5-2016

Ant Colony Optimization for Continuous Spaces

Rachel Findley
University of Arkansas, Fayetteville

Follow this and additional works at: https://scholarworks.uark.edu/csceuht

Part of the Artificial Intelligence and Robotics Commons, Software Engineering Commons, and the Theory and Algorithms Commons

Citation

This Thesis is brought to you for free and open access by the Computer Science and Computer Engineering at ScholarWorks@UARK. It has been accepted for inclusion in Computer Science and Computer Engineering Undergraduate Honors Theses by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu.
Ant Colony Optimization
for Continuous Spaces
Ant Colony Optimization
for Continuous Spaces

An Undergraduate Honors College Thesis

in the

Computer Science Department
College of Engineering
University of Arkansas
Fayetteville, AR

by

Rachel Findley
Abstract

Ant Colony Optimization (ACO) is an optimization algorithm designed to find semi-optimal solutions to Combinatorial Optimization Problems. The challenge of modifying this algorithm to effectively optimize over a continuous domain is one that has been tackled by several researchers. In this paper, ACO has been modified to use several variations of the algorithm for continuous spaces. An aspect of ACO which is crucial to its success when optimizing over a continuous space is choosing the appropriate object (solution component) out of an infinite set to add to the ant’s path. This step is highly important in shaping good solutions. Important modifications to this component were made in this research include using a Gaussian distribution as well as incorporating vector direction (Informative Pheromone) when evaluating the expected pheromone amount at any given candidate solution component. The results show that any variation of the algorithm herein which utilizes Informative Pheromone provides more accurate results than the others.
Acknowledgments

I would like to thank my research advisor, Dr. Gashler, for his unwavering patience, excellent guidance, and infectious enthusiasm during the completion of this research project. I would also like to thank Dr. Patitz for his practical advice and encouragement in the writing stages. Lastly, I would like to thank my husband for listening to me talk endlessly about ants when he mostly likely would have rather been doing other things.
# Contents

1 Introduction ................................................. 1

2 Background .................................................... 3
   2.1 Biological Basis .......................................... 3
   2.2 ACO Metaheuristic for Discrete Spaces ............... 4
      2.2.1 The Traveling Salesman Problem ................. 5
   2.3 ACO for Continuous Spaces ............................ 6

3 Implementation ............................................... 8
   3.1 Details on the Algorithm ............................... 8
   3.2 Terms ................................................. 11

4 Results ...................................................... 13
   4.1 Pheromone Concentration Visualization .............. 15
   4.2 Cost Comparison ....................................... 15

5 Conclusion .................................................. 19

References .................................................... 20
List of Figures

2.1 Bridge Experiment [2] ................................................................. 4
2.2 Traveling Salesman Problem [5] .................................................. 5
3.1 ACOV main algorithm ............................................................... 8
3.2 Construct Ant Solution Component for ACOV ............................ 9
4.1 Concentration of Pheromone and Ant Paths for the Simple Terrain after 700 iterations 14
4.2 Concentration of Pheromone and Ant Paths for the Bridge Terrain after 700 iterations 16
4.3 Result for a Simple Terrain Map ............................................... 17
4.4 Result for a Terrain Map inspired by the Bridge Problem ............... 17
Chapter 1

Introduction

Ant Colony Optimization (ACO) is an optimization algorithm inspired by the biological behavior of ants. In nature, ants initially search for resources by randomly searching the area around their nest. When a resource is discovered, the ant will return to the nest marking the ground with a substance called pheromone. The amount of pheromone left by an ant can correspond with the attractiveness of the resource. When other ants leave the nest to search for food, they will gravitate towards paths with greater pheromone densities. Over time, a particular path will become most popular amongst the ants in the population. This path is also observed to be rather efficient.

The algorithm inspired by this behavior was first proposed in the early nineties and a meta-heuristic was later written. The metaheuristic was found to be particularly successful in finding semi-optimal solutions to Combinatorial Optimization Problems. Combinatorial Optimization Problems (COPs) are those which consist of a finite set of objects which can be ordered in such a way as to create a low cost solution. Each object in the set is referred to as solution component. Solutions to COPs are essentially permutations of the objects which result in the lowest cost. Although the metaheuristic for ACO has been found successful in finding near-optimal solutions to COPs such as the Traveling Salesman Problem, many real life problems cannot be easily represented as a discrete space. The question then becomes how can the ACO metaheuristic be modified to find near-optimal solutions for continuous spaces as well. This research seeks to develop useful modifications to ACO which could allow it to be used in areas such as terrain map navigation and motion planning in robotics.

