Ant Colony Optimization for System Architecture Design

Colin P. F. Shields[a,1], Micheal J. Sypniewski[a], David J. Singer[a]

[a]*Department of Naval Architecture and Marine Engineering, University of Michigan*

Abstract

In this paper...

Keywords:
Ants, Systems

1. Introduction

1.1. System of systems

1.2. ACO

2. Problem Formulation

2.1. Network Representation of Systems

G is the spatial network.

2.2. Resilience Objective Function

2.3. Representative Cost Objective Function

2.4. Penalty Function

3. ACO Approach

3.1. Conventional 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 (Dorigo et al., 1996), ACO has since been successfully employed in many instances of combinatorial optimization problems (Dorigo and Di Caro (1999) provide an overview). The ACO approach

*colinsh@umich.edu

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 preventing premature convergence.

The original ACO algorithm was the Ant System (AS) approach (Dorigo et al., 1996) which had ants probabilistically follow a pheromone trail along a TSP graph. 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.

3.2. 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 (Stützle and Hoos, 2000)(add hypercube), and local search and reinitialization (Ashraf and Mishra, 2013) aim to improve solution quality and search capability. Multi-objective Ant Colony Optimization (MOACO) extensions such as (Angus and Woodward, 2009;

Iredi et al., 2001) use combinations of multiple heuristics and pheromone trials to encourage ants to explore solutions along a Pareto front. García-Martínez et al. (2007) and Rada-Vilela et al. (2013) 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 Iredi et al. (2001), 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 straightforward 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 (3). Iredi et al. (2001) 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.

4. System of system ACO

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 system ψ in the SoS $mathcalS$. In the proposed method, each system network in the SoS is created independently for each ant's solutions. The composite of these networks can then be evaluated to

quantify the performance of the SoS, giving a fitness to the ant's solution. Creating each system network can be split into two steps: structure design and capacity design.

4.1. System structure design

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 Yang et al. (2010), 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. However, for the remainder of this approach, transition rules will use a single heuristic. The chosen heuristics guide the generation process towards feasible solutions and are not suitable to be assigned to an objective. Further, the complexity of the objective functions makes it difficult to create an informative heuristic. The details of each heuristic will be discussed in Section 4.3.

Once the number of branches is chosen, b directed edges are added to the solution network by applying (3) b times. In Algorithm 1 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 1 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,

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

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 1 which is initialized by setting a branch at each source node s in system ψ. The final step in Algorithm 1, 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.

4.2. System capacity design

Generating network structures provides new information about emergent interdependence in the SoS. However, understanding how interdependence

Algorithm 1 Network structure algorithm

```
1: procedure NETWORK STRUCTURE(netStructure(ψ, antID))
2:     Initialize G^ψ as empty network
3:     N = s
4:     Complete = 0
5:     while not Complete do
6:         newN = {}
7:         for each node i ∈ N do
8:             t = antTermination(i, antID)
9:             if not t then
10:                b = antBranch(i, antID)
11:                n = antNode(i, b, ,antID)
12:                append n to newN
13:            end if
14:        end for
15:        if newN is empty then
16:            Complete = 1
17:        end if
18:        N = newN
19:    end while
20:    pruneGraph(G^ψ)
21: end procedure
```

affects the functionality of the SoS (in this case survivability) requires evaluating flow through the systems. In this paper, flow is addressed through independent min-cost network flow analysis of each system. Flow constraints due to interdependent routings are captured by edge capacities allocated for each network. Capacitated networks are developed similarly to the network structure, using pheromones and an adapted transition rule. After Algorithm 1 has completed, the ant choses a capacity c of each network edge G^{ψ}_{ij} using the transition rule,

$$p_{ijc} = \frac{C_{ijc}^{\lambda_k \alpha} \cdot C_{ijc}^{\prime(1-\lambda_k)\alpha} \cdot \xi_{ijc}^{\beta}}{\sum_{h \in \mathcal{C}^{\psi}_{ij}} C_{ijh}^{\lambda_k \alpha} \cdot C_{ijh}^{\prime(1-\lambda_k)\alpha} \cdot \xi_{ijh}^{\beta}} \tag{6}$$

where C contains pheromones for capacity transitions, ξ is the capacity heuristic, and $\mathcal{C}^{\psi}$ are the possible edge capacities for system ψ. Choosing capacities for each edge in G^{ψ}, sysCapacity(G^{ψ}, antID), completes the network for system ψ.

