Using Ant Colony Optimization in the Real-World

Nathan Spilker

November 9, 2021

Word Count: 2512

Abstract

Ant Colony Optimization (ACO) algorithms are a powerful tool to solve hard combinatorial optimization problems. Since their discovery, many variants have been created to solve various computation problems in academic settings. A recent development in the field is to solve real-world engineering problems with ACO algorithms. A survey is presented over various implementations of ACO algorithms developed and implemented in the real-world. Examples of real-world engineering problems for which ACO solutions have been implemented are Vehicle Routing Problems, Water Reservoir Management, and Scrap Minimization in a Bar Mill. All of these use algorithms classified as ACO algorithms, but with minor changes to their formulation.

1 Introduction

Ant Colony Optimization (ACO) algorithms are members of a broader set of algorithms: Swarm Intelligence Optimization Algorithms [1]. The first ACO algorithm was created by Dorigo in 1996; the initial algorithm was referred to as the Ant System (AS) [2]. At their core, ACO algorithms are graph search algorithms that base their search heuristics on the behavior of ant colonies searching for a food source. Ants live in colonies and forage for food collectively, initially searching around their nest randomly. As each individual ant searches, it leaves a pheromone trail on the ground. While searching, ants tend to choose the trail with the most pheromone with the highest probability. If an ant finds a food source, it will make a return trip and, depending on the quality and quantity of the food source, will lay a pheromone trail with varying strength accordingly. This allows other ants to find their way to the food source. It has been shown that this behavior creates trails that converge on near-optimal paths to food sources [3]. The AS was the first algorithm to create a search based on this behavior, and it was applied to the Traveling Salesperson Problem (TSP) [2, 4]. Since their creation, ACO algorithms have been applied to problems such as vehicle routing problems [5], sequential ordering problems [6], protein folding [7], data mining [8], and others. More recently, implementations of the ACO algorithm have been tested on real-world engineering problems, proving their usefulness as an optimization algorithm outside of the academic setting [9]. For each of these problems, the formulation of the ACO algorithm must be adjusted to accommodate a particular problem. Although transitioning from theory to real-world application requires adjustments, the implemented algorithms are all still considered to be ACO algorithms. However, before we can discuss these adjustments, we must first look at the basic ACO algorithm and its formulation.

2 ACO Algorithm

2.1 Algorithm Definition

The various implementations of ACO algorithms take different forms, but all of them generally follow the basic algorithm shown in Figure 1.

```
procedure ACO algorithm for combinatorial optimization problems
    Initialization
    while (termination condition not met) do
        ConstructAntSolutions
        ApplyLocalSearch        % optional
        GlobalUpdatePheromones
    end
end ACO algorithm for combinatorial optimization problems
```

Figure 1: High-level outline of basic ACO algorithm [9].

This algorithm treats the graph as a collection of

nodes, C. A set of n ants will be traversing this graph, and call each ant's trajectory through the graph s_p. At each time step, given a trajectory s_p, there is a set of feasible nodes to jump to, given by $\mathcal{N}(s_p) \subset C$. The pheromone level between two nodes i and j is denoted as τ_{ij}. Now, we define each function within the ACO algorithm.

Initialization

Initialize n ants to traverse the graph, each with initial trajectory $s_p = \{0\}$. Initialize all pheromone levels to a chosen value τ_0.

ConstructAntSolutions

At each time step, this function will extend each ant's partial solution s_p to one more node. Each ant, given their trajectory s_p with final node i, must probabilistically select a jump to a node $c_i^j \in \mathcal{N}(s_p)$. The probability that the ant selects a jump is commonly given as [9]:

$$p(c_i^j|s_p) = \frac{\tau_{ij}^\alpha [\eta(c_i^j)]^\beta}{\sum_{c_i^l \in \mathcal{N}(s_p)} \tau_{il}^\alpha [\eta(c_i^l)]^\beta}, \quad \forall c_i^j \in \mathcal{N}(s_p)$$

Here, $\eta(\cdot)$ is a function that assigns a weight to the jump c_i^j. α and β are parameters that influence how heavily τ and $\eta(\cdot)$ affect the algorithm, respectively, and may be tuned differently for each problem.

ApplyLocalSearch

After the candidate solutions are updated, they may be further updated by local search algorithms, or finding better neighboring solutions to the ant trajectories. In general, ACO algorithms apply what are called daemon actions here [10]. These are problem-specific or centralized actions that take place to find a more optimal partial solution that the individual ant actions cannot find.