Although several solutions to ACO for continuous spaces have been created in the past, this research sets out to use some aspects of past solutions as well as new concepts in developing ACO for continuous spaces. Two of the main challenges this research addresses are:

1. How to determine viable solution components when there is an infinite number of available
solution components.

2. How to choose from viable solution components in such a way that more attractive components are recognized and can be chosen without quickly converging to a poor solution. Early convergence could limit exploration through the space, resulting in suboptimal solutions.

These challenges were addressed by using a Gaussian distribution as well as Informative Pheromone to estimate the expected pheromone at a candidate solution component. Other features developed in this research include using evolutionary algorithm approaches as well as directing solution components towards the destination when wandering out of bounds (referred to as Keep In Bounds). It was found that any variation on the algorithm developed in this research which utilized Informative Pheromone was most successful in finding a near-optimal path through the terrain map test case.

This paper discusses previous solutions to ACO for continuous spaces and describes the implementation of the ACO algorithm used in this research, ACO Variable (ACOV). Results are then be analyzed, comparing several variations of the algorithm.
Chapter 2

Background

2.1 Biological Basis

In nature, ants are remarkably successful in locating resources. From their nests, worker ants initially search the surrounding area randomly until a food source is found. After some time, ants march steadily along preferred paths. These paths are developed using a form of passive, non-symbolic communication called stygmetry. Through stygmetry, ants modify their environment by depositing varied concentrations of a substance called pheromone along paths that successfully lead to a desired resource. In some cases, ants deposit pheromone in proportion to the desirability of the food source discovered along the path. Once pheromone is deposited, all future ants searching in any given previously explored locus will make path decisions based on pheromone densities laid by other ants.

During early observations of ant behavior, Deneugbourg et al. designed an experiment in which ants were presented with two bridges that led to a food source. A visual representation of the experiment can be found in 2.1. In the case where the two bridges were equal, ants at first chose randomly between the two options. Over time, due to chance, one bridge accumulated more pheromone, becoming the preferred path. In the case that one bridge was significantly longer than the other, the ants which randomly chose the shorter bridge reached the food source faster, depositing pheromone as they returned to the colony. This resulted in other ants choosing the shorter bridge with greater probability due to its higher density of pheromone. These observations were the inspiration for Ant Colony Optimization (ACO) with the first algorithm, Ant System, being created in the early nineties.
2.2 ACO Metaheuristic for Discrete Spaces

Since the development of Ant System, many variations of ACO have been developed all with a common metaheuristic found to be quite successful in solving Combinatorial Optimization Problems (COP). These problems can be defined as $P = (S, f)$ where $S$ is a finite set of objects and $f$ is a function from $S \rightarrow R^+$ where $R$ is a positive cost value of the object $s \in S$. The object can often be a subset or represented as a graph. The best solution is the solution with a permuted subset of $S$ which results in the lowest summed cost of the object. [2] The ACO metaheuristic for solving COPs is broken into 4 major components: Initialization, Construct Ant Solution, Apply Local Search (aka Daemon Actions), and Global Pheromone Update. During Initialization, all parameters and pheromone values are initialized. Each artificial ant begins with an empty solution set. During Construct Ant Solution, a feasible solution component is chosen and incorporated into that ant’s partial solution. Solution components with greater attractiveness are chosen with greater probability. The probability of choosing a solution component is given by normalizing the value of pheromone at that solution component over the sum of all solution components in the partial solution. [9] The Apply Local Search step is also known as Daemon Actions and is an optional part of the metaheuristic which allows for local searching, problem-specific actions, and/or other actions that cannot be made by individual ants during each construction step. Lastly, Global Pheromone Update is the component which seeks to make good solutions appear particularly attractive. It has
two elements: Evaporate Pheromone and Deposit Pheromone. Pheromone evaporation allows for pheromone concentrations to dissipate over time, eliminating less attractive solutions. Pheromone Deposit determines and stores an appropriate quantity of pheromone for each solution component. The quantity of pheromone at each solution component is proportional to the attractiveness of that path overall. [5]

