# Ant Colony Optimization for Continuous Spaces

**by**

**Lachlan Kuhn**

**Bachelor of Engineering (Software)**

**Supervised by**
**Dr. Marcus Gallagher**

**A thesis submitted to**
**The Department of Information Technology and Electrical Engineering**
**The University of Queensland**

**October 2002**

Lachlan Kuhn
32 Athlone St
Tingalpa 4173
18th October, 2002

Professor Simon Kaplan
Head of School of Information Technology and Electrical Engineering
University of Queensland
St Lucia, QLD 4072

Dear Professor Kaplan,

In accordance with the requirements of the degree of Bachelor of Engineering in the division of Software Engineering, I present this thesis entitled '*Ant Colony Optimization for Continuous Spaces*'. The work was supervised by Dr. Marcus Gallagher.

I declare that all the work is my own, except where appropriate references have been cited, and that this work has not been previously submitted for assessment at this or any other institution.

Yours sincerely,

Lachlan Kuhn

## Acknowledgements

I would like to sincerely thank my supervisor, Dr Marcus Gallagher, for his direction and guidance throughout the course of this thesis. His suggestions and feedback during the year have been invaluable to me in completing the work presented here.

Also, I wish to thank the Romans for sanitation, medicine, education, wine, public order, irrigation, roads, a fresh water system and public health.

## Abstract

Ant Colony Optimisation is a recent algorithm used for solving optimisation problems. The algorithm is modelled on the behaviour of real ant colonies, and has traditionally been used exclusively for solving problems in the discrete domain. This thesis fully implements and evaluates a specialized version of Any Colony Optimisation capable of searching continuous spaces, and evaluates its performance under a range of conditions and test cases.

# Table of Contents

# List of Figures

# Chapter 1 : Introduction

The Ant Colony Optimization Algorithm is a relatively recent approach to solving optimization problems by simulating the behaviour of real ant colonies. The Ant Colony System (ACS) models the behaviour of ants, which are known to be able to find the shortest path from their nest to a food source. Although individual ants move in a quasi-random fashion, performing relatively simple tasks, the entire colony of ants can collectively accomplish sophisticated movement patterns. Ants accomplish this by depositing a substance called a *pheromone* as they move. This chemical trail can be detected by other ants, which are probabilistically more likely to follow a path rich in pheromone. This trail information can be utilised to adapt to sudden unexpected changes to the terrain, such as when an obstruction blocks a previously used part of the path (Figure 1.1).

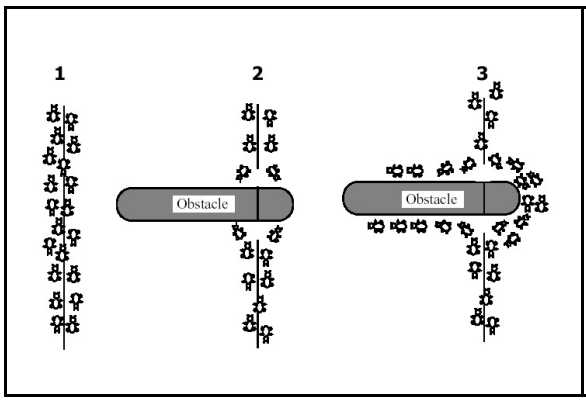



Figure 1.1 Ants moving between the nest and a food source are blocked by an obstacle.

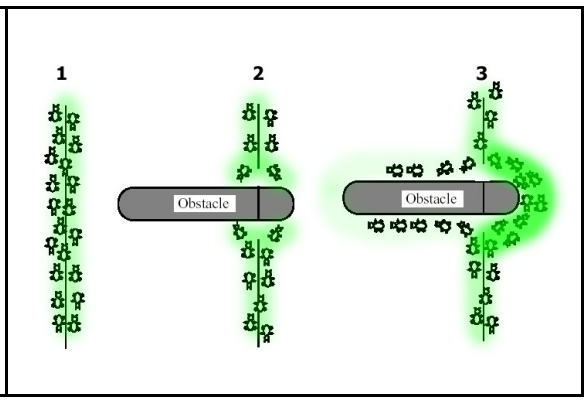



Figure 1.2 Pheromone build-up allows ants to re-establish the shortest path.

The shortest path around such an obstacle will be probabilistically chosen just as frequently as a longer path - however the pheromone trail will be more quickly reconstituted along the shorter path, as there are more ants moving this way per time unit (Figure 1.2). Since the ants are more inclined to choose a path with higher pheromone levels, the ants rapidly converge on the stronger pheromone trail, and thus divert more and more ants along the shorter path.

This particular behaviour of ant colonies has inspired the Ant Colony Optimization algorithm, in which a set of artificial ants co-operate to find solutions to a given optimization problem by depositing pheromone trails throughout the search space.

Existing implementations of the algorithm deal exclusively with discrete search spaces, and have been demonstrated to reliably and efficiently solve a variety of combinatorial optimization problems, such as the Travelling Salesman Problem (TSP).

## 1.1 The Travelling Salesman Problem

The TSP is extensively studied in literature dealing with optimisation, and is considered a standard test-bed for the evaluation of new algorithmic ideas – indeed, good performance for the TSP is considered reasonable proof of an algorithm's usefulness. The TSP is the problem of a salesman who wants to find the shortest possible trip through a set of cities on his tour of duty, visiting each and every city exactly once.

The problem space can essentially be viewed as a weighted graph (Figure 1.3), containing a set of nodes (cities). The objective is to find a minimal-length circuit of the graph. The ACS solves the problem by using a population of ants to construct tours by moving from city to city on the graph, traversing the edge that connect the cities, and applying pheromone trails to the arcs that connect the nodes. The pheromone strength associated with each arc is modified as the algorithm runs (Figure 1.4), and ants are able to utilize this pheromone trail information to help them build "good" solutions.

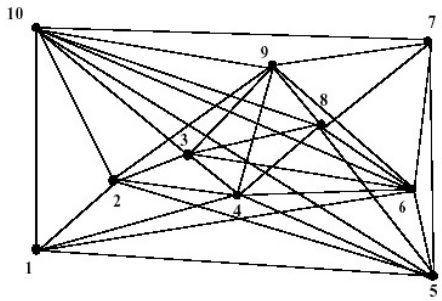



Figure 1.3 Trail distribution at the beginning of the search.

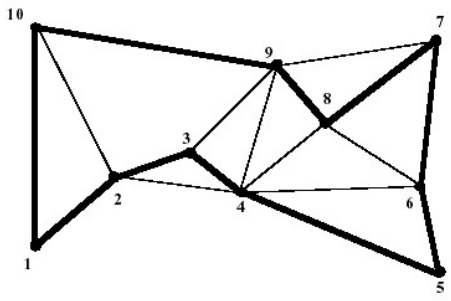



Figure 1.4 Trails distribution after 100 generations.

Ants exploring the TSP are probabilistically biased by the pheromone trails they encounter, and are more likely to select a path that is rich in pheromone. This information sharing leads to the construction of good solutions, and preferential exploitation of the more desirable search paths.

## 1.2 Applications areas of ACO

As mentioned, the ACO algorithm has been successfully applied to a number of different problems, the most famous example being the Travelling Salesman Problem. Other similar application examples include the Sequential Ordering Problem and the Vehicle Routing Problem. ACO has also been used to solve instances of the Quadratic Assignment Problem, as well as telecommunication network problems involving routing. All of these examples involve the ordering of a discrete number of *n* items, where the size of the search space is limited to *n!* permutations.

To date, almost all of the work in the area of ACO has concerned discrete optimization problems, and little work has considered the possibility of applying the ant colony metaphor to continuous space optimisation.

## 1.3 Thesis Objectives and Motivation

The principal goal of this thesis was to implement a specialised version of the ACO algorithm capable of searching continuous spaces. The motivation for implementing this new optimisation technique may not be immediately apparent (since there already exists numerous other well-known and highly refined algorithms for searching continuous spaces) - however, it is thought that using this “natural” metaphor will integrate several useful features into the search, similar to those features that make Simulated Annealing so unique. Features relevant to ACO include a highly efficient form of best-path exploitation (pheromone detection), and a sensible mechanism for exploration (probabilistic path selection). It will be shown that CACO quickly converges on “good” solutions – due in part to the fact that information gained during the search can help to reduce the time and effort expended in looking in unpromising regions, and guide subsequent search to areas likely to contain good solutions.

## 1.4 Document Overview

In the remainder of this thesis, the design, implementation, and evaluation of a Continuous Ant Colony Optimisation algorithm (CACO) is discussed in detail, with particular attention given to the way in which the various elements of the original ACO algorithm were adapted and modified to handle continuous spaces. This document is divided into several sections, as outlined below:

Chapter 2 : Background
This chapter offers a review of some of the work that is currently being conducted in the area of discrete Ant Colony Optimisation, and provides a more detailed look at the application of ACO to various optimisation problems. A basic overview of the algorithm is also discussed, explaining the procedure on which the system is modelled.

Chapter 3 : Algorithm Design
This section details the design choices that were made when implementing the continuous version of the algorithm, and describes in general terms how the sub-components of the system function together to achieve the desired emergent properties. This chapter also focuses on how elements of discrete ACO are adapted to work in the continuous domain, and introduces some new concepts that were specifically devised to overcome new problems that arose.

Chapter 4 : Algorithm Performance
In this chapter, the CACO algorithm is tested by evaluating its performance on a range of well-known continuous optimisation problems. CACO is also compared to some other stochastic optimisation methods. In addition to this, the effects of parameter setting are examined to investigate the algorithm dynamics.

Chapter 5 : Conclusion
This final chapter contains a summary of the work undertaken during the project, and addresses issues such as scaling the algorithm to handle higher-dimensional problems. Areas of possible future work in the CACO algorithm are also considered. This is followed by a bibliography.

# Chapter 2 : Background

Ant Colony Optimisation was first proposed in 1992 by Marco Dorigo, and in the decade since its introduction, a growing number of researchers have been involved in further developing it. Literature on the topic of Ant Colony Optimization is currently largely restricted to journals and research papers, as this is still an emerging algorithm that has not yet weaved its way into mainstream optimisation texts. The following paragraphs briefly outline some of the better papers that may be considered relevant to this project.