GlobalUpdatePheremomes

Finally, we apply this function, which updates the pheromone levels τ to make the best partial solutions more desirable for paths in following iterations. This step is a direct analogy to ants laying down pheromone along their paths. This includes two actions: a pheromone evaporation and a pheromone deposit. The pheromone evaporation prevents partial solutions from converging on locally optimal solutions and encourages the algorithm to explore new territory. The pheromone deposit places pheromone along the most optimal partial solutions. A common implementation for the pheromone update is given in [9]. Write the set of best partial solutions as S_{upd}. This is the set of solutions along which the algorithm will deposit pheromone. Write the pheromone update as:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s)$$

Here, ρ is referred to as the evaporation rate of pheromone. The right term, or deposit term, increases the pheromone level τ_{ij} if the jump corresponding to this pheromone level c_i^j is within a partial solution s which is within the set of best partial solutions S_{upd}. It updates according to the function $g(s)$ which maps the partial solution to an integer, often taken to be proportional to the inverse of length of the partial solution.

2.2 Discussion and Limitations

The ACO algorithm presented is a discrete space path optimization algorithm. This is suitable for discrete search space problems, such as the TSP [4]. However, for many applications, an algorithm which searches continuous space is required. Continuous search space ACO algorithms have been developed and are presented in [11, 12]. Both of these and other continuous ACO algorithms help in transitioning from academic implementations of ACO to real-world implementations [12]. The ACO algorithm also has many parameters that must be chosen by the operator. The initial pheromone level τ, the jump weight function $\eta(\cdot)$, the strength exponents of the pheromone level and weight function α and β respectively, the construction of set of best partial solutions S_{upd}, the pheromone evaporation constant ρ, the deposit function $g(s)$, and any daemon actions must be chosen in each instance, and quality values are highly problem-specific. Value definitions are often initially chosen based on prior formulations of the problem.

3 Real-World Implementations

3.1 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a problem calling for computation of the optimal routes for a fleet of vehicles to traverse to meet customer demands [13]. This problem was first proposed by Dantzig and Ramser in 1959 as the problem of routing a fleet of gasoline delivery trucks between a bulk terminal and service stations [14]. This problem, although a hard combinatorial problem to solve, often needs to have added complexity to represent real-world scenarios [16].

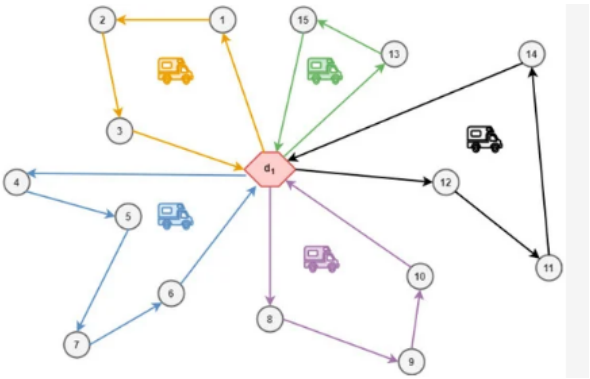

Figure 2: Depiction of the VRP [15].

Many variants of this problem have been formulated to better represent real-world vehicle routing problems. An example of one these formulations are the VRPTW, the vehicle routing problem with time windows, used when constraints on delivery times are present [16]. Another example is the VRPPD, the vehicle routing problem with pick-up and delivery, where transport items are distributed throughout the network [16]. The VRPPDTW is a combination of the two, a vehicle routing problem with pickup and delivery and time windows.

Rizzoli et al. provide solutions using variants of ACO for many problems taken directly from real-world vehicle routing problems in [16]. This paper analyzes a VRPTW for a major supermarket chain in Swirzerland, a VRPPDTW for a leading distribution company in Italy, a VRPTW for a logistics company in Padua, Italy, and a VRP for a Swiss oil distribution company. All of these problems were solved with variant ACO algorithms. In each case, the ACO solution showed considerable improvement over a solution created by a human planner [16]. This paper will present the variant ACO algorithm used to solve the VRPTW problem for a supermarket chain in Switzerland, and the results obtained.

VRPTW for Supermarket Chain

The client in this problem is one of the major supermarket chains in Switzerland. The problem is to distribute goods to over 600 stores throughout Switzerland. The goods must be delivered within time windows for adequate personnel to receive the goods. The road network is represented as edges connecting nodes, with edges rescaled based on the the average speed a truck will drive between the nodes. Nodes are given as the central distribution center and the stores. Thus, the problem is a discrete-space problem. In order to account for the time window aspect of this problem, distances between nodes are adjusted based on the time window: a node will effectively be closer if a shop closes soon [16].