2.2.1 The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is an NP-Hard problem for which ACO has been successful in finding an approximate solution. In fact variants of ACO have been shown to have "world class performance" in solving TSP. TSP is a problem wherein there is a set of cities which a salesman must visit during his business trip. The cities are known as well as the distance between each. The goal for the salesman is to find the shortest path (or lowest cost path) which allows him to visit each city during his trip. In the ACO solution to this problem, the set of cities is represented as a fully connected graph where each city is a vertex and the edges are the connection between each city. An example of a solution to this problem is found in 2.2 Simulated artificial ants traverse through the graph, choosing vertices with greater pheromone with a higher probability and visiting each vertex only once. When an ant visits a vertex, pheromone is deposited along the edge between the current vertex and the previous vertex. Pheromone is modified over time by ants to make more
2.3 ACO for Continuous Spaces

ACO is most well-known for solving COPs which, by nature, consist of a finite set of objects which are permuted to create a solution. However, in real life, many problems cannot easily be represented in a discrete space. In fact, it is interesting to consider the fact that the biological inspiration for ACO is not set in a discrete space, but rather a continuous one. The question becomes how can the metaheuristic for ACO be used in an infinite search space? In fact, ACO has been implemented in a continuous space with several variations, some following the metaheuristic more closely than others.

One approach proposed by Socha and Dorigo claims to utilize the ACO metaheuristic without making "any major conceptual changes." This algorithm, ACO Continuous (ACOR), varies from traditional ACO in the way that each solution component is chosen. Where in ACO a finite set of variables is chosen with a discrete probability distribution, ACOR uses a continuous probability distribution, specifically a Gaussian probability density function. [9]

Another approach, Continuous Ant Colony Optimization (CACO) was developed by Bilchev and Parmee in the mid nineties. [1] This algorithm introduces the concept of a nest from which all ants begin their search. From the nest, a set of vectors are created, searching outward. At each iteration, ants choose different vectors with a certain probability and make random moves while following the selected path. At the end of each iteration, the vectors are updated to reflect improved paths. This algorithm differs from the original metaheuristic in that there is no incremental construction of steps.

Monmarche, Venturini, and Slimane propose yet another solution which does not follow the original ACO metaheuristic, but draws its inspiration from a particular species of ant called Pachycondyla Apicalis. This algorithm is called API, derived from Pachycondyla APIcalis. This algorithm begins by sending lone hunters to search globally and form promising hunting sites. Each ant performs local random exploration on their site to further determine its value. Ants also participate
in tandem running where two ants compare the desirability of their sites. The ant from the less promising site moves to the more promising site. In this way, more promising sites are, over time, given more attention. Another interesting feature of API is that periodically the nest is moved to a new location in space. This serves as a "restart operator" of sorts. Ants cooperate in order to choose the most promising new nest site. This helps avoid the formation of local minima. [8]

Yet another approach utilizes stygmergic information as well as direct communication between ants in the search space in order to derive a solution. Global ants use a genetic algorithm to search regions in the search space. Local ants set down pheromone within those regions to make particular paths more attractive to the entire colony. In this algorithm, Continuous Interacting Ant Colony (CIAC), proposed by Dreo and Siarry, there is no incremental construction of steps. [6]

Lastly, a more recent approach proposed by Liu, Dai and Gao uses a position distribution model to handle ant foraging within a continuous space. At the beginning, all ants are uniformly distributed throughout the space. Every point in space is assumed to be a resource with varying levels of usefulness. Ants determine the quality of their beginning position. Other ants perceive the spatial concentrations of pheromone throughout the space and migrate to higher concentration areas. [7]

The solution proposed in this paper, called ACO Variable (ACOV), also seeks to find an approach for continuous domains. It uses some similar techniques to those above such as combining with other evolutionary algorithmic models, utilizing a Gaussian distribution, and initializing all ants to start at a nest.
Chapter 3

Implementation

The well-known metaheuristic for ACO includes Initialization, Construct Ant Solution, Apply Local Search (aka Daemon Actions), and Global Pheromone Update. When applying this metaheuristic for a continuous space, the challenge becomes how to:

1. Determine viable solution components for each construction step when there is an infinite number of available solution components.

