Ant colony optimization method for bi-objective distribution system design space exploration

Colin P. F. Shields[a], Mike J. Sypniewski[a], David J. Singer[a]

[a] *University of Michigan, United States*

Abstract

ACO

Keywords:
System architecture, network theory, early-stage ship design

1. Introduction

Focus on distribution system design, introduce optimizing distribution system of systems

Distribution systems: Importance to ship design Difficulty Approaches

Challenge - How to explore a multi-system design space

2. Background

This section introduces network representations of vessel and system of systems, a system of systems survivability metric, and ant colony optimization methods.

2.1. Network Representations of Vessels and Systems

Generation and analysis methods for distribution systems often use network representations of systems and the physical vessel [1, 2, 3, 4]. Networks, or graphs, are a collection of objects, called nodes, and the connections between them, called edges [5]. Generally, networks are mathematically defined by their adjacency matrix A. The adjacency matrix is an $[n \times n]$ matrix where n is the number of nodes in the network. The elements of A denote the network edges: $A_{ij} = 1$ if an edge exists from node i to j, otherwise $A_{ij} = 0$.

In this paper, we use two network representations, one for vessel configuration and the other for systems. These networks allow the consideration of

interactions between system elements and the physical vessel without requiring geometric CAD modeling. The proposed method optimizes the network representing the system to explore how interactions between system components in the physical vessel affect system performance. This makes the proposed approach applicable to the earlier stages of ship design when design detail required for CAD modeling is not available. Further, network representations can scale with design fidelity [2, 6] meaning the method can be used throughout the design process.

2.1.1. Vessel Configuration Network

Vessels can be represented as a network V by defining modeling volumes and their adjacencies in physical configuration. Shown in Figure (here), the vessel is first divided into volumes, for instance compartments or structural zones. These volumes become nodes in V and adjacencies between volumes are edges.

The configuration of system components in the vessel are defined by the allocation of components to the volumes in V. Component allocation is mathematically defined by matrix a $[k \times n]$ matrix ζ, where k is the number of installed system components and n is the number of volumes represented in V. If component i is allocated to volume j, $\zeta_{ij} = 1$, otherwise $\zeta_{ij} = 0$.

Convert ship to network picture here.

2.1.2. Distribution System Network

Distribution systems between shipboard components can be thought of as a set of paths through the vessel configuration. The composite of these paths is a network that is necessarily a subset of V. However, there may be multiple types of distributions systems (i.e. electrical power or chill water) and their routings may be co-located throughout the vessel. Describing the connections within a single type of system and the interdependencies between systems due to shared routing paths requires a network of networks representation. A network of networks, or multiplex network, is defined as a set of m network layers where each layer l is an adjacency matrix of the same n nodes. Within each layer, the edges between nodes are independent of the edges in other layers. For example, an edge would exist between a generator and a crane in the electrical power distribution layer, but would not exist in the chill water system distribution layer.

Physical relationships within the vessel configuration define the possible routes for the distribution system between components. Thus, the nodes in

the distribution system multiplex D are the same nodes defined in the vessel configuration network V. Within multiplex layers, edges between nodes denote routing connections for the represented distribution system. In this paper, systems are modeled as directed, capacited network to capture resource flow. Each edge $D_{l,ij}$ in a multiplex layer l is weighted with capacity c, $c \in C_l$, defining maximum resource flow from node i to node j, where C_l is a set of positive integers representing the possible installed capacities for system l. Components allocated within the vessel are flow sources and sinks, denoted by their demand d. The demand of component i in system layer l is denoted by $d_{l,i}$. Following conventional notation, sinks have positive demand and sources have negative demand. Figure (here) demonstrates a multiplex network for two distribution systems.

demonstration

2.2. ACO

Ant Colony Optimization (ACO) is an evolutionary meta-heuristic optimization scheme inspired by the foraging behavior of ants. Originally applied to the Traveling Salesman Problem [7], ACO has since been successfully employed in many instances of combinatorial optimization problems ([8] provides an overview). The ACO approach uses colonies of ants which explore a solution space and deposit artificial pheromones to guide future ants. Ants who find strong solutions deposit relatively larger amounts of pheromone, positively reinforcing its solution and increasing the local search power. The simple mechanics of AS create a powerful optimization algorithm that guides solution design while preventing premature convergence.