## 2.1 Previous Work

A good introductory paper [1] describing the Ant System has been written by Dorigo & Colorni, which describes in considerable detail the way a system of cooperating agents (ants) can be used for stochastic combinatorial optimization problems. This paper clearly outlines the principles of pheromone deposition and evaporation, and how the resultant properties of the system are a due to positive feedback from pheromone information. The paper also examines the results of several trials, comparing the effects of different run-time parameters on the performance of the algorithm, and showing optimal values for different scenarios. The Ant System algorithm is also compared against Simulated Annealing and Tabu Search for a set problem, showing how the Ant System performs compared to these other common optimisation methods.

Another journal publication by Dorigo [2] details the explicit application of ACO to problems like the TSP, and demonstrate fully how a discrete problem can be solved using this technique with “state-of-the-art performance”. In another recent paper [3], Dorigo and Mealeau showed that ACO is very much similar to the stochastic gradient descent algorithm. Dorigo, regarded as the pioneer in the field, has published dozens of other papers on the topic of ACO, ranging from introductory papers [4], to those discussing in depth the results of actual experiments [5], and papers discussing possible applications to other areas [6], a number of which are actually investigated by other researchers in subsequent papers [7].

Some more recent ACO adaptation ideas have proposed a hybrid version combining ACO with local search [8]. This new version is capable of outperforming all known

algorithms on a vast class of benchmark problems, including the Quadratic Assignment Problem and the Sequential Ordering Problem.

Information pertaining strictly to ACO for *continuous* design spaces is virtually non-existent, as this has not been attempted previously. Indeed, only one article [9] actually suggests an Ant Colony metaphor for continuous spaces, and it is the proposal in this paper that this thesis is based upon. The paper suggests a method of mapping the discrete nature of ACO onto a continuous search space by using a discrete (yet dynamic) structure – by representing a finite number of search directions as vectors. This idea is explained in more detail in Chapter 3.

## 2.2 Adaptation Problems

While the articles referred to above present a comprehensive description of how ACO can be used as an optimisation tool (and how effective it is), a substantial portion of the detail delivered in these papers has little relevance to this project. It is certainly true that there are a large number of common elements in both the discrete and continuous models of the ACO system – (for instance, pseudo-random localised search, pheromone trails and biased path selection), however there are also a number of significant differences. The lack of well-defined paths for transitions between “nodes” in the continuous domain means that movement between search states is now purely dependant on the nature of the heuristics that direct the ants through the search space. Similarly, placing and updating pheromone trails becomes another major issue. Even in Bilchev & Parmee’s paper [9] which proposes ACO for continuous spaces, neither of these issues is addressed. Resolution of this problem was a significant part of the algorithm design phase.

The specifics of the search technique for continuous ACO - in particular, exploration and pheromone trail placement - are quite similar to the approach adopted in the discrete version of the algorithm, however there are a number of important distinctions. These differences are largely accounted for by the fact that there is not a finite number of elements to be visited in the continuous domain, nor is there a finite number of paths between nodes. In theory, ants should be able to visit any of an effectively infinite number of points – and as a result, the task of recording pheromone trails in the system soon becomes an unmanageable task, unless a different approach is used.

## 2.3 ACO Structure

As mentioned previously, little work until now has considered extending the Ant Colony paradigm to continuous spaces. The approach for doing this outlined by Bilchev and Parmee [9] gave only a vague suggestion as to how this might be accomplished, and a number of major issues are ignored completely in their treatment. The design of the algorithm was therefore one of the most significant components of the work undertaken in this thesis.

The general structure of the ACO algorithm is as follows:

```
initialize colony()
evaluate(t)
while (not termination_condition){
      time ← time + 1
      add_trail(t)
      send_ants(t)
      evaluate(t)
      evaporate(t)
}
end
```

Each section of this pseudo-representation of the algorithm is discussed in detail in the following chapter.

The initialisation of the colony is discussed in Sections 3.1 and 3.5.
The laying of pheromone trails and path selection is discussed in Section 3.2.
Ant movement and solution convergence is discussed in Section 3.3.
Evaporation of pheromone trails is discussed in Section 3.4.
Algorithm termination conditions are discussed in Section 3.6.

In addition to these topics, some new ideas are introduced to the Ant Colony System aimed to help minimise the possibility of local optima trapping. The discussion attempts to steer away from implementation-specific details, and aims to provide a clear, descriptive explanation of the behaviour of the Continuous Ant Colony Optimisation Algorithm.

# Chapter 3 : Algorithm Design

## 3.1 Modelling the Nest Neighbourhood

The design of a specialised version of the Ant Colony Optimization algorithm for searching continuous (non-discrete) domains was a significant problem. Existing (discrete) versions of the algorithm are able to handle highly constrained order-based problems, however the ant colony metaphor cannot be directly applied to continuous spaces unless some kind of order-based representation is invented. To do this, it is necessary to first model a continuous nest neighbourhood with a discrete structure.

This is accomplished by representing a finite number of directions as vectors starting from a base point – in this scenario, referred to as the *nest* (Figure 3.1). The nest is a central point from which all ants begin their search. The vectors coming from the nest represent the directions that ants may choose to travel when they leave the nest. These vectors can be weighted with a value representing a pheromone concentration, which serves as an indicator of the success or "fitness" of ants that have chosen to explore that particular path.

Ants are capable of updating these vector weightings (or pheromone trails), so that the values correspond to the likelihood of finding a good solution in the region described by the vector. This enables ants to determine the more promising areas of the search space by comparing pheromone levels of available direction vectors. The points described by the vectors are also able to evolve over time according to the ants' fitness (Figure 3.2). This effectively allows pheromone trails to adapt to the terrain, enabling ants to find increasingly better solutions.

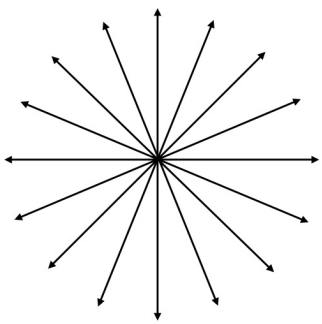


Figure 3.1 The nest (centre point), with a number of direction vectors indicating the initial directions ants will travel.

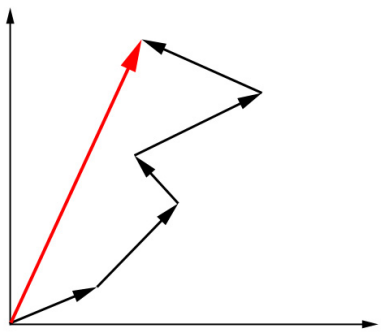


Figure 3.2 Evolution of a direction vector, showing how improvements in an ants' fitness results in vector displacement.

The use of direction vectors and pheromone concentrations to store information about the suitability of solutions that ants have encountered is useful for a number of reasons. The main advantage is that areas containing poor or unpromising solutions are effectively assigned lower priorities, allowing more computational resources to be spent looking for better solutions in areas already known to contain good solutions. This means that although there may be an increasingly small chance than ants will select a low pheromone-level path for inspection, exploration of these areas is not expressly forbidden, just increasingly unlikely as the algorithm progresses. This is because the selection of direction vectors to explore is probabilistically determined, and the best paths are not guaranteed selection each time.

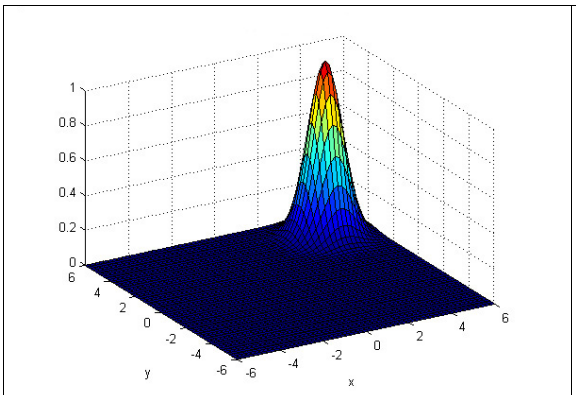



Figure 3.3 A simple terrain containing one global maximum.

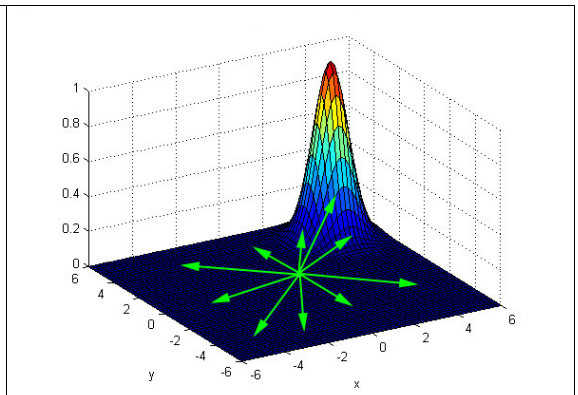



Figure 3.4 Ants are scattered outwards from the nest.

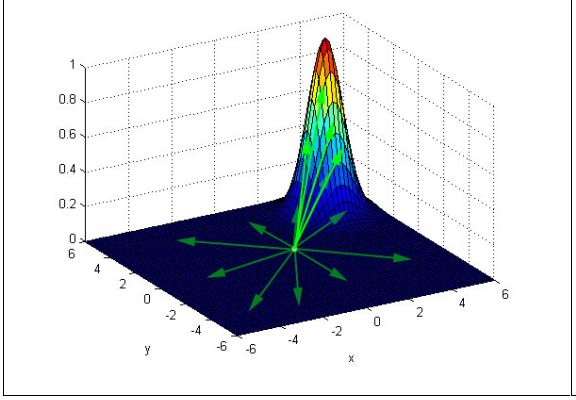



Figure 3.5 Pheromone vectors gradually evolve, encouraging ants to explore promising regions.

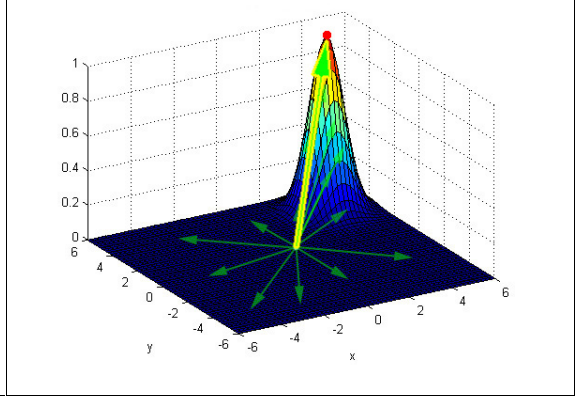



Figure 3.6 The optimal global solution is discovered.