2. Choose from viable solution components in such a way that more attractive components are recognized and can be chosen with greater probability.

The solution tested in this work utilizes some similar concepts to those found in the background while also developing some of its own without quickly converging to a poor solution.

3.1 Details on the Algorithm

Figure 3.1 shows the pseudo-code for the ACO algorithm used in this research. The algorithm iterates for $N$ iterations, each time invoking several subroutines in order to find the solution component for the ant. It begins by calling the $\text{take\_a\_step()}$ function which represents the Construct Ant Solution component of the ACO metaheuristic. In this function, one solution component is added to the ant’s partial solution as shown in Figure 3.2. This is also where the main difference

```c
void Colony ()
{
    for upto iterations{
        take_a_step();
        if(has_reached_goal()){
            return_home();
        }
    }
}
```

Figure 3.1: ACOV main algorithm
between ACO for discrete spaces versus ACO for continuous spaces becomes apparent. In the discrete case, a solution component would be chosen from the set of unvisited solution components. Solution components with greater pheromone would be chosen with higher probability. In the continuous case, there are infinite solution components to choose from. This issue is addressed in this research by selecting a random set of candidate solution components (called steps in ACOV) from the space. From these candidate steps, those with the greatest expected pheromone are chosen with greater probability.

In order to calculate the expected pheromone, the `measurePheromone` function evaluates the expected quantity of pheromone at the candidate step and this value is used to determine the probability with which that candidate will be chosen. The function evaluates the quantity of pheromone expected by summing the expected pheromone density contributed by every other step found in the Pheromone List (the set of previously chosen steps for which pheromone amounts have already been assigned). This is always done by using a Gaussian distribution similar to the approach proposed by Sorcha and Dorigo which specifies that an ant must sample from a Gaussian distribution in order to select a viable solution component to add to the partial solution [9]. In this thesis, the Gaussian distribution is utilized as can be seen in Equation 3.1, where

- $p_i$ is the pheromone amount at the $i^{th}$ step in the Pheromone List
- $d$ is the squared distance between the candidate step and $p_i$, and
• $s$ is the number of pixels equivalent to one step.

\[
\sum_{i \in \Omega} p_i \cdot e^{-d/(s^2)}
\]  

(3.1)

The Gaussian distribution here gives the extent to which the pheromone amount at $p_i$ will be relevant to the amount of pheromone expected to be found at the candidate solution component $i$. This means that if a step in the Pheromone List is significantly far away from the candidate step, the pheromone amount will have little to no impact on the quantity of pheromone found at the candidate component solution. These values are summed over all steps in the Pheromone List giving the estimated pheromone value for the candidate step. In addition to using the Gaussian distribution, the code has an option to encode the direction of the step (Informative Pheromone) in the amount of pheromone. This is done by performing the dot product between the candidate step vector with all $i$ step vectors in the Pheromone List. This version of the equation is similar to Equation 3.1:

\[
\sum_{i \in \Omega} (dx \cdot dx_i + dy \cdot dy_i) \cdot p_i \cdot e^{-d/(s^2)}
\]  

(3.2)

The step vectors are evaluated when each candidate step is chosen by finding the vector between the previous step and the candidate step. The dot product of the step vectors is evaluated in order to determine if the candidate step is traveling in the general direction of the destination. If not, it will often be running in the opposite direction of the steps in the Pheromone List and will, therefore, have a smaller expected pheromone value. Once a candidate step is chosen, the $(x, y)$ location of the ant as well as the cost of the ant’s partial solution are updated.

Next, the algorithm evaluates if it has reached its goal. This evaluation is done in the `has_reached_goal()` function which returns true if the current step has arrived within a certain radius of the destination. This means a complete solution has been found and the Global Pheromone Update component of ACO must be performed. In this research, global pheromone update is implemented in two ways: Classic Selection or Evolutionary Selection. If the ant is utilizing Classic Selection, all the amount
of pheromone for each step in the Pheromone List will be evaporated by a constant amount. Also, if the size of the Pheromone List exceeds 10,000, then those steps which have a pheromone value below the average pheromone value for all steps in the Pheromone List will be removed. This serves to maintain a reasonably sized Pheromone List for run time purposes as well as to weed out less attractive steps from the Pheromone List. Lastly, the new set of steps found in the ant’s solution will be added to the Pheromone List. If Evolutionary Selection is being used, evaporation occurs only on that ant’s current path by the same evaporation rate. The Pheromone List length is checked here as well and reduced if necessary. The steps in the ant’s solution are added to the Pheromone List.