The original ACO algorithm was the Ant System (AS) approach [7] which had ants probabilistically follow a pheromone trail along a network. An ant at node i would chose to move to a neighboring node j using on a transition rule based on the amount of pheromone τ_{ij} between the node pair. The transition rule, defined as

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{h \in \Omega} \tau_{ih}^{\alpha} \cdot \eta_{ih}^{\beta}} \tag{1}$$

where η_{ih} is some heuristic information about the preference of node h, α and β are the influence factors of the pheromones and heuristic, and Ω is the set of neighbors of node i. After each ant created a solution to the TSP, pheromones were updated as

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^{A} \Delta\tau_{ij}^{k} \tag{2}$$

where ρ is the evaporation pheromone evaporation rate, A is the total number of ants, and $\Delta\tau_{ij}^{k}$ is the amount of pheromone ant k deposits based on the quality of its solution. After the pheromones are updated, the next generation of ants are started, using the new pheromones to guide their transition rules. Ant generations are iterated, progressively reinforcing successful paths, until a convergence criteria or computation time is met.

2.3. ACO Extensions

Since its inception, significant extensions have been made to the original AS algorithm in order to improve solution quality and add multi-objective optimization functionality. Variants such as elitist solution updating (elitist ants), pheromone constraints [9](add hypercube), and local search and reinitialization [10] aim to improve solution quality and search capability. Multi-objective Ant Colony Optimization (MOACO) extensions such as [11, 12] use combinations of multiple heuristics and pheromone trials to encourage ants to explore solutions along a Pareto front. [13] and [14] provide extensive reviews of MOACO method and applications.

The proposed SoS design method is a bi-objective optimization between survivability and representative cost. Following the approach used by [12], pheromones for each objective are encoded in $M = (\tau_{ij})$ and $M^{'} = (\tau_{ij}^{'})$ respectively. Solution search is guided along the Pareto front by separating m ants into p colonies of m/p ants and creating composite pheromones that interpolate between objectives. In the most straight-forward implementation, each ant k, $k \in [1, m/p]$ in a colony uses weighting factor $\lambda_k = \frac{k-1}{m/p-1}$ to determine the influence of each objective. More complicated weighting rules create disjoint or overlapping λ-intervals between colonies. Regardless of weighting, the transition rule (1) is modified to

$$p_{ij} = \frac{\tau_{ij}^{\lambda_k \alpha} \cdot \tau_{ij}^{'(1-\lambda_k)\alpha} \cdot \eta_{ij}^{\lambda_k \beta} \cdot \eta_{ij}^{'(1-\lambda_k)\beta}}{\sum_{h \in \Omega} \tau_{ih}^{\lambda_k \alpha} \cdot \tau_{ih}^{'(1-\lambda_k)\alpha} \cdot \eta_{ih}^{\lambda_k \beta} \cdot \eta_{ih}^{'(1-\lambda_k)\beta}} \tag{3}$$

where η_{ij} and $\eta_{ij}^{'}$ are heuristic information for the first and second objectives respectively. It should be noted that each colony p has separate pheromones M_p and $M_p^{'}$ which ants from that colony use in Equation 3. [12] proposed

two pheromone update rules of this method: update by origin and update by region of non-dominated front. The former updates only an ant's colony of origin with solution pheromones. The latter updates a colony's pheromones based on a solution's location on the non-dominated front, in effect making colonies responsible for specific solution spaces. The following section will apply elements of these extensions to a create an bi-objective ant colony SoS design exploration method.

3. SoS ACO Methodology

System of systems design requires significant alteration and extension of the existing ACO methods. The largest alteration is that solutions are composed of multiple interdependent networks instead of the single network solutions of classic approaches. SoS solution generation thus requires each ant to create a network of each distribution system's paths through V. In the proposed method, for each system $i = 1, \ldots, l$, an ant creates the system network D_i independently of the other systems. The composite multiplex network D is then permuted in a localized search method and initial and permuted solutions are evaluated to quantify the performance of the SoS solutions. After a full generation of ants have been generated and evaluated, solutions on the Pareto front are used to update the pheromone matrices for the next ACO generation. This process is described in Figure (here) and the remainder of this section will detail its execution and describe a naval survivability demonstration.