The proposed solution to the VRPTW problem uses an implementation of the MACS-VRPTW called ANTROUTE. MACS stands for Multiple Ant Colony Search, and the MACS-VRPTW algorithm was first presented in 1999 by [17]. The MACS-VRPTW algorithm uses two ant colonies to search the graph. One colony, ACS-VEI, minimizes the vehicles, while the other colony, ACS-TIME, minimizes the time. The colonies are independent, but search the same graph. ACS-VEI starts by computing a solution using v vehicles. ACS-TIME then begins running using v vehicles trying to minimize time. Now ACS-VEI looks for a solution using $v-1$ vehicles. If it computes a solution, ACS-TIME runs with $v-1$ vehicles. This is repeated until an optimal solution which utilizes minimum trucks and takes minimum time is found [16].

The optimal solution calculated by MACS-VRPTW was compared with a route created by a human planner, and is shown in Figure 3. The AR-RegTW was constricted to regional travel and one hour time windows, whereas AR-Free used relaxed time windows and allowed global travel.

	Human planner	AR-RegTW	AR-Free
Total number of tours	2056	1807	1614
Total km	147271	143983	126258
Average truck loading	76.91%	87.35%	97.81%

Figure 3: Computer-generated tours vs. man-made tours [16].

The result obtained from the ACO algorithms showed considerable improvement over the human planner, showing promise in its use as a strategic planner for similar VRP problems [16].

3.2 Water Reservoir Optimization

Kumar and Reddy provide an implementation of real-world multi-purpose reservoir optimization in [19]. The reservoir considered in their model was the Hirakud reservoir in Orissa state, India. This reservoir is a multi-purpose reservoir, and the water that is available in the reservoir is used in the following order of priority: for flood control, drinking water, irrigation, and hydropower generation. Due to the multiple objectives, the

problem is quite complex. The exact formulation is outside the scope of this survey, but an overview will be provided here.

In order to transform the problem into a combinatorial optimization problem, the reservoir volume was divided into storage classes, effectively discretizing the state space. Therefore, a version of discrete-space ACO was formulated for this problem. A global pheromone update rule was used; only the global best ant partial solution deposited pheromone. This was chosen after iteration over ACO algorithm definition, and led to higher quality solutions being found.

The algorithm ran for two cases: a short-time horizon of 1 year and a long-time horizon of 36 years. The performance of the ACO algorithm was compared to another optimization algorithm, Genetic Algorithm (GA) [20]. In the case of the short-time horizon, the ACO algorithm performed similarly to the GA, and yielded a marginally higher average hydropwer output of 1511.83 MkWh (Million kilowatt hours), versus the GA's 1502.28 MkWh. In the long-time horizon case, the ACO algorithm significantly outperformed the GA, producing an average hydropwer output of 1522.68 MkWh, versus the GA's 966.63 MkWh. The data for the long-time horizon case is presented in Figure 4.

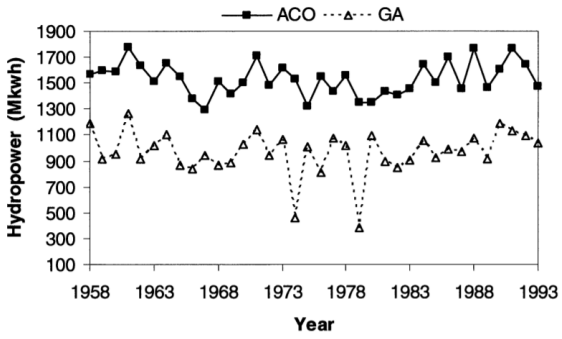

Figure 4: ACO vs. GA for the long-time horizon reservoir optimization problem. [19].

The authors propose the reason for ACO's outperformance over the GA is due to the ACO algorithm being better suited to deal with the large number of constraints and large number of time steps [19]. The ACO algorithm also ran considerably faster than the GA for the long-time horizon case [19]. The ACO algorithm's outperformance over the GA prove it to be a viable algorithm for large time scale planning of reservoir resource allocation.

3.3 Scrap Minimization Problem

This problem deals with minimizing scrap steel created in a bar mill. The bar mill can take a pieces of stock metal, or blooms, roll them into mother beams, and finally cut them into bars of specified length, section shape, and section size. Scrap is the leftover material after cutting the mother beams into bars. A way increase productivity in a bar mill is to cut several mother beams at once [18]. A schematic of this process is shown in Figure 5.