It is necessary to adopt this approach of modelling the nest neighbourhood with a discrete number of vectors due to the fact that unmediated pheromone trail laying in a continuous search-scape would be unmanageable, owing to the fact that there are no well-defined “nodes” or “edges”. If trail information was retained for every trip

made by every ant over every generation, the data-structure needed to store this information would become unmanageably large, especially for problems containing any more than a few variables.

### 3.2 Direction Vector Selection

When an ant leaves the nest, it must make a decision about which way to travel. This decision is influenced by the pheromone trails leading out from the nest, which are represented by weighted directional vectors.

To implement the probabilistic nature of the direction vector selection mechanism, a special structure was devised, modelled partially on a roulette wheel. A set of *n* initially equally-sized slots is set up, representing the *n* direction vectors in the system (Figure 3.7). These slots each represent a particular range of numbers. Next, a random number is generated which will correspond to a particular slot. This process of random number generation and slot matching is analogous to the spinning of a roulette wheel.

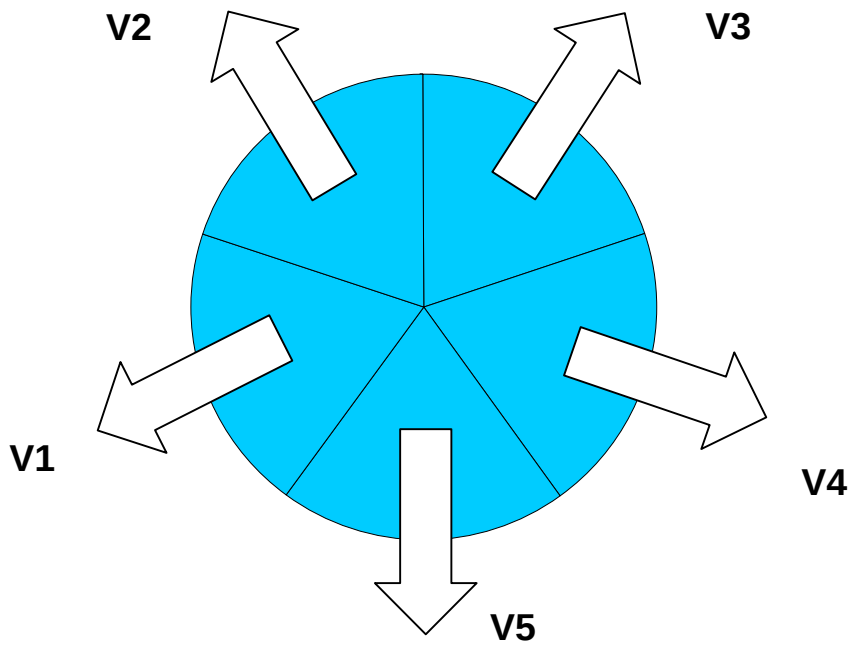



Figure 3.7 Five equally weighted direction vectors, each with an equal probability of selection for future exploration.

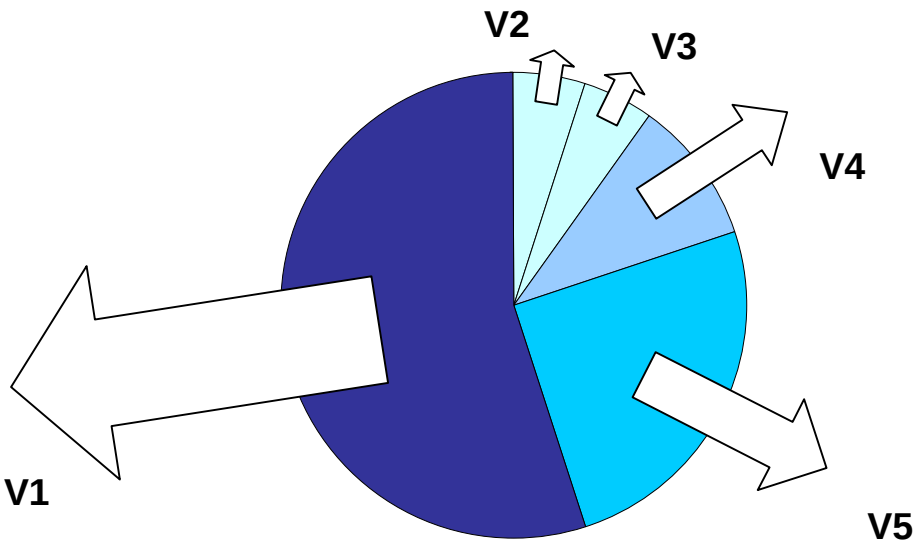



Figure 3.8 Pheromone build-up now favours vector V1, which points to a very promising region. Vector V5 remains unchanged, whereas V4, V3 and V2 are areas which probably should be avoided.

As ants begin to discover the characteristics of the terrain they are exploring, they will update the pheromone concentration values associated with the vector they are exploring. A pheromone level that is significantly higher than other values in the system indicates that ants exploring that vector are having some degree of success, having found a region that possibly warrants more attention. The system will now assign the vector in question a wider range of numbers, meaning that there is now a greater probability of selecting the vector when another random number is generated.

Correspondingly, vectors showing less promise have their range restricted, meaning that there is a smaller chance that the vector will be selected for exploration. This allows the majority of ants to investigate the more promising regions, while minimizing the effort expended in exploring areas not likely to contain good solutions. In effect, shrinking the size of unpromising vector slots helps to steer the ants away from areas that seem to yield poor solutions.

To help minimize the chances of ants converging prematurely on local optima, a mechanism was put in place to ensure that the probability of selecting any given vector can never fall to zero. This also ensures that certain areas of the search space are not completely impossible to reach; in other words, *"no search is impossible"*.

### 3.3 Ant Movement

In the Ant Colony system, ants are simple agents with limited scope, sent out to explore a search space. An individual ant is effectively blind, and cannot see what other ants are doing. Ants can, however, make informed decisions about the paths they choose to explore by utilizing pheromone concentration information.

When an ant selects a particular direction in which to travel, it is instantaneously transported to the location of the best previous ant to have chosen that same direction. From this point, the ant makes a random movement to some new location, within some maximum search radius defined for that stage of the search. This radius gradually contracts throughout the duration of the search, so that the ants can converge on a solution with a high degree of accuracy.

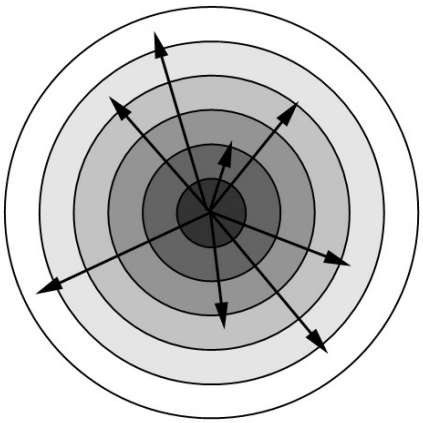


Figure 3.9 Initial model for ant displacement, with denser probability regions closer to the current position of the ant.

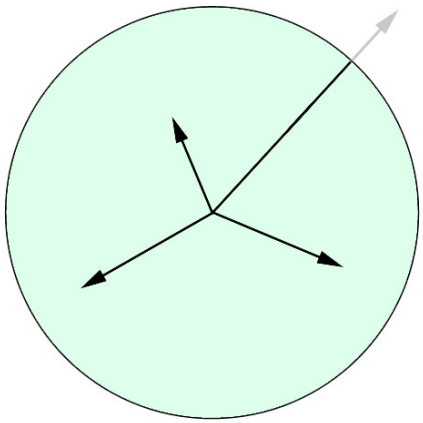


Figure 3.10 Later model for ant displacement, with all areas within the maximum exploration radius allowed. Ants may only move further than this under special circumstances (See section 3.4).

The initial implementation used for the ant movement procedure assigned regions closest to the ant's current location with higher probabilities (Figure 3.9). This method used a Gaussian distribution approach to determine where the ant would move, with small displacements favoured over larger ones. The ants were still constrained by a search radius, however ants were generally unlikely to move the maximum distance available to them. This was found to unnecessarily restrain the ants, and resulted in an effective radius significantly less than the maximum. To combat this problem, the probability regions were removed completely from the algorithm, and replaced with a simpler "pure random positioning" system (Figure 3.10), in which all areas within the search radius are equally likely to be selected, provided that they did not violate boundary conditions for the search.

When an ant moves to a new location, it next evaluates its fitness. The fitness of an ant is a measure of the “goodness” of the solution that the ants’ position represents. If the ant reports a fitness greater than the best current fitness recorded for the current direction vector being explored, the vector position is updated, and the vector’s pheromone concentration will be appropriately incremented. If the new position is not an improvement, it is disregarded, and the ant defaults back to the last known best position in the following cycle.

As the algorithm progresses and pheromone concentrations begin to reveal the promising regions of the terrain, the search radius for individual ant movement begins to contract. This gradual reduction in the range available to ants allows the algorithm to home in on a very precise solution. The constriction of the radius effectively gives the ants a decreasing amount of space to move, making exploration of new areas less of a priority, and forcing ants to converge on the local optimum they are exploring.

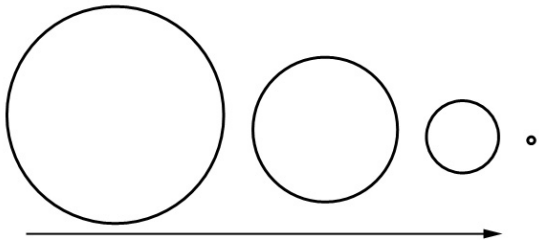


Figure 3.11 The search radius gradually diminishes, allowing the ants to converge on a solution.