Process

3.1. SoS Generation

Complications in solution generation arise from differences in SoS architecture design and the TSP network traversal problem. First, SoS's are not well suited for design by a single ant. They often include multiple sources and sinks which must be connected through different system types. This connectivity is directed - sources supply sinks - and branching meaning a lone ant would be unable to create a system without significant network traversal and looping. To address this, the proposed method allows an ant to create multiple branches each time an ant leaves a node. This method has been employed in multi-path routing of wireless sensor networks by [15], but is significantly expanded in this application. For each node i in the network

containing the SoS, a set of branching pheromones B are stored for the number of branches b an ant may create where $b \in [1, \mathcal{N}_i]$ and $\mathcal{N}_i$ is the number of neighbors of i. An ant at node i then choses how many branches b to create using the following transition rule,

$$p_{ib} = \frac{B_{ib}^{\lambda_k \alpha} \cdot B_{ib}^{'(1-\lambda_k)\alpha}}{\sum_{h \in \mathcal{N}_i} B_{ih}^{\lambda_k \alpha} \cdot B_{ih}^{'(1-\lambda_k)\alpha}}. \tag{4}$$

Note that unlike the original implementation of bi-criterion ACO, (4) does not use a heuristic. In testing, no heuristic improved the solution quality or convergence behavior of the algorithm.

Once the number of branches is chosen, b directed edges are added to the solution network by applying (3) b times. In Algorithm **??** this process is called antBranch(i, antID) where antID defines the ant number and colony for calculating λ_k. Each time (3) is applied, an edge from i to the chosen neighbor j is added to the network G^{ψ} which defines the structure of system ψ. j then removed from the subsequent transition calculations, ensuring that b branches are added in total. In Algorithm **??** this process, adding b directed edges from i, is called antNode(i, b, antID) which returns a list of nodes chosen. This process, is repeated for each active branch in a solution generation step, enabling an ant to create complex branching and converging system structures.

Introducing branching behavior creates difficulty with the termination criteria for an ants solutions. Classic applications to TSP consider a solution completer once all nodes in a network are covered. In SoS design, it may be undesirable to cover all possible nodes due to increased routing cost or the creation of unnecessary system interdependencies. Avoiding this requires a new termination criteria. An obvious approach is to terminate branches of the solution once they encounter a system sink. However, this prevents routing sinks in serial and artificially limits solution freedom. To avoid this, the proposed method uses a consensus-based approach to termination: once all system sinks are included in the solution, branches at node i chose to termination outcome t, $t \in [0, 1]$ following the transition rule,

$$p_{it} = \frac{T_{it}^{\lambda_k \alpha} \cdot T_{it}^{'(1-\lambda_k)\alpha} \cdot \zeta_{it}^{\beta}}{\sum_{h \in [0,1]} T_{ih}^{\lambda_k \alpha} \cdot T_{ih}^{'(1-\lambda_k)\alpha} \cdot \zeta_{ih}^{\beta}} \tag{5}$$

where termination pheromones for i are stored in T and ζ is the termination heuristic. Probabilistically choosing $t = 0$ continues the branch and

$t = 1$ terminates the branch at node i, preventing it from moving to any other nodes. Note that because, branches are prevented from terminating unless all sinks are included in the network, $p_{i1} = 0$ until that condition is met. This process, antTermination(i, antID) causes solution generation to conclude once all branches have been terminated. These steps are combined with the standard ACO transition rule to generate network structures for each system in the SoS. Networks are generated with Algorithm **??** which is initialized by setting a branch at each source node s in system ψ. The final step in Algorithm **??**, pruneGraph(), removes branches from graph that do not contribute to a path between sources and sinks. These extra branches are either a remainder from the termination process or small off-shoots paths that go to a non-source, non-sink node and return following the same path.

Creating each system network can be split into two steps - network design and capacity design - which are explained in Section 3.1.

feasible systems, decision rules, capacity.