This algorithm also uses the nest concept developed in CACO [1]. When each ant is initialized, it begins at the nest location and after an ant finds a solution, it is reinitialized to start at the nest and search for a new solution. This base algorithm is otherwise the same as the traditional metaheuristic for ACO. An option to combine this base algorithm with other capabilities is built in by setting flags. These flags determine if Evolutionary or Classic Search is used to update pheromone, if Informative Pheromone or Classic Pheromone is used or if Keep Out of Bounds is applied. Descriptions of these flags as well as important terms are listed below.

3.2 Terms

The following presents a list of terms used to aid in the understanding of this paper’s proposed approach.

**Iterations:** The number of construction steps total for each ant over the course of the algorithm.

**Ant:** an agent searching for a solution (path) through the terrain map.

**Step:** Object which stores an x, y location on the terrain map as well as the amount of pheromone attributed to that location. This object also stores the vector information needed to determine the direction of the pheromone. Direction of the pheromone is the vector between the previous step and the current step.
**Path:** Set which includes all steps included in ant’s partial solution.

**Cost:** the summed pixel brightness value (lower RGB value) of all steps included in an ant’s solution.

**Pheromone List:** A set of steps which have been discovered throughout the search space. The pheromone list size is limited, pairing down steps with a pheromone amount below the average pheromone amount.

**Keep In Bounds (KB):** If a step is out of bounds, it is updated until it finds an inbounds x, y value. In finding this value, it is limited to angles close to the direct angle between itself and the destination.

**Classic Selection (CS):** When an ant arrives at the destination, pheromone is evaporated at a set rate for all steps in the Pheromone List. Then, the current ant’s path is added to the Pheromone List such that future ants can access this information when choosing a candidate pheromone object.

**Evolutionary Selection (ES):** Pheromone is evaporated differently from CS. ES compares the cost of the ant’s solution with the best cost solution so far. If the current ant’s solution is better than the best cost solution so far, the current ant’s solution becomes the best cost solution and pheromone is evaporated across the entire Pheromone List. However, if the current ant’s path is worse than the best path, the steps in that ant’s path are evaporated before being added to the Pheromone List.

**Classic Pheromone:** Pheromone for a candidate step is estimated using a Gaussian distribution alone.

**Informative Pheromone (IP):** Pheromone for a candidate step is estimated using a Gaussian distribution as well as the vector direction of the candidate step vector.
Chapter 4

Results

In order to test the proposed implementation of ACO for continuous domains, two terrain map problems were used. The terrain maps used were both greyscale images. The cost of the path was determined based on the brightness of the pixel of each solution component. This meant that paths composed of mainly black pixels are more cost effective than those composed of white or grey pixels. For each terrain, the nest is set at pixel (100, 100) and is shown as the red dot in the image. The destination is set at (400, 400) and is shown as the blue dot in the image. Each terrain is 500x500 pixels. The first terrain map used is referred to in this paper as Simple Terrain. This map has a band of black along the top of the image and the right side. The most cost effective solution would be the shortest path from the nest to the destination within the black territory. The second terrain map draws its inspiration from the bridge experiment used by Deneugbourg et al. when observing ant behavior. This terrain map has two entirely black paths leading near the destination with one path being longer than the other. The algorithm was tested on both maps over 700 iterations with the following combinations of algorithm components discussed in the implementation: Classic Selection (CS), Classic Selection and Keep In Bounds (CSKB), Classic Selection and Informative Pheromone (CSIP), Classic Selection, Keep in Bounds and Informative Pheromone (CSKBIP), Evolutionary Selection (ES), Evolutionary Selection and Keep In Bounds (ESKB), Evolutionary Selection and Informative Pheromone (ESIP), and Evolutionary Selection, Keep In Bounds and Informative Pheromone (ESKBIP). The best cost path at each iteration was compared among the 8 algorithm variations. Also, pheromone density of ant paths were drawn at every construction step to show how it evolved over time for each algorithm variation.
Figure 4.1: Concentration of Pheromone and Ant Paths for the Simple Terrain after 700 iterations
4.1 Pheromone Concentration Visualization