Generating a network routing of system ψ does not guarantee the sink demands are satisfied for that system, only that there is a directed path from the source(s) to sink(s). Further, each ant creates every system of the composite SoS which must have all sink demands satisfied. Initially, the ants will create random solutions to each system with little assurance they will meet the SoS demands. Only accepting feasible solutions to the SoS - which satisfy all sink demands - into the solution pool helps the algorithm learn to create feasible solutions. Feasibility of $\mathcal{S}$ is checked with sinkSat($\mathcal{S}$) which evaluates if the sink demands in each system are met and propagates demand failure through network interdependencies. Including the feasibility criteria, the SoS solution process is defined in Algorithm 2.

4.3. Heuristics

Each transition rule in structure and capacity design steps use a single heuristic or no heuristic in the case of (4). The heuristics in (3), (5), and (6) are constructed to guide the generation process toward feasible solutions which satisfies the sink demands of the SoS. Feasibility heuristics were chosen to limit computation time by reducing the number of solutions that did not meet sink demands. Further, the complexity of the objective functions makes it difficult to provide a useful heuristics without considering the other systems in the SoS.

Algorithm 2 SoS design algorithm

```
1: procedure SoS DESIGN(sosDesign(n, p))
2:     satisfied=0
3:     for each ant i ∈ n in colony p do
4:         antID=(i,p)
5:         while not satisfied do
6:             Initialize S_antID as empty
7:             for each system ψ ∈ S do
8:                 G^ψ = netStructure(ψ, antID)
9:                 G_c^ψ = sysCapacity(G^ψ, antID)
10:                add G_c^ψ to S_antID
11:            end for
12:            satisfied=sinkSat(S_antID)
13:        end while
14:        Add S_antID to solution pool
15:    end for
16: end procedure
```

The heuristic for the modified version of (3) is structured to add preference to nodes that are closer to sinks that are not included in the current solution. This not only helps reduce the number of steps the algorithm must complete, it works to prevent cycling behavior. Cycling in solution generation occurs when a cyclic connected path has relatively high pheromone levels through the entire path. It is then possible for the ant to continually travel around the cycle without progressing the overall solution. To mitigate this, the heuristic η scales with the number of consecutive iterations in Algorithm 1 that have not added a new node to the network. The heuristic is defined as,

$$\eta_{ij} = \frac{NC}{(l_j/d + a)} \tag{7}$$

where NC is the number of consecutive iterations without added nodes, l_j is the shortest distance from node j to a sink not included in the solution, d is the diameter of G, and a is a constant to prevent dividing by 0 when j is a sink that is not included in the solution.

The termination heuristic ζ used in (5) is more straightforward, enforcing the termination criteria - all sinks are included in the solution. Further, only a choice of $t = 1$ terminates the branch, $t = 0$ has not effect. Thus ζ may be

defined as $\zeta_{i0} = 1$ regardless of system state and ζ_{i1} as,

$$\zeta_{i1} = \begin{cases} 1 & \text{if all sinks included} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

The final heuristic used in this method is ξ in (6) which guides the transition rule for edge capacity. Edge capacity is added once the network structure is known. This allows the capacity heuristic to use the relationship between the source-sink connections in a system that an edge supports. For an edge ij, c^t_{ij} is the total demand of sinks that have a path from a sources including ij. Similarly, c^s_{ij} is the total magnitude of sources that have a path to a sink including ij. The minimum of c^t_{ij} and c^s_{ij} represents the maximum usable capacity of the edge $c^u_{ij} = min(c^t_{ij}, c^s_{ij})$. With the goal of meeting sink demand at minimum capacity cost, the heuristic preferences the transition rule towards capacities near c^u_{ij}. ξ_{ijc} is calculated as follows,

$$\xi_{ijc} = 1 + \frac{1}{1 + |c^u_{ij} - c|} \tag{9}$$

The heuristics discussed in this section have been effective in solving the SoS design problem proposed in this paper. However, there are likely alterations to the feasibility heuristics that will improve solution generation. Additionally, heuristics that support the bi-objective framework in Iredi et al. (2001) could be developed and applied instead of the feasibility approach.