3.2. System Generation

Generating solutions to an SoS design problem is more complicated than traditional solution to the TSP. First, SoS's are not well suited for design by a single ant. They often include multiple sources and sinks which must be connected through different system types. This connectivity is directed and capacitated - sources supply sinks through flows - and may be connected by multiple branches of the distribution system. This means a lone ant would be unable to create a system without significant network traversal and looping. To address this, the proposed method allows an ant to create multiple branches each time an ant leaves a node. This method has been employed in multi-path routing of wireless sensor networks by [15], but is significantly expanded in this application.

Incorporating branching walks into solution generation requires additional ant behaviors and associated transition rules. Unlike generating solutions to the TSP, where the only decision involved is which edge to traverse next, this method employs three decision types. The first decision occurs at the start of every step of the SoS generation process. Each ant in the walk uses a branching pheromone to select the number of branches it will create. Then for the number of branches selected, directed edges are added to the solution through an iterative application of the traditional ACO transition process. Finally after some termination criteria is met, the ant adds a flow capacity to each edge in the solution based on a third transition rule.

Each transition has a unique pheromone type, but uses the bi-criterion transition rule from [12]. XXX describes a generalized decision process, XXX, which takes a set of pheromones and heuristics and probabilistically returns a decision based on Equation 3. XXX also incorporates a dissipation factor X which reduces the pheromone in P corresponding to the decision process output for the remainder of the ants walk. This mimics ants "eating" the pheromone trail as they pass through it as seen in (cite ACS). In testing, the additional dissipation factor has been effective at preventing excessive cycling behavior during solution generation.

Algorithm here

Using XXX, Figure XXX describes the random walk process for creating single systems of the SoS. The penultimate step in the process, pruning, removes branches from graph that do not contribute to a path between sources and sinks. These extra branches are either a remainder from the termination process or small off-shoots paths that go to a non-source, non-sink node and return following the same path. The final step in the process checks the flow feasibility of the design, that the sink demands for the system are satisfied based on minimum cost flow.

Walk here

After the walk is completed, each edge in the solution is allocated a capacity through,

Capacity decision

where

Creating a solution to a SoS design problem requires implementing this process for each system in the SoS. The resulting multiplex D is a single ant's solution which can then permuted in local search and evaluated in the pheromone update process.

3.3. Local Search

ACO heuristic methods provide a coarse-grained search of a solution space, this is often augmented with a local optimizer to help refine the search (cite). Local search use the solution developed by an ant and attempt to improve it through methods ranging from genetic algorithms, to linear programming, to heuristic solvers. In this work, the local search is used to change pieces of the SoS networks with a crossover mutation scheme. This section will describe generally how local search is included in the SoS design algorithm. The exact local search algorithm used in a SoS design problem is dependent on the objectives being optimized. Later in this paper,

an example of local search for minimum cost and maximum survivability is provided.

After an ant completes the SoS generation process, its solution D acts as the parent for the search method. Depending on the employed method, the local search will create one or more child solutions. In the simplest process, the parent and child solutions are then added to the solution pool for evaluation and pheromone update. However, some local search methods may force premature convergence of the algorithm or limit the ACO exploration power. To mitigate this, we introduce a crossover that treats the child solutions as possible mutations to the parent solution.

Crossover is evaluated by parent-child pairs. First the network differences between the parent and child are identified. Then for each difference, the network element from the child solution is accepted with probability p_c. This creates a mixed SoS solution between the parent and child whose level of mixing is controlled by the definition of p_c. At extremes, $p_c = 0.0$ or $p_c = 1.0$ the mutation is either fully the parent or fully the child, respectively. The level of crossover may be uniform or function of the algorithm progression. For example, the p_c can start at 0.0 in the first generation of the ACO and increase as the algorithm progresses.

Crossover on networks introduces connectivity issues because the accepted elements of the child may not be connected to elements of the parent. To prevent disconnect or extraneous edges and nodes, after the crossover process is completed each network in the SoS solution is pruned with the same process described in Section 3.2. Following crossover, the original parent and mutated solutions are evaluated by the objective function and added to the solution pool.

3.4. Pheromone Update

The final step in the ACO process is updating the pheremone trials of each ant. However, the extensions that have been added to this method, complicate the application of (2). Before discussing the mechanics of the pheromone update, the exact structure of the pheromones used in the algorithm should be clarified. Because each system $i = 1 \dots l$ are designed independently, each has a separate pheromone structure used in the associated transition rules. Extending to incorporate bi-objective optimization requires using P^i and P'^i which are data structures containing all pheromone information for the first and second objective respectively. Further, following the approach in [12] each colony p has unique pheromone structure P_p^l and $P_p'^l$ for each system.

