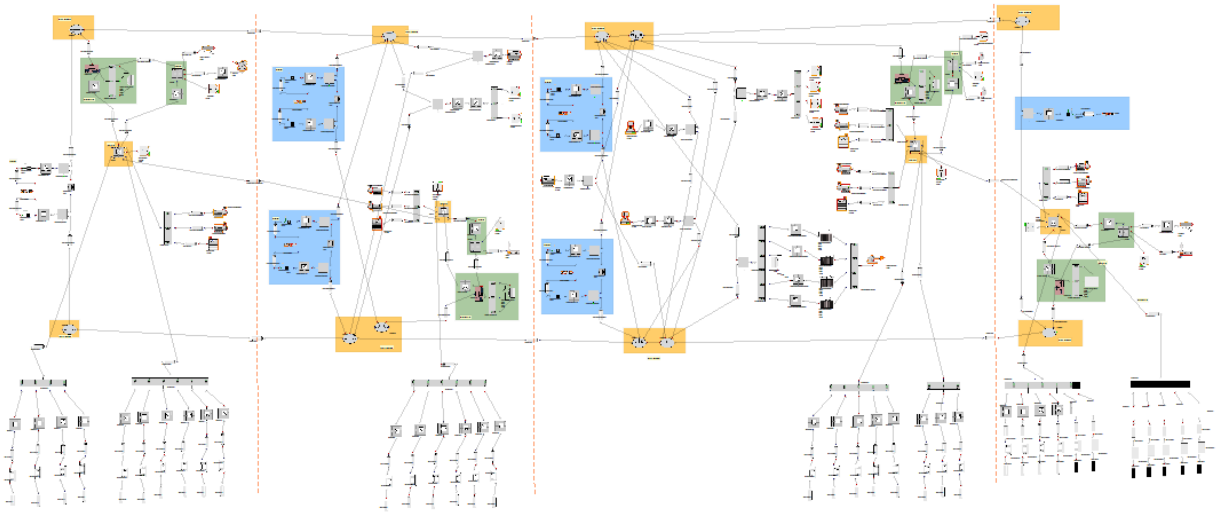

Figure 9: Large example: electrical one-line diagram

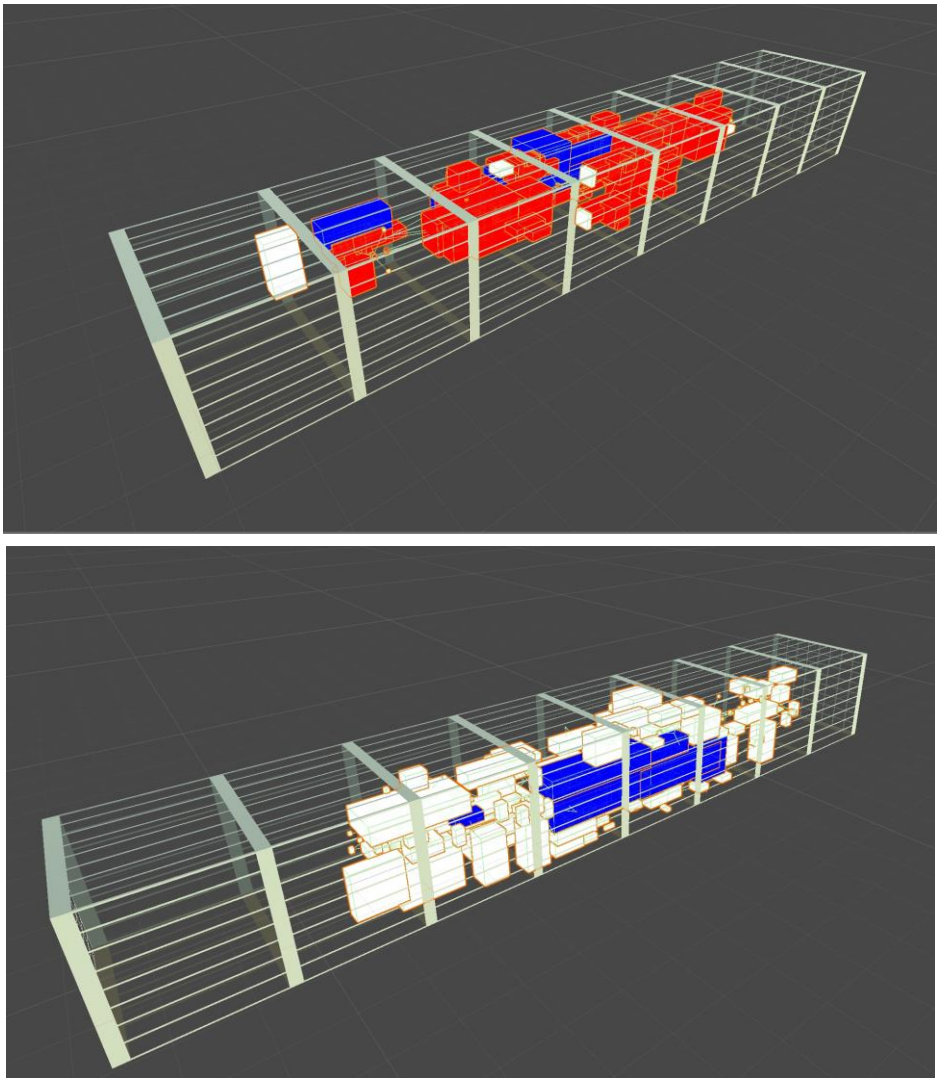

Figure 10: Large example: beginning position (top) and final position (bottom). Blue components are fixed, red components overlap, and white components are neither fixed nor overlapping.

[5] B. Langland, R. Leonard, R. Smart, and R. A. Dougal, "Modeling and data exchange in a concurrent and collaborative design environment for electric ships," in *2015 IEEE Electric Ship Technologies Symposium (ESTS)*, June 2015, pp. 388–394.

[6] M. Ferrante, J. Chalfant, C. Chryssostomidis, B. Langland, and R. A. Dougal, "Adding simulation capability to early-stage ship design," in *2015 IEEE Electric Ship Technologies Symposium (ESTS)*, June 2015, pp. 207–212.

[7] S. G. Kobourov, "Force-directed drawing algorithms," in *Handbook of graph drawing and visualization*, R. Tamassia, Ed. Chapman and Hall/CRC, 2013, pp. 383–408.

[8] J. Chalfant, Z. Wang, and M. Triantafyllou, "Expanding the design space explored by S3D," in *2019 IEEE Electric Ship Technologies Symposium (ESTS)*, August 2019.

DaMarcus Patterson is a recent graduate of the Massachusetts Institute of Technology, with a bachelor's degree in Electrical Engineering and Computer Science. He will be joining MasterCard as an associate analyst.

Dr. Julie Chalfant, PhD, is a research scientist in the Design Laboratory of the MIT Sea Grant College Program and a retired officer of the U.S. Navy. Her current areas of research include ship design and integration, including early-stage ship design tools. She received her PhD from MIT.

Dr. Michael Triantafyllou, PhD, is the Henry L. and Grace Doherty Professor in Ocean Science and Engineering; a Professor of Mechanical and Ocean Engineering; and the Director of MIT Sea Grant. His research interests include biomimetic ocean robots and sensors, flow-structure interaction, and dynamics and control of ocean vehicles.