Various methods of radius reduction were considered, with three different options actually being implemented. The first method uses a fixed decremental amount, which is subtracted from the search radius after each ant in the current generation has moved. This results in a constant rate of restriction (Figure 3.12), with each generation having a proportionally smaller search area than the last. The second method uses a fixed radius restriction factor (some value less than 1), by which the radius is multiplied after each generation. This option allows ants to narrow down their search more rapidly initially, and to spend longer in detailed local solution exploration (Figure 3.13). The third method uses a similar approach, but instead is more biased towards global exploration. This approach leaves the ants with more time to investigate the properties of the entire search scape, and undergoes rapid convergence on solutions towards the end of the algorithm’s life (Figure 3.14).

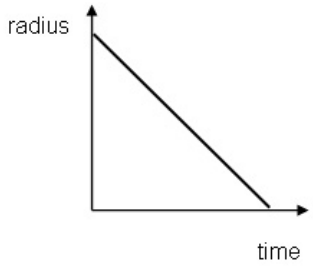



Figure 3.12 Constant rate of radius reduction.

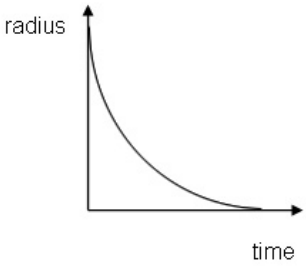



Figure 3.13 Exponential decay of radius, model #1.

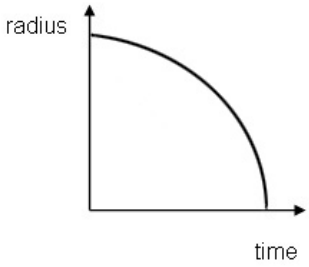



Figure 3.14 Exponential decay of radius, model #2.

For all of these models, the value (or rate) of radius reduction is typically set very small. This is necessary to ensure that the algorithm does not terminate before the ants are given sufficient time to explore the terrain, and pheromone trails are given ample opportunity to accumulate. Setting the values extremely small also ensures that ants do not prematurely converge on solutions that may not be global optimums. If the radius restricts too quickly, a situation may arise in which an ant no longer has the range required to escape the current local optima. This is obviously undesirable behaviour, and so appropriately small values are needed to prevent this from happening.

Other techniques for radius reduction were considered as well, including a number of more complicated shrinkage patterns (such as the one shown in Figure 3.16). However, this was found to introduce unnecessary complexity to the algorithm, as determining at what point rates should change resulted in more variables to control. The choice of which technique to use was left as a parameter to investigate during the evaluation phase of the algorithm (Chapter 4).

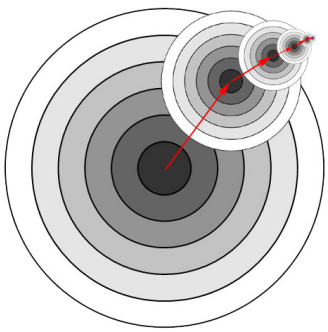


Figure 3.15 The gradual restriction of the search radius allows ants to “home in” on solutions with an increasing degree of precision.

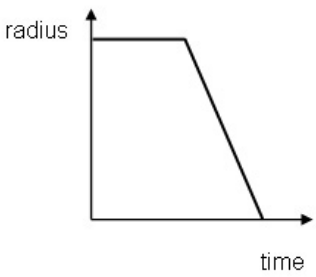



Figure 3.16 A more complex radius restriction model, in which the radius only begins to contract after a certain number of generations.

## 3.4 Food Source Exhaustion

Ants that do not result in improvement do not participate in the trail laying process, and the reverse process (trail evaporation) helps divert attention away from the area in question. As mentioned previously, trail evaporation cannot cause a vector to disappear entirely, as there is a minimum value determining the smallest possible amount of pheromone a trail can contain. This prevents the probability of selecting an unpromising vector ever falling completely to zero, and encourages ants to explore other more promising vectors.

When an ant is stuck in a region yielding no good results, the pheromone concentration of the vector being explored will begin to fall. Obviously, if an ant cannot reach a better solution, the pheromone trail will continue to evaporate until there is a very small probability than subsequent ants will bother looking in that direction. When this situation arises (analogous to food source exhaustion in a real ant colony environment), ants have to either pick another path to explore, or risk starvation. A mechanism was built into the algorithm to model this scenario, in which ants inspecting paths with minimal pheromone concentrations are marked “starving”, and are therefore granted special manoeuvrability powers. These ants are able to exceed the maximum search radius defined for that particular moment of the search, allowing them to (hopefully) break out of their area of low-performance, and encounter greener pastures.

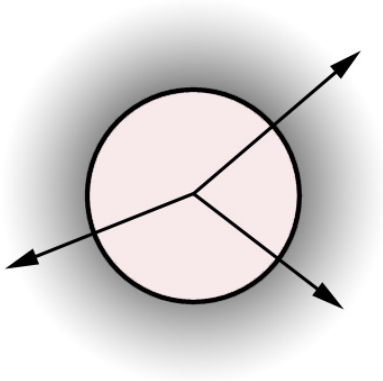
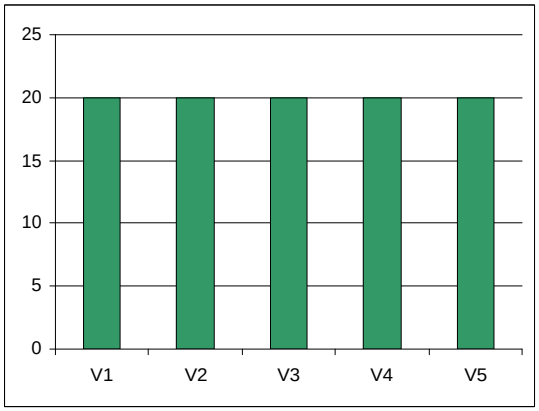


Figure 3.17 Ants that are classified “starving” are able to escape the “event horizon” defined by the search radius, and reach new territory.

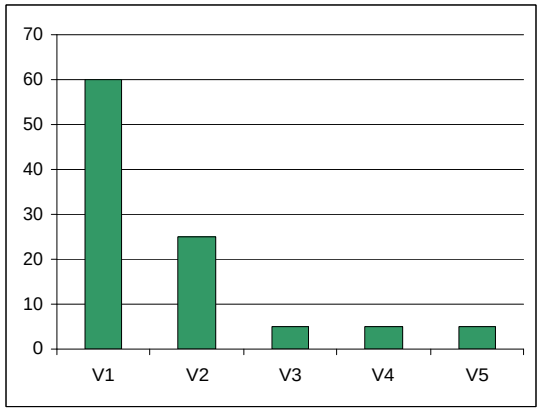
These “desperate” ants are therefore capable of escaping local optima, provided that sufficiently better solutions are being discovered elsewhere.

Figure 3.18



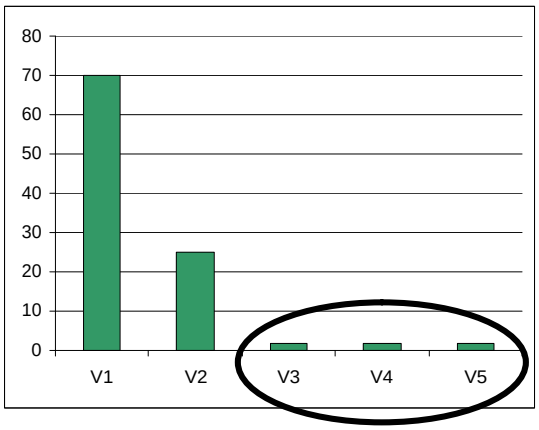

Five vectors, each with exactly equal pheromone concentrations.

Figure 3.19



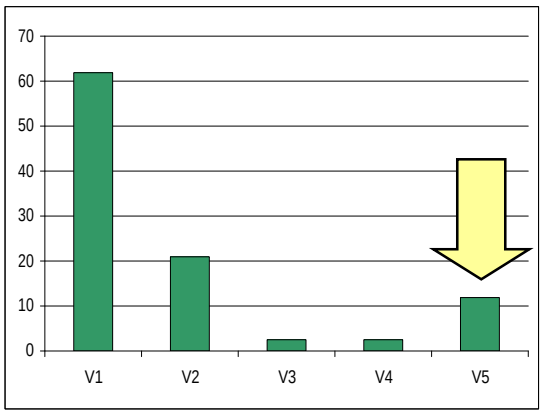

Vectors V1 and V2 are both yielding good results, whereas V3-5 are doing poorly.

Figure 3.20




Vectors V3-5 have reached their minimum possible values.

Figure 3.21




A starving ant exploring V5 has escaped the region, and found a better solution.

The diagrams above illustrate the concept of food source exhaustion and “starving” ants. In Figure 3.18, five vectors are shown, each with an equal concentration of pheromone – a condition that would be present in the start state. Figure 3.19 shows the system some time later, in which vectors V1 and V2 have shown significant improvement, whereas the remaining three vectors have not. Figure 3.20 illustrates the point at which vectors V3-5 have undergone trail evaporation to the point where they have reached their minimum allowable values. From this point onwards, any ant to pick one of these vectors would be granted Super Ant powers, and allowed to move extra large distances due to the desperation of the situation. In Figure 3.21, Vector V5 has improved substantially, indicating that one of the starving ants successfully jumped to a better position. When a vector recovers sufficiently, ants are de-classified as starving, and normal radius-bound exploration continues.

## 3.5 Initial Nest Configuration

The location of the nest is of particular importance to the performance of the algorithm. The nest is the point from which all the ants first move, and located by default in the centre of the search space. This position is calculated by examining the boundaries of the search, and initialising the initial ant and vector co-ordinates to this point.

There are, however, some circumstances where initialising the nest to the centre of the search space may not be appropriate. Consider the simple parabolic dish, shown in Figure 3.22. If the objective function is being maximized, there would be no problem, however if we were trying to minimize the function, the ants would already be sitting on the global solution by default (Figure 3.23). While this might be considered extraordinarily good luck, it provides no useful information about the performance of the algorithm, and for problems where the nest happens to fall in exactly at the global optimum, the location can be manually over-ridden.

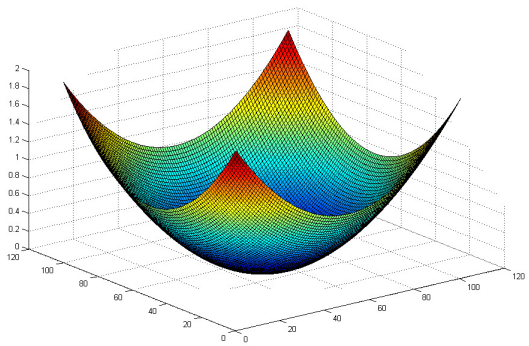


Figure 3.22 A parabolic bowl.