Ants from colony p use their respective structures during their generation process.

The proposed method adapts the generic update rule with a normalized quality function and elitist pheromone updates. Colony's pheromone structures are only updated by ants from that colony which increasing selection pressure and encourages search in less dense regions of the non-dominated front [12]. These adaptations provide two benefits: Normalization allows pheromones to be updated relative to their solution quality while maintaining stable pheromone scale [16]. Elitist updating accelerates solution convergence and improves solution quality by only allowing non-dominated solutions in a generation to update their pheromones [12].

After each ant in the generation creates feasible SoS solutions, solutions are evaluated by the objective functions. Non-dominated solutions s_{upd} are selected to update their pheromone trails, where $s_{upd,i} = [s_i, s'_i]$ and s_i and s'_i are the fitness for the first and second objective, respectively. Solution scores are normalized to the global best objective score s_{best} and s'_{best} by the quality functions $F(s)$ and $F'(s')$. $F(s)$ shows the quality function for a maximization and $F'(s)$ for a minimization.

$$\begin{aligned} F(s) &= \frac{s}{s_{best}} \\ F'(s') &= \frac{s'_{best}}{s'}. \end{aligned} \tag{6}$$

The pheromone update increments, θ_i and θ'_i, for ant i with fitness $[s_i, s'_i]$, are normalized to the total quality of all updating solutions. Pheromones increments are defined as,

$$\begin{aligned} \theta_i &= \frac{F(s_i)}{\sum_{a \in s_{upd}} F(a)} \\ \theta'_i &= \frac{F'(s'_i)}{\sum_{a \in s_{upd}} F'(a)}. \end{aligned} \tag{7}$$

The original pheromone update, (2), can be generalized to any pheromone in colony p, as

$$P_p \leftarrow (1-\rho)P_p + \rho \sum_{s \in s_{upd}} \theta_s \tag{8}$$

where ρ is the pheromone dissipation rate, p_s is the colony of origin for the ant producing s, and δ is the Dirac delta function.

Each pheromone in the pheromone structures are updated using (8), increasing system traits that appear in the non-dominated solutions. For example, if an ant from colony p has edge $D_{l,ij}$ with capacity c, $P^{l}_{p,ijc}$ and $P'^{l}_{p,ijc}$ are updated by θ and θ' respectively.

The update procedure described in this section is conducted after every ant has created feasible solutions to the SoS. This completes the SoS ACO, described in XXX encapsulates pheromone updating and S is the generation's solution pool. Convergence criteria can either by computation time, Pareto front characteristics, or based on some analysis of the pheromone structures (see [9] for pheromone convergence analysis).

3.5. Application

In naval vessels, sustaining operations while damaged, or survivability, is a critical aspect of distribution system design (cite here). While general design studies and guidelines provide suggestions for improving vessel survivability [17, 18], methods for optimizing the survivability of interdependent naval distribution systems are not currently available. The proposed ACO SoS design method will be used to demonstrated on a two system maximum survivability, minimum cost design study.

3.5.1. Survivability Fitness Function

The primary objective of naval distribution system design is to ensure the vessels mission capability in normal operating conditions and when damaged [19]. Thus distribution systems must be designed to sustain operation under a variety of damage scenarios. This requires a method for prediction damage to vessel and the subsequent failures it may cause. However, damage is not limited to local area of an attack. Sustaining damage to the distributed system may trigger successive failure elsewhere in the system. This type of cascading failure is particularly damaging to interdependent networks [20, 21] such as the naval distribution system of systems. Cascading failure prevention has been studied in general [22, 23] and naval specific cases [4]. However, these studies are limited to more theoretical damage scenarios such as occupation probabilities or single edge loss. This section will detail a survivability fitness function to consider cascading failures in interdependent distribution system networks under a generalized damage model.