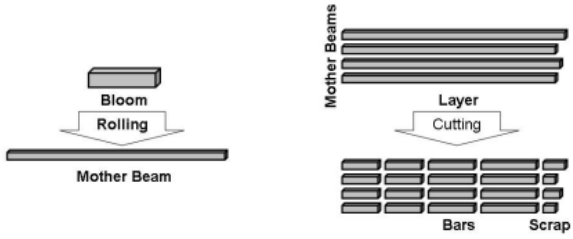

Figure 5: Schematic of bar mill processes [18].

The problem is the minimization of scrap created by cutting groups of mother beams, or layers, into bars, given bar length requirements, and bar number, or bundle, requirements. An example solution is shown in Figure 6.

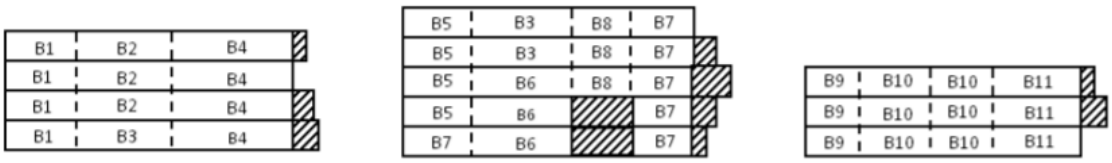

Figure 6: Example solution for 11 bundles (B1-B11) across three layers. [18].

The ACO algorithm to solve the scrap minimization problem presented by Diaz et al. in [18] is very similar to the ACO algorithm outlined previously. ConstructAntSolutions was implemented as follows:

$$p(b^*, l) = \frac{[\tau_{b^*,l}]^\alpha [\eta_{b^*,l}]^\beta}{\sum_{b \in C_l} [\tau_{b,l}]^\alpha [\eta_{b,l}]^\beta}, \; \eta_{b,l} = \frac{1}{1 + (N_l^p - N_{l,b}^p)}$$

Here, $p(b^*, l)$ is given as the probability of adding bar b^* to the current layer l. C_l is the set of feasible bundles for layer l. N_l^p is the number of feasible patters for layer l before the assignment, and $N_{l,b}^p$ is the number of feasible patterns after assigning bar b [18].

GlobalUpdatePheremomes was implemented by proposing the cost of solution Δ as:

$$C(\Delta) = S(\Delta) + kU(\Delta)$$

where $S(\Delta)$ is the scrap generated in the partial solution, $U(\Delta)$ is the unassigned material, and k is a constant. The function $g(s)$ within the GlobalUpdatePheromones function is then taken to be $\frac{1}{C(\Delta)}$.

The results of the ACO simulation were compared to real-time production data on the floor of the modelled bar mill, and it matched or outperformed the operator [18]. This proves that the ACO algorithm provides a viable solution to be used in practice. Future work into this problem may include characterizing the problem from blooms to bundles, instead of taking the number of layers and dimensions of each layer as a given to the problem [18].

4 Conclusions and Further Research

The ACO algorithm is a very versatile tool for optimization problems. It is highly problem dependent, and the formulation of the algorithm strongly affects the quality of solutions. This survey has presented several case studies and their formulations of the ACO algorithm for real-world engineering applications. The VRPTW can accurately model a large Swiss supermarket chain's logistics system, and this was solved using the ACO algorithm MACS-VRPTW by Rizzoli et al. They proved that the ACO solution outperforms a human planner's solution. Rizzoli et al. also created solutions with ACO algorithms that outperformed human planners for a VRPPDTW for a distribution company in Italy, a VRPTW for a logistics company in Italy, and a VRP for a oil distribution company in Switzerland [16]. Kumar and Reddy showed that the ACO algorithm can outperform the GA for the allocation of reservoir resources in India. Diaz et al. showed that the ACO algorithm can be used to improve the minimization of scrap steel creation in bar mills, comparing the solution against real-world bar mill data. Other examples of ACO algorithms being tested in the real-world include Shop Floor Scheduling [21] and Train Scheduling [22].

Further research into ACO algorithms for solving real-world problems may include more prevalent usage of continuous ACO. All of the problems outlined in this survey have been discrete-space problems, but many real-world engineering problems are continuous-space. A lot of interest has recently been focused on solving continuous-space optimization problems using continuous ACO algorithms [11, 12, 23, 24].

Although formulating problems that model real-world engineering problems often requires special care, and, if an ACO algorithm is chosen to find an optimal solution for this problem, formulating the ACO algorithm can get quite complicated. Despite this, this survey has shown several instances where ACO algorithms have been formulated to solve problems that closely model real-world engineering problems. The solutions reached in these cases have shown improvement over current best techniques [16, 18, 19]. ACO algorithm research is still very active, and it is likely that they will be used to solve a wider range of real-world engineering problems in the future.