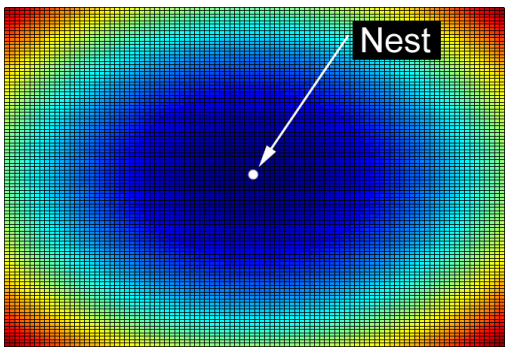



Figure 3.23 The nest is initialised to the centre of the search space by default.

Determining an alternate position for the nest can be done a number of different ways, including random assignment, or placing the nest a “sufficiently difficult” distance away. Of course, if the algorithm was being used to solve a real-world problem, we would not complain if the ants chanced to be initialised close to a solution. However, when benchmarking the performance of the Ant Colony algorithm against other well-known algorithms, it would be improper to claim the performance was superior to another just because the test-suite of functions all contained solutions at (or very near) the starting point of the algorithm. This issue is addressed in more detail in Chapter 4.

## 3.6 Algorithm Termination Conditions

An important consideration for the design of the algorithm was to decide the conditions for which the algorithm is said to have “finished”. Because the circumstances under which the algorithm might be used are many and varied, there exist a number of different termination scenarios. These include:

A maximum number of evaluations (EMAX) have been performed
A maximum number of generations (GMAX) have been executed
The search radius (R) has shrunk to zero
The solution is sufficiently close to a known global solution

The first two conditions are likely to be used when comparing the performance of the Ant Colony algorithm to the performance of some other well-known search technique. In this case, the best solution for some given number of fitness function evaluations would be recorded over a number of trials for the two algorithms being compared. GMAX, the number of generations, is an ACO-specific parameter which depends on the number of ants populating the search (GMAX / EMAX = No. of Ants). As is the case with EMAX, terminating the algorithm after some fixed number of generations would most often be used to gain an understanding of how the algorithm performs with time.

Another possible reason to terminate the search would be when the search radius has shrunk to zero. This signifies that the ants can progress no further, since a zero-sized radius implies a zero-area search zone. The r = 0 condition does, in effect, bind the algorithm to some fixed number of evaluations (since the radius decreases with respect to time only), however if the radius has been allowed to decay gradually, we can be far more certain that the ants have discovered a solution that is locally precise.

Finally, if a function is being tested for which we already know the global solution(s), we can choose to stop the algorithm as soon as it discovers an answer that is within a given acceptable range of the global answer. This allows us to determine exactly how many cost-function evaluations were needed before the algorithm successfully identified a solution with an acceptable proximity of the global optimum.

## 3.7 Implementation Environment

The Continuous Ant Colony Optimisation (CACO) algorithm was coded in ANSI C, and was successfully compiled and tested in OpenBSD 3.0, SunOS 5.8 (Sparc) and the Windows32 environments. Because the code was fully portable between a number of different platforms, evaluation trials were carried out on several different systems. Testing focussed strictly on cost-function evaluation comparisons, and so CPU and memory issues that would have affected time-based comparisons were avoided completely.

# Chapter 4 : Algorithm Performance

## 4.1 Initial Algorithm Verification

Before testing the algorithm on complex problems, it was first necessary to validate that the algorithm behaved as was originally intended. To achieve this, the algorithm was tested on a number of simple problems, including the inclined plane (Figure 4.1) and the parabolic curve.

In order to observe the movement of the ants as the algorithm progressed, ant position and fitness values were recorded for each generation. Plotting this information for selected generations gave a clear visual description of how the ants responded to the subtleties of the environment as it was explored. Figure 4.2 shows the ants after the first generation, scattered approximately evenly around the nest, while Figures 4.3 and 4.4 show the migration of ants up the slope over subsequent generations as the algorithm progresses.

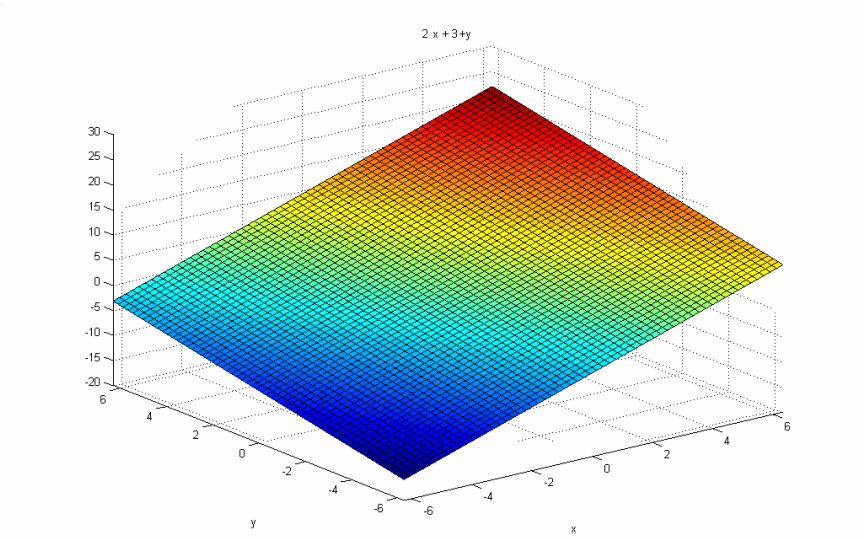



Figure 4.1 An inclined plane, described by the equation **z = 2x + y + 3**

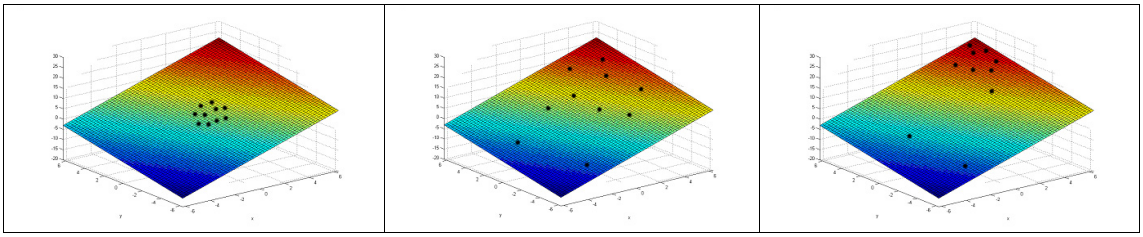


Figure 4.2 Generation = 1. Ants have just left the nest.

Figure 4.3 Generation = 20. The majority of ants have migrated to the top end of the plane.

Figure 4.4 Generation = 50.

Following this initial verification of the algorithm's behaviour, tests were conducted on some more complex surfaces. Direction vector information was gathered when the algorithm had undergone a sufficient number of generations for the ants to converge on solutions.

The information gathered from these trials indicated that the ants were indeed capable of simultaneously exploring multiple promising regions in parallel. The surface $F(x, y) = x^3 + 3y^2$ shown in Figure 4.6 contains 2 global maximums, both of which were discovered by the ants. Similarly, for the surface $F(x, y) = x2 + y^2$ (shown in Figure 4.7), all 4 solutions were discovered.

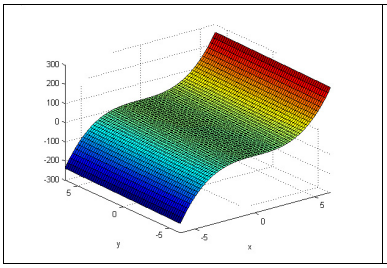


Figure 4.5 A surface with a distribution of optimal solutions lying on the extremity of the search.

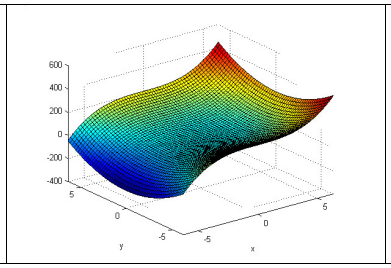


Figure 4.6 Surface with two optimal solutions.

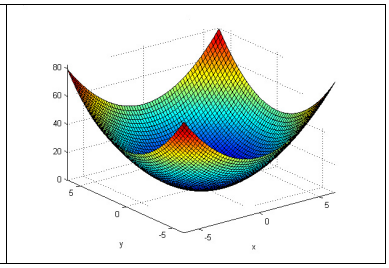


Figure 4.7 Surface with 4 optimal solutions,

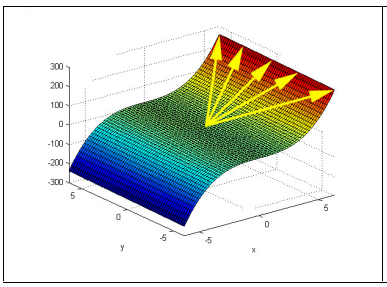


Figure 4.8 Ants discover a number of solutions along the border of the search space.

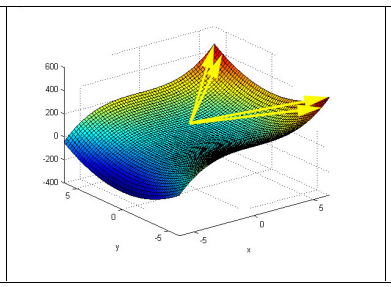


Figure 4.9 Ants converge on the two corners. Arrows show direction vectors.

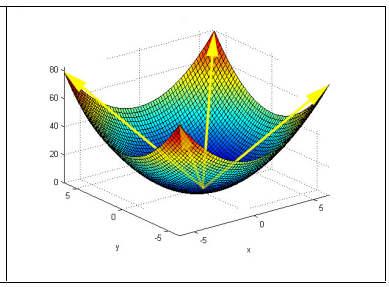


Figure 4.10 Ants discover all 4 solutions in all 4 corners.

A unique feature of the Any Colony algorithm is that these solutions are developed in parallel, due to the fact that ACO is a population-based optimisation technique. The information sharing amongst ants exploring the system allows better solutions to be attained faster, but also allows independent local search to progress virtually unhindered. This gives ACO the ability to locate multiple optimums (if more than one exists); a feature that many other optimisation methods can only achieve by running for several iterations.