Distribution system survivability in naval vessels is inherently tied to the geometric locations of interdependent systems [19]. When the vessel is damaged, components and distribution routings in the damaged area are isolated until their functionality is assessed. In this model, component functionality is tracked in a vector F where $F_i = 1$ if component i is functional and $F_i = 0$ otherwise. Components that are not functional have their demand set to 0 in each of its corresponding elements of d. The vessel functionality ψ is the percentage of functional components after damage is assessed. Given a distribution system network D that is contained in the geometric vessel representation V, the initial damage the vessel can be modeled as:

1. Identify the network elements (node and edges) of V corresponding to the area(s) that are damaged.
2. For each damaged element in V, remove that element from the distribution system multiplex layers.
   (a) If the element is a node V_{jj}, remove node $D_{i,jj}$ and edges connected to that node $D_{i,\cdot j}$ and $D_{i,j\cdot}$ for multiplex layers $i = 1 \dots l$. Components located at V_{jj} are marked as not functional in F.
   (b) If the element is an edge V_{jk}, remove edge $D_{i,jk}$ and it reciprocal edge $D_{i,kj}$ for multiplex layers $i = 1 \dots l$.

Initial damage can be defined in a number of ways to model different scenarios. For example the removal of a set of nodes and edges to represent specific hit, removal of a set of nodes and edges at a random location to represent a unique damage pattern, or probabilistic removal for incidental damage or more complex attacks. After initial damage is applied, the cascading effects are evaluated. In this fitness function, failure will be evaluated based on resource flow through the SoS network D using their demand d. Flow demand satisfaction of each distribution system layer D_i is evaluated for demands d_i with a minimum cost flow algorithm. Every component whose demand in is not satisfied in any system layer, it is marked as not functional in F and has its demand set to zero. If any components are found to be not functional, the flow evaluation process is run again, otherwise the process terminates and ψ is calculated.

In the presented design study, damage is applied as the uniform random removal of four nodes. The damage is chosen to demonstrate the possible applications of the survivability evaluation function, not model a specific scenario. Due to the random damage, survivability evaluated 35 times for each SoS design and the average functionality $\bar{\phi}$ is taken as the fitness value.

3.5.2. Cost

This study uses a cost based fitness function as a trade off to survivability. Without cost, the ACO forces solutions towards the most densely connected distribution systems as those tend to be more survivable. The resulting fully connected distribution system network is unlikely to represent a realistically feasible solutions due producibility concerns. However, a network based SoS producibility metric is not available so a simple cost metric is used as a surrogate to similar effect. The SoS cost fitness ψ is the sum of capacities over all edges in an SoS D.

3.5.3. Local Search

The coarse-grained ACO search is refined with the local search and crossover method described in Section 3.3 using two child solutions - one attempting to optimize each fitness function. Crossover probability p_c increases with the number of generations like

pc equation

After an ant generates a solution, the child solution for survivability is created by making every edge in D bi-directional. If the edge from node j to k in layer i exists, but the reciprocal edges $D_{i,kj}$ does not, $D_{i,kj}$ is added to the graph with the same capacity weighting as $D_{i,jk}$. The survivability local search creates potential flow reroutings without adding new network branches. More powerful survivability improvement methods exist in the literature, but the random damage cases make secondary optimization computationally expensive and not analytically tractable. Edge reciprocation is independent of the damage case, easily computed, and is effective for the presented study.

The minimum cost child solution is simply the edges of D used in a min-cost max-flow solution. The min-cost max-flow graph f_i is computed for each multiplex layer i using the respective component demands. If an edge in distributed system network has no flow in f_i, it is removed from D_i. While there may be many combination of min-cost max-flow solutions for a distribution system, this local search uses a single random solution. Once again, this is not the most elegant methods, but it is computationally efficient and is effective for the problem.

3.5.4. Vessel and System

4. Results and Discussion

Application results, why it matters

5. Conclusion

Broaden out

References

[1] J. W. Gillespie, A. S. Daniels, D. J. Singer, Generating functional complex-based ship arrangements using network partitioning and community preferences, Ocean Engineering 72 (2013) 107–115. doi:10.1016/j.oceaneng.2013.05.007.

[2] D. T. Rigterink, Methods for Analyzing Early Stage Naval Distributed Systems Designs, Employing Simplex, Multislice, and Multiplex Networks, Ph.D. thesis, University of Michigan (2014).