References

[1] F. Yang, P. Wang, Y. Zhang, L. Zheng and J. Lu, "Survey of swarm intelligence optimization algorithms," *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 544-549, 2017.

[2] M. Dorigo, V Maniezzo, and A Colorini, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, Vol.26, No.1, pp. 1-13, 1996.

[3] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life Reviews*, Vol.2, pp. 353–373, 2005.

[4] D. Applegate, R.E. Bixby, V. Chvátal, W.J. Cook, "The Traveling Salesman Problem: A Computational Study," Princeton University Press, Princeton, 2006.

[5] J. Bell, P. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, Volume 18, Issue 1, pp. 41-48, 2004.

[6] L.M. Gambardella, M. Dorigo, "An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem." *INFORMS Journal on Computing* 12(3), pp.237-255, 2000.

[7] A. Shmygelska, R. Aguirre-Hernández, H.H. Hoos, "An Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem." In: Dorigo M., Di Caro G., Sampels M. (eds) *Ant Algorithms. ANTS 2002. Lecture Notes in Computer Science*, vol 2463. Springer, Berlin, Heidelberg, 2000.

[8] R. S. Parpinelli, H. S. Lopes and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 321-332, 2002.

[9] M. Dorigo, T. Stützle, "Ant Colony Optimization: Overview and Recent Advances." In: Gendreau M., Potvin JY. (eds) *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, vol 146. Springer, Boston, MA, 2010.

[10] M. Dorigo, G. Di Caro, "The Ant Colony Optimization meta-heuristic", In: D. Corne, M. Dorigo, F. Glover (eds) *New Ideas in Optimization*, McGraw Hill, London, pp. 11–32 1999.

[11] K. Socha, M. Dorigo, 2008. "Ant colony optimization for continuous domains," *European J. Oper. Res.* 185 (3), pp. 1155–1173, 2008.

[12] M. Omran, S. Al-Sharhan, "Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems," *Engineering Applications of Artificial Intelligence*, Volume 85, Pages 818-829, 2019.

[13] Toth, Paolo, and Daniele Vigo, eds. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.

[14] G. Dantzig, J. Ramser, "The Truck Dispatching Problem," *Management Science*, 6(1), pp. 80-91, 1959.

[15] L. Kovács, A. Agárdi, T. Bányai, "Fitness Landscape Analysis and Edge Weighting-Based Optimization of Vehicle Routing Problems," *Processes*, 8(11), pp. 1363, 2020.

[16] A. E. Rizzoli, R. Montemanni, E. Lucibello, L. M. Gambardella, "Ant colony optimization for real-world vehicle routing problems" *Swarm Intell*, 1(2), pp. 135–151, 2007.

[17] L. M. Gambardella, E. Taillard, G. Agazzi, "MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows." In D. Corne, M. Dorigo & F. Glover (Eds.), *New ideas in optimization*, London: McGraw–Hill, pp. 63-76, 1999.

[18] Diaz et al., "An ACO Algorithm to Solve an Extended Cutting Stock Problem for Scrap Minimization in a Bar Mill," In Dorigo et al. (eds), *Swarm Intelligence 9th International Conference, ANTS 2014*, Brussels, Belgium, pp. 13-24, 2014.

[19] D.N. Kumar, M.J Reddy, "Ant Colony Optimization for Multi-Purpose Reservoir Operation." *Water Resour Manage*, 20, pp. 879–898, 2006.

[20] X. Yang, "Chapter 6 - Genetic Algorithms," *Nature-Inspired Optimization Algorithms (Second Edition)*, Academic Press, pp. 91-100, 2021.

[21] A. Vogel, M. Fischer, H. Jähn, H., T. Teich, "Real-World Shop Floor Scheduling by Ant Colony Optimization," pp. 268-273, 2002.

[22] M. Reimann, J.E. Leal, "Single Line Train Scheduling with ACO." In: Middendorf M., Blum C. (eds) *Evolutionary Computation in Combinatorial Optimization. EvoCOP 2013. Lecture Notes in Computer Science*, vol 7832, Berlin, Heidelberg, 2013.

[23] Yan-jun, Li, and Wu Tie-jun. "An adaptive ant colony system algorithm for continuous-space optimization problems." Journal of Zhejiang University-Science A 4.1 (2003): 40-46.

[24] T. Liao, K. Socha, M. A. Montes de Oca, T. Stützle and M. Dorigo, "Ant Colony Optimization for Mixed-Variable Optimization Problems," in IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 503-518, Aug. 2014, doi: 10.1109/TEVC.2013.2281531.