Figure 4.1 shows a visual representation of pheromone density and ant paths after 700 iterations for each algorithm variation tested in this research. The terrain map is also shown. The reader can estimate the most reasonable path by looking at the terrain map which would be the shortest length path in the black territory. The pheromone density and ant path images show the density in white and the path of different ants in green. Because this image is taken at iteration 700, different ants may only have partial solutions created explaining why some ants have not arrived near the destination yet. An informal visual interpretation of the data leads one to believe CSIP, CSKBIP, ESIP, and ESKBIP have the best pheromone densities with paths following along the density quite well.

The pheromone density and ant paths for the Bridge Terrain are shown in Figure 4.2. The Terrain Map for this test case shows two paths, one running right from the nest and down towards the destination (slightly longer path) and one running down from the nest and right towards the destination (shorter path). Both path options are entirely black. Therefore, it is clear that the path running down from the nest and right towards the destination should be the preferred path. A visual interpretation of the images shows that once again, CSIP, CSKBIP, ESIP, and ESKBIP show the best pheromone density and ant paths.

From these visual interpretations of the data, it is clear that Informative Pheromone is the common thread in the success of the algorithm. In terms of accuracy of the algorithm, the effects of Keep In Bounds appears negligible as does the effects of Evolutionary Selection.

4.2 Cost Comparison

Further investigation was performed by comparing the best cost path at each iteration for each algorithm variation. Figure 4.3 shows the cost comparison over iterations for the Simple Terrain map. This graph shows, once again, that those variations including the Informative Pheromone component were most cost effective over 700 iterations.
Figure 4.2: Concentration of Pheromone and Ant Paths for the Bridge Terrain after 700 iterations
Figure 4.3: Result for a Simple Terrain Map

Figure 4.4: Result for a Terrain Map inspired by the Bridge Problem
The cost comparison over iterations for the Bridge Terrain is found in Figure 4.4. This comparison shows, yet again, the variations with Informative Pheromone performing better overall. It is noted that the Informative Pheromone variations with Classic Selection seemed to converge more quickly to lower cost paths. However, these versions of the algorithm had significantly longer runtimes. The runtime of CSIP and CSKBIP was around 160 minutes each whereas the runtime of ESIP and ESKBIP was around 20 minutes each. The difference in the runtimes is due to the nature of Classic Selection vs. Evolutionary Selection. In CS, pheromone over the entire Pheromone List is evaporated each time an ant returns home. In ES, however, pheromone is only evaporated over the entire Pheromone List if the current ant’s path is lower cost than the lowest cost path found so far. When determining which algorithm works best, the runtime is important as a shorter runtime may make the algorithm more scalable, even if the results are somewhat less accurate.
Chapter 5

Conclusion

ACO is an algorithm that has been traditionally used to solve Combinatorial Optimization Problems. These problems find a solution by permuting the possible solution components to find a low-cost solution. By nature, such problems exist within a discrete space. The challenge to extend the original ACO metaheuristic to handle continuous spaces is one that has attracted a variety of research. Many solutions have been proposed, many breaking greatly from the original metaheuristic. The solution developed in this research does break from the original metaheuristic in some ways, but has been found to discover low cost paths in continuous spaces. This work chose to prioritize solutions that work well within reasonable runtimes over those that closely follow the original metaheuristic with success.

In this work, many variations of the algorithm were compared. The best variations were found to have one common vein: they all used Informative Pheromone. Informative Pheromone is used in calculating the pheromone amount at any given component solution. It uses vector math to calculate the direction of the vector at the given component solution. This information proved vital in accurately representing pheromone density throughout the space and led to the best solutions.

Although Classic Selection and Evolutionary Selection did not prove to be highly important in finding accurate solutions, Evolutionary Selection was shown to be much more efficient in terms of runtime. In the Bridge Terrain experiment, Classic Selection variations with Informative Pheromone performed slightly better than Evolutionary Selection variations with Informative pheromone; however, Evolutionary Selection variations had significantly better runtimes. In cases where accuracy is the most important factor, CSIP variations will prove to be the best algorithm choice; however in cases where runtime is more important, ESIP variations will be most effective.
REFERENCES