[3] R. A. Dellsy, M. C. Parker, D. T. Rigterink, Multi-Scale, Interdisciplinary Systems Analysis for Naval Platforms, Naval Engineers Journal 2 (127) (2015) 93–100.

[4] T. A. Trapp, Shipboard Integrated Engineering Plant Survivable Network Optimization, Ph.D. thesis, Massachusetts Institute of Technology (2015).

[5] M. E. J. Newman, The structure and function of complex networks, SIAM Review 45 (2) (2003) 167–256.

[6] J. W. Gillespie, A Network Science Approach to Understanding and Generating Ship Arrangements in Early-Stage Design, Ph.D. thesis, University of Michigan (2012).

[7] M. Dorigo, V. Maniezzo, A. Colorni, Ant System : Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics 26 (1) (1996) 29–41.

[8] M. Dorigo, G. Di Caro, The Ant Colony Optimization Meta-Heuristic, in: New Ideas in Optimization, Vol. 2, 1999, pp. 11–32.

[9] T. Stützle, H. H. Hoos, MAX-MIN Ant System, Future Generation Computer Systems 16 (8) (2000) 889–914. doi:10.1016/S0167-739X(00)00043-1.

[10] M. Ashraf, R. Mishra, Extended Ant Colony Optimization Algorithm (EACO) for Efficient Design of Networks (2013) 939–950.

[11] D. Angus, C. Woodward, Multiple objective ant colony optimisation, Swarm Intelligence 3 (1) (2009) 69–85. doi:10.1007/s11721-008-0022-4.

[12] S. Iredi, D. Merkle, M. Middendorf, Bi-criterion optimization with multi colony ant algorithms, Evolutionary Multi-Criterion Optimization, Proceedings 1993 (2001) 359–372. doi:10.1007/3-540-44719-9.

[13] C. García-Martínez, O. Cordón, F. Herrera, A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP, European Journal of Operational Research 180 (1) (2007) 116–148. doi:10.1016/j.ejor.2006.03.041.

[14] J. Rada-Vilela, M. Chica, Ó. Cordón, S. Damas, A comparative study of Multi-Objective Ant Colony Optimization algorithms for the Time and Space Assembly Line Balancing Problem, Applied Soft Computing 13 (11) (2013) 4370–4382. doi:10.1016/j.asoc.2013.06.014.

[15] J. Yang, M. Xu, W. Zhao, B. Xu, A multipath routing protocol based on clustering and ant colony optimization for wireless sensor networks, Sensors 10 (2010) 4521–4540. doi:10.3390/s100504521.

[16] C. Blum, M. Dorigo, HCACO: The hyper-cube framework for Ant Colony Optimization, IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 34 (2) (2004) 1161–1172. doi:10.1109/TSMCB.2003.821450.

[17] N. Doerry, Designing electrical power systems for survivability and quality of service, Naval Engineers Journal 119 (2007) 25–34. doi:10.1111/j.0028-1425.2007.00017.x.

[18] S. L. Y. Kok, Naval Survivability and Susceptibility Reduction Study - Surface Ship, Ph.D. thesis, Naval Postgraduate School (2012).

[19] C. J. V. Amy, Considerations in the design of naval electric power systems., Power Engineering Society Summer Meeting, 2002 IEEE 1 (2002) 331–335.

[20] J. Gao, S. V. Buldyrev, H. E. Stanley, S. Havlin, Networks formed from interdependent networks, Nature Physics 8 (1) (2011) 40–48. doi:10.1038/nphys2180.
URL http://www.nature.com/doifinder/10.1038/nphys2180

[21] C. M. Schneider, N. a. M. Araújo, H. J. Herrmann, Algorithm to determine the percolation largest component in interconnected networks, Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 87 (2013) 1–5. doi:10.1103/PhysRevE.87.043302.

[22] C. M. Schneider, N. a. M. Araujo, S. Havlin, H. J. Herrmann, Towards designing robust coupled networksarXiv:1106.3234, doi:10.1038/srep01969.
URL http://arxiv.org/abs/1106.3234

[23] C. D. Brummitt, R. M. D. Souza, E. A. Leicht, Suppressing cascades of load in interdependent networks 109 (12). doi:10.1073/pnas.1110586109.