4.4. SoS 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 systems are designed independently, each has a separate pheromone structure $P^\psi = [\tau, B, T, C]^\psi$ which are used to create the transition rule for an ant creating a solution to system ψ. Extending to incorporate bi-objective optimization requires using P^ψ and P'^ψ for the first and second objective respectively. Further, following the approach in Iredi et al. (2001) each colony p has unique pheromone structure P^ψ_p and P'^ψ_p for each system in $\mathcal{S}$. Ants from that colony p use their respective structure 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 (Iredi et al., 2001). These adaptations provide two benefits: Normalization allows pheromones to be updated relative to their solution quality while maintaining stable pheromone scale, similar to Blum and Dorigo (2004). Elitist updating accelerates solution convergence and improves solution quality by only allowing non-dominated solutions in a generation to update their pheromones (Iredi et al., 2001).

After each ant in the generation creates feasible SoS solutions, solutions are evaluated by the survivability and representative cost objective functions. Non-dominated solutions s_{upd} are selected to update their pheromone trails, where $s_{upd,i} = [s_i, s'_i]$. Solution scores are normalized to the global best objective score s_{best} and s'_{best} respectively by the quality functions $F(s)$ and $F'(s')$. For the maximum survivability, minimum cost problem the quality functions are defined as,

$$\begin{aligned} F(s) &= \frac{s}{s_{best}} \\ F'(s') &= \frac{s'_{best}}{s'}. \end{aligned} \tag{10}$$

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

$$\begin{aligned} \theta_i &= \frac{F(s_i)}{\sum_{a \in s_{upd}} F(a)} \\ \theta'_i &= \frac{F'(s'_i)}{\sum_{a \in s_{upd}} F'(a)}. \end{aligned} \tag{11}$$

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

$$X_p \leftarrow (1-\rho)X_p + \rho \sum_{s \in s_{upd}} \theta_s \cdot \delta(p, p_s) \tag{12}$$

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 (12), increasing system traits that appear in the non-dominated solutions. For example, if an ant from colony p has edge ij in G^ψ with capacity c, $C^\psi_{p,ijc}$ and $C'^\psi_{p,ijc}$ are updated by θ and θ' respectively. This process is straightforward for edge, branches, and capacities. However, it is not obvious how to chose which termination pheromones to update because there is no way to identify which nodes branches were terminated at based on the network solution. Instead, nodes with out-degree $k^{out} = 0$ are considered as terminated nodes, $t = 1$. Treating termination this way creates regions of the network that have high termination probabilities when they are not often included in high fitness solutions. Ants are likely to end branches in these regions, while the branches are likely to continue in regions associated with successful solutions.

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 Algorithm 3, where updatePheromones(S) 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 Stützle and Hoos (2000) for pheromone convergence analysis).

Algorithm 3 SoS ACO algorithm

```
1: procedure ACO(ACO(n, p))
2:     while Some convergence critera not satisfied do
3:         S=sosDesign(n, p)
4:         updatePheromones(S)
5:     end while
6: end procedure
```

5. Results

6. Discussion

7. Conclusion

Ants

Angus, D., Woodward, C., 2009. Multiple objective ant colony optimisation. Swarm Intelligence 3 (1), 69–85.

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

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

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

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

García-Martínez, C., Cordón, O., Herrera, F., 2007. 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), 116–148.

Iredi, S., Merkle, D., Middendorf, M., 2001. Bi-criterion optimization with multi colony ant algorithms. Evolutionary Multi-Criterion Optimization, Proceedings 1993, 359–372.

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

Stützle, T., Hoos, H. H., 2000. MAX-MIN Ant System. Future Generation Computer Systems 16 (8), 889–914.

Yang, J., Xu, M., Zhao, W., Xu, B., 2010. A multipath routing protocol based on clustering and ant colony optimization for wireless sensor networks. Sensors 10, 4521–4540.