## 4.2 Test Function Suite

To test the performance of the CACO algorithm more rigorously, a standard set of test functions was used to evaluate it. The test suite consisted of 5 well-known optimisation functions [REFERENCE], suitable for optimization in 1 or more dimensions. These functions range from reasonably simple to relatively difficult, and the performance of CACO for each of the 5 test functions is discussed in the next section.

In order to obtain a valid comparison of CACO across all of the test cases, a number of parameters within the algorithm had to be set to pre-determined values prior to execution. A brief discussion of these parameters is as follows:

**A - Number of ants** : The number of ants in the system is the number of agents sent out to explore the direction vectors. This value was set to 50 to allow a reasonably-sized population.

**V - Number of direction vectors** : This corresponds to the number of regions of interest that ants may be exploring in the system at any given time. The quantity V also determines the number of pheromone trails the system needs to track. This value was set to 12.

**P - Pheromone Update Rate :** The maximum rate at which pheromone concentrations are able to accumulate. The existing pheromone concentration for each vector (some value between 0 .. 1) is multiplied by P to obtain the new value. For these trials P was set to 0.05.

**E - Evaporation Rate :** The rate at which pheromone trails evaporate. This value was set to 0.95. If a vector has not undergone any improvement in the current cycle, the existing pheromone concentration is multiplied by E.

**R - Initial Search Radius** : The initial scope (maximum single displacement) of the ants is defined as some proportion of the diagonal distance across the search space. For these trials, this value was set to 10, meaning that ants could not move any further than $1/10^{th}$ of the total distance across the search area in any single move.

**F - Radius Restriction Factor** : The rate at which the search radius R is being decreased. The radius decreases at the rate of R * F. For these trials, F was set to 0.999, allowing a sufficiently gradual rate of restriction.

**A - Accepted Range :** This value specifies the acceptable “closeness” a solution must be to the (known) global optimum before it will be considered good enough to allow algorithm termination. For these trials, A was set to 0.001, where the global solution was 0.0. This means that any solution evaluating to below 0.001 was considered “optimal”.

These values were chosen somewhat arbitrarily, but were set to values that seemed realistically appropriate at the time of implementation. Setting these values prior to running a primary batch of trials was essentially just a formality to ensure control parameters were kept consistent between test cases.

When running CACO for each test function, statistical data was generated for 100 trials (executions). In each trial, information regarding the total number of function evaluations required to find the “optimal” solution was gathered, as well as the best solution the system held at the time of termination. In the next section, CACO is compared to blind (random) search for each of the 5 test functions, and the results are discussed for each.

### 4.2.1 Rosenbrock

The Rosenbrock function is described by the expression –

$$F_1(\vec{x}) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right] \qquad \text{for the range } \left[-30,30\right]^n,\ n \geq 2$$

The global solution to this function is defined as -

$$F_1(\vec{x}^*) = 0 \qquad \vec{x}^* = \{1,1,1,\ldots,1\}$$

In other words, in the 2-dimensional problem being considered for this trial, the global solution lies at the point (1, 1), where the function evaluates to exactly 0.0. As can be seen in the figure below, the surface described by the function resembles a parabolic sheet.

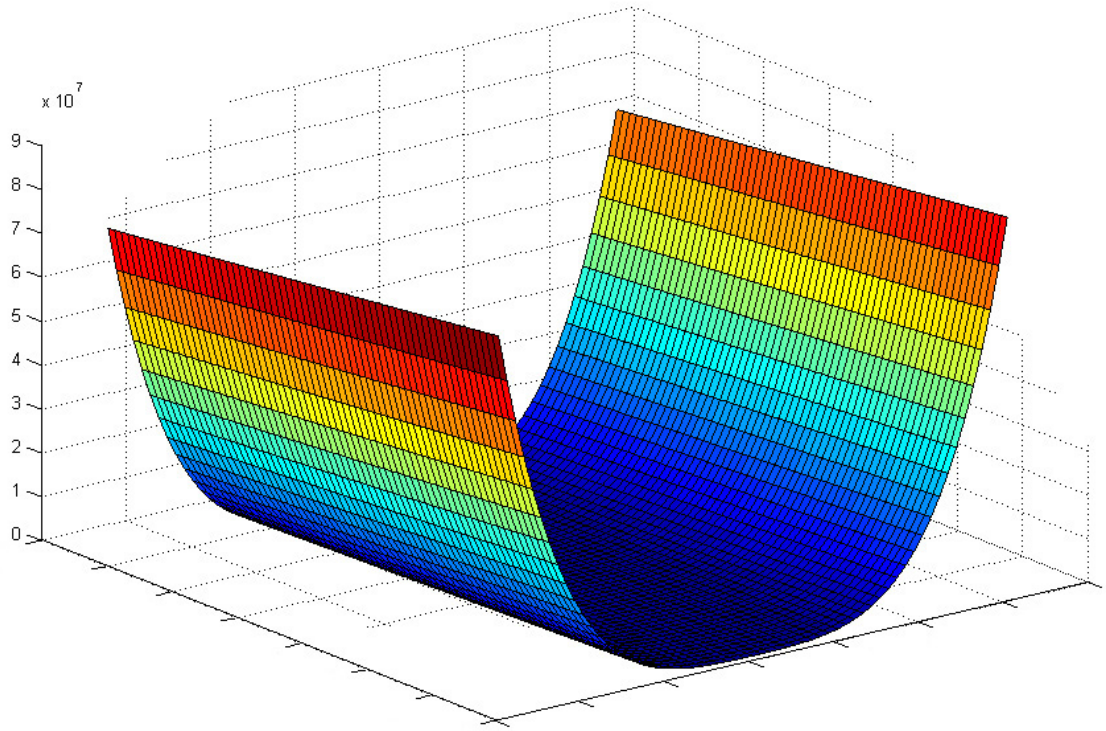



Figure 4.11 A plot of the Rodenbrock function

Trial results for the Rosenbrock function showed that CACO out-performed blind search by several orders of magnitude, averaging only ~18300 cost function evaluations to find the optimal solution (Figure 4.12), compared to the ~11.9 million evaluations taken by blind search (Figure 4.13).

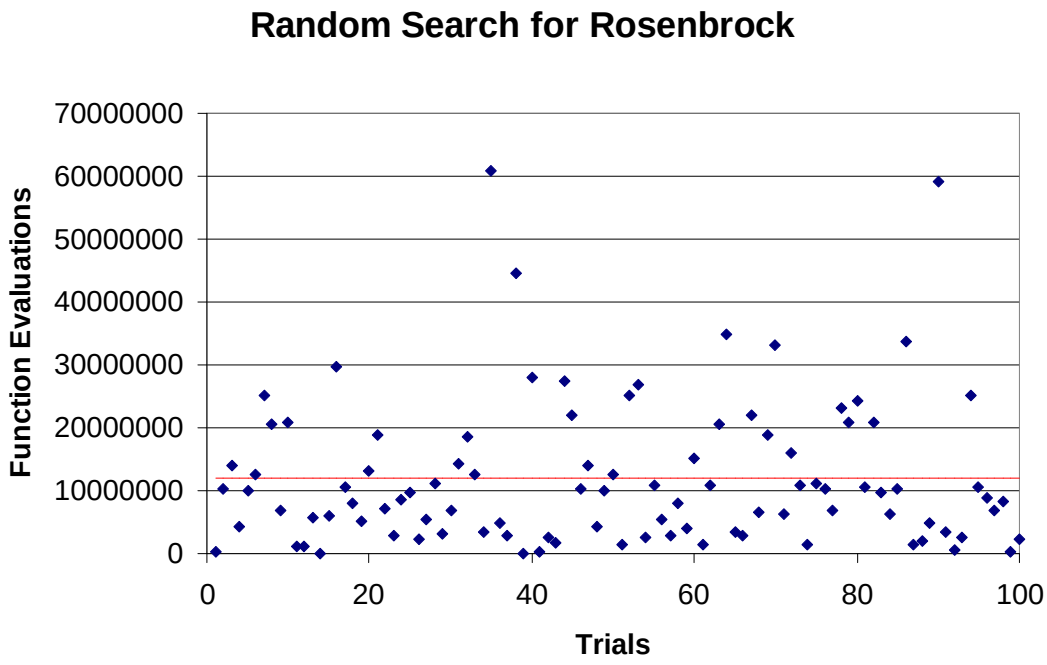



Figure 4.12 Trial results for random search, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

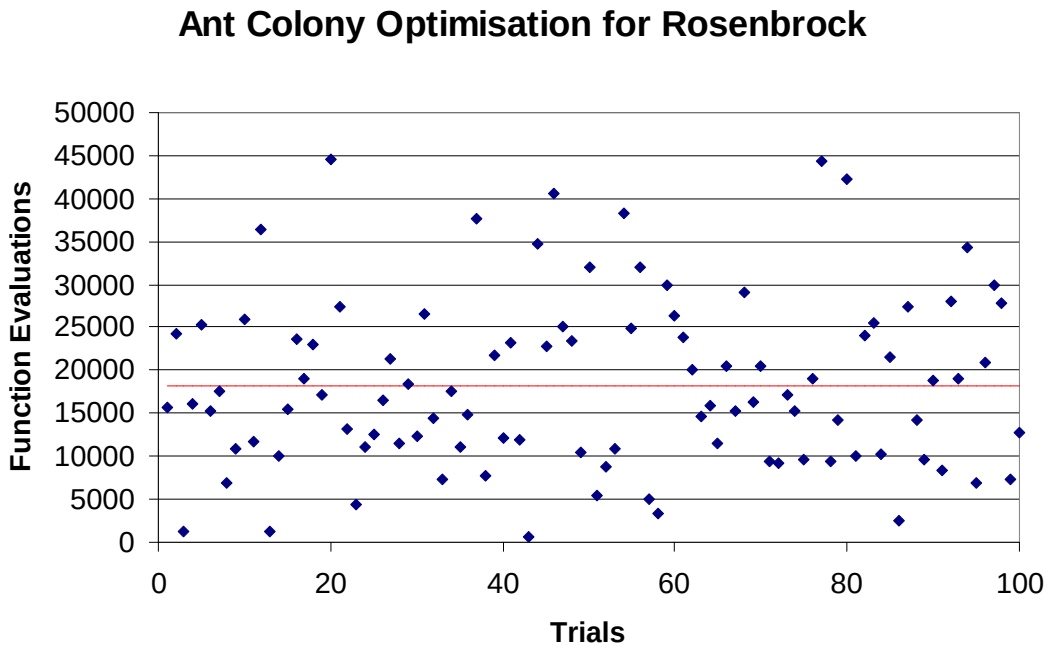



Figure 4.13 Trial results for the Ant Colony Optimisation algorithm, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

### 4.2.2 Rastrigin

The Rastrigin function is a far more complex surface, containing multiple local optima dispersed over a wide area. The function is described by the expression –

$$F_2(\vec{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right] \qquad \text{for the range } [-5.12, 5.12]^n,\ n \geq 1$$

The global solution for this function is defined as -

$$F_2(\vec{x}^*) = 0 \qquad \vec{x}^* = \{0,0,0,\ldots 0\}$$

One curiosity of this function is that the global optimum exists at the point (0,0). Because CACO initialises the nest to this point, it was necessary to relocate the nest so that some meaningful run-time information could be obtained. In order to do this, the nest was positioned at a random point within the bounds of the search at the beginning of each trial run. This ensured that there was no “biased placement” of the nest in positions likely to result in faster-than-average solution discovery.

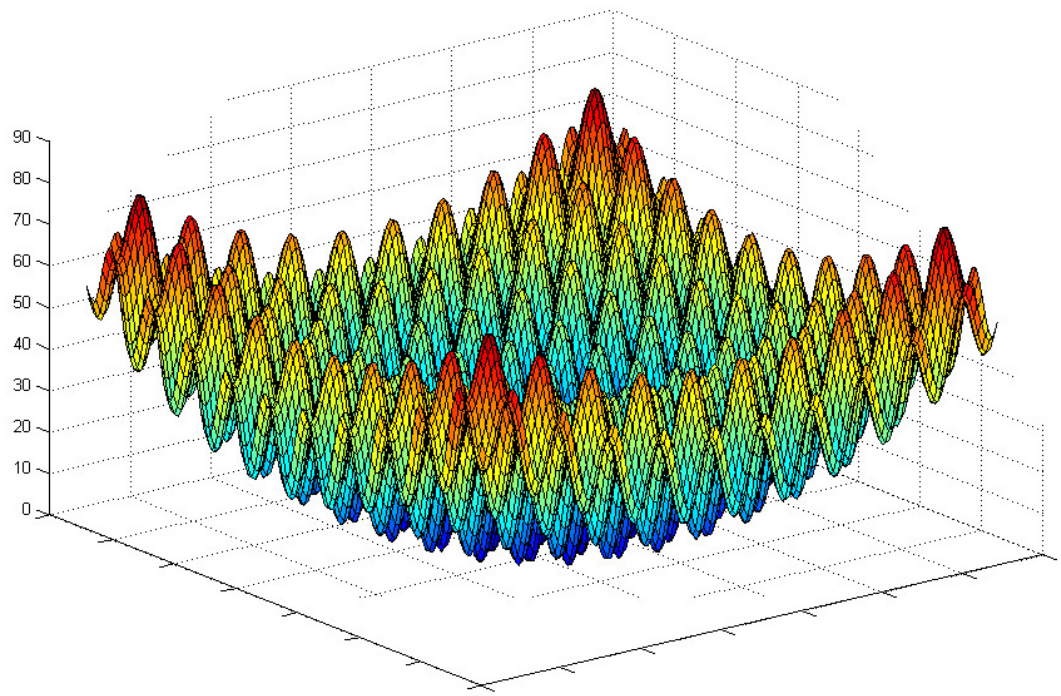



Figure 4.14 A plot of the Rastrigin function

Once again, CACO easily out-performed blind search, taking an average of ~9900 cost-function evaluations, compared to the ~5.5 million needed for blind search.

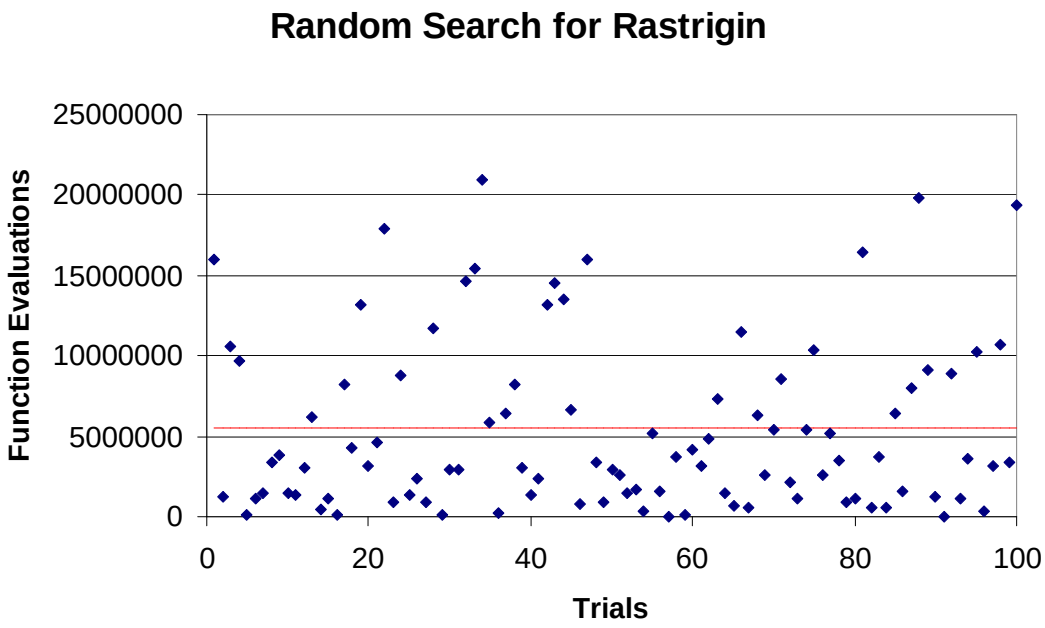



Figure 4.15 Trial results for random search, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

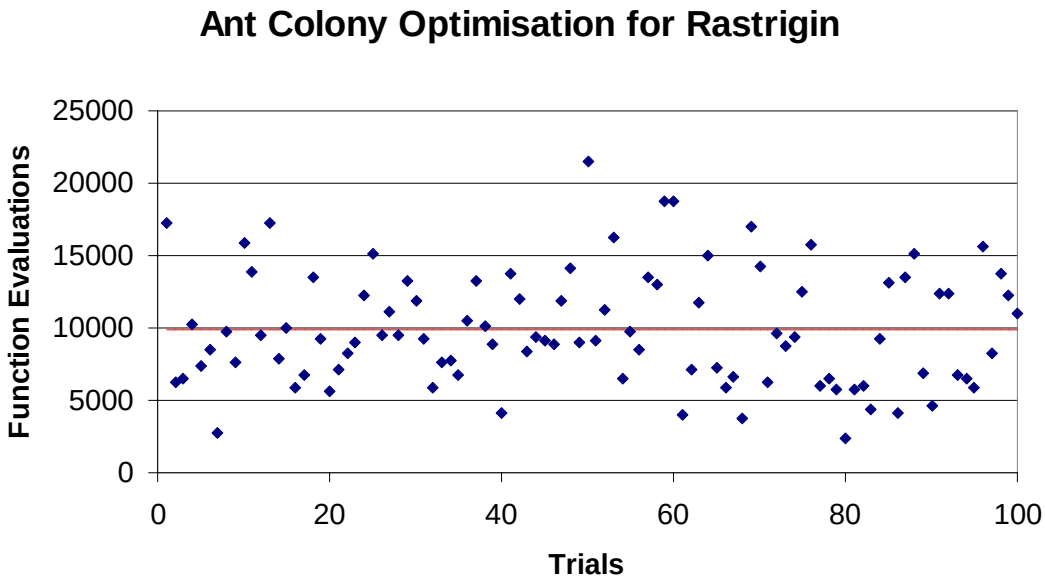



Figure 4.16 Trial results for the Ant Colony Optimisation algorithm, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

### 4.2.3 Schwefel

The Schwefel function is described by the expression –

$$F_3(\vec{x}) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right) \qquad \text{for the range } [-500,500]^n,\ n \geq 1$$

The global solution to this function is defined as –

$$F_3(\vec{x}^*) = -418.98291n \qquad \vec{x}^* = \{s, s, s, ..., s\}, \quad s = 420.97$$

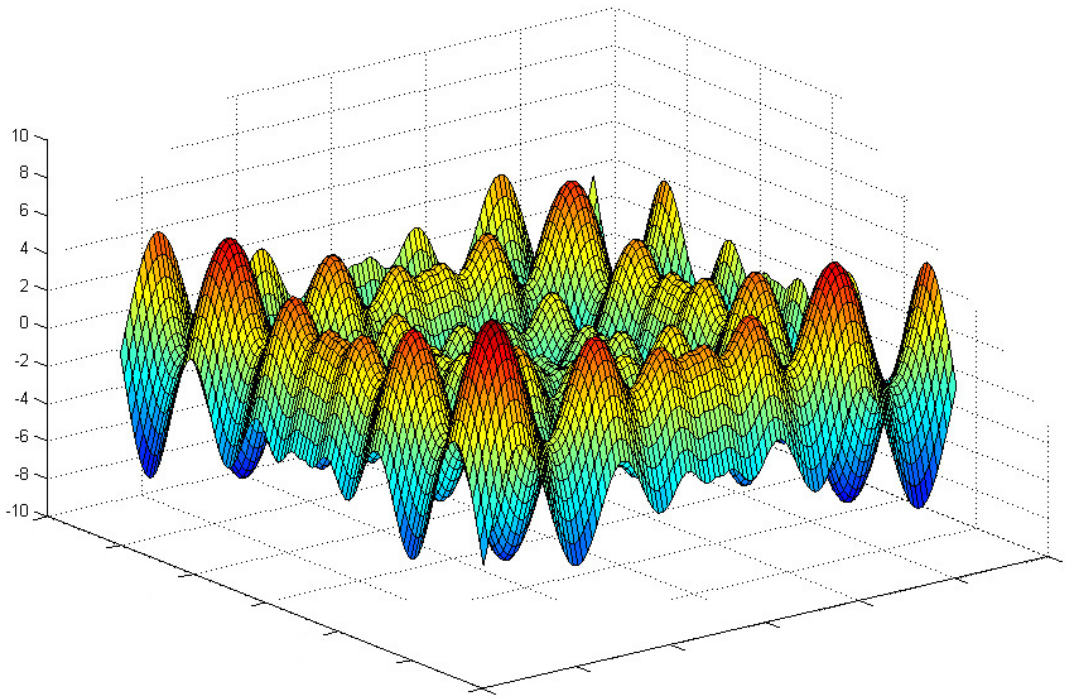



Figure 4.17 A plot of the Schwefel function

Initial tests for the Schwefel function found that CACO *failed* to find the global minimum in over 70% of cases. It was not immediately clear why this was happening, at first, however further investigation revealed a close correlation between the starting point (location of the nest) and the success rate. Eventually, it was discovered that ants were becoming trapped in local minima, but were unable to escape due to two main reasons. The first problem was that ant starvation was not occurring, as the ants were all being drawn to the same region of attraction (ie, all the ants were encountering the same (relatively) good area, and were not "complaining" about the quality of the solutions they were finding there. The second (and more significant) problem was that ants did not have enough "range" to escape from these local optimums. Essentially, ants were converging on solutions they could reach, but

did not have a large enough search radius to escape the local basins of attraction they had been pulled into. This phenomena is explained in Figure 4.18, which shows a cross-sectional plot of points taken from a plane intersecting the x-axis of the Schwefel function. In this diagram, ants are positioned at two separate local minima. Ant 1 is at the “higher” position, and is capable of reaching better territory by moving left a distance equal to the maximum allowable search radius. Ant 1 will eventually encounter the point occupied by Ant 2. Ant 2, however, will never be able to reach the global optimum, because it has insufficient range to move to better ground. The maximum search radius prevents Ant 2 from ever discovering the global solution, due to the width of the barrier in between the ant and the lower point.

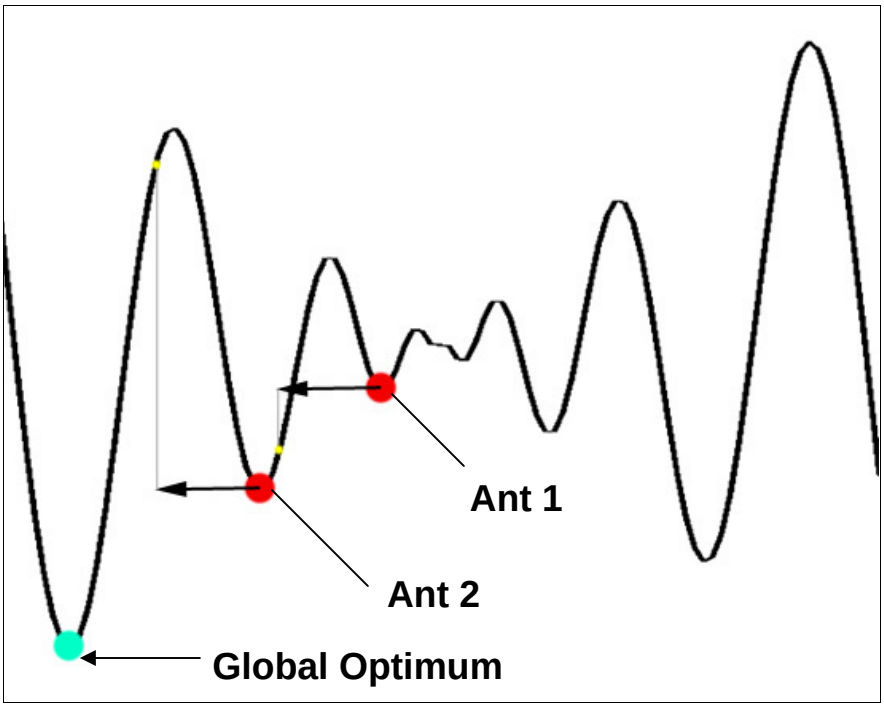



Figure 4.18 A small search radius prevents ants from overcoming large obstacles, and may prevent them from discovering the global optimum.

From the above illustration, it is clear why the matter of ant range affects the performance of CACO on the Schwefel equation. Simply put, for the cases where the ant nest is situated on the “far” side of an impassable barrier, ants are never able to reach the optimal solution. However, when the ants begin on the near side, there is no (impassable) obstacle preventing them from eventually finding the solution. In two (or more) dimensions, the problem is no different. Figure 4.19 shows a topological plot of the Schwefel function, with the optimal solution being located in the lower left corner (dark blue). As can be seen in the diagram, a relatively wide L-shaped region of unpromising solutions exists not far from the edge of the boundary (indicated by

the red and yellow). Assuming that the nest has been (randomly) positioned somewhere inside the region bounded by this fringe, ants will be unable to pass over into the better areas. In effect, the search space has been partitioned into two distinct areas – those within which ants can reach the optimum, and those where ants can't.

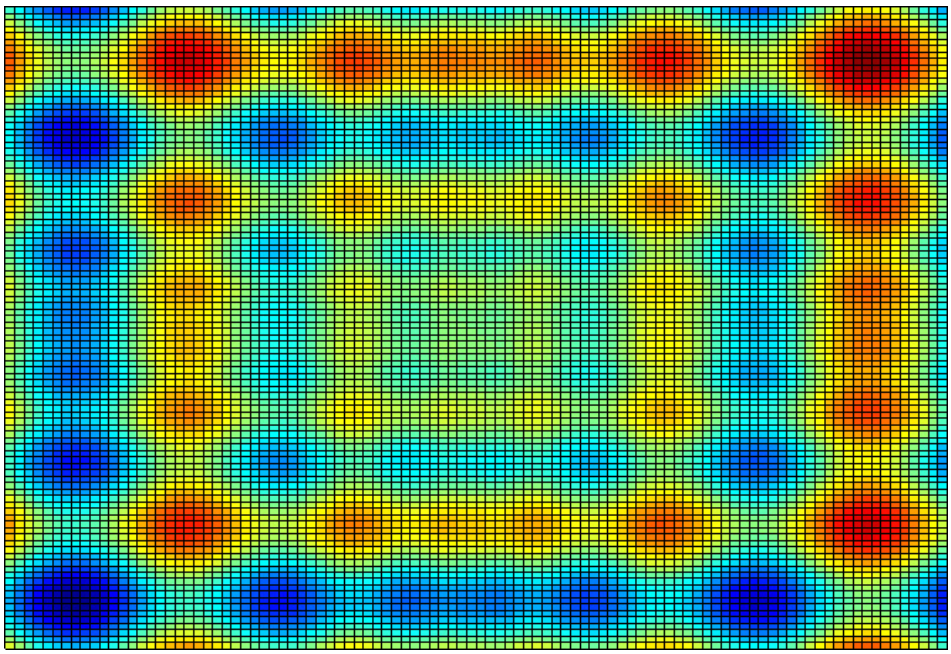


Figure 4.19 Topology of the Schwefel function, showing the fringe preventing certain ants from ever reaching the global solution.

This particular problem is not limited only to CACO, but also to many other algorithms that employ similar tactics of refining a local search result to obtain better solutions. For instance, the issue described here as "insufficient range" can be likened to setting the system temperature too low in Simulated Annealing.

To overcome this problem, the simplest approach is to widen the starting radius. This allows ants greater flexibility and less restriction in movement, allowing them to successfully climbs over hills that they would otherwise be unable to climb. A drawback to this is that the algorithm will take slightly longer to converge, since the rate of restriction is independent of the length of the radius. However, it must be said that, generally speaking, having an optimal solution to the problem would be worth the fractionally longer execution time required to find it.

To generate some usable results for CACO handling the Schwefel function, the initial search radius was doubled. The results are shown in Figures 4.20 and 4.21.

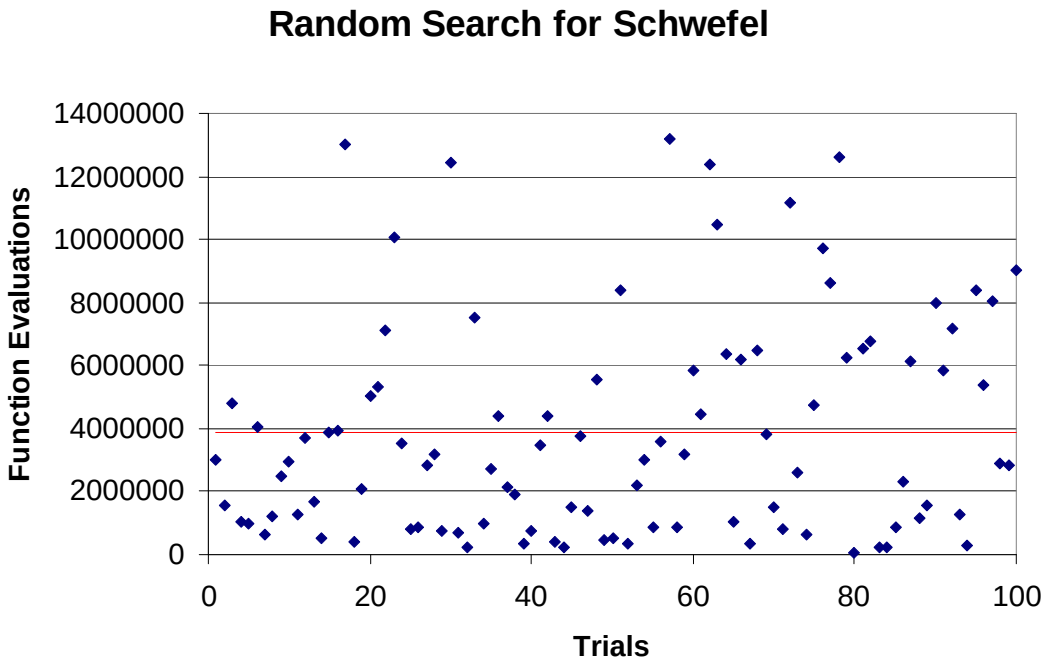



Figure 4.20 Trial results for random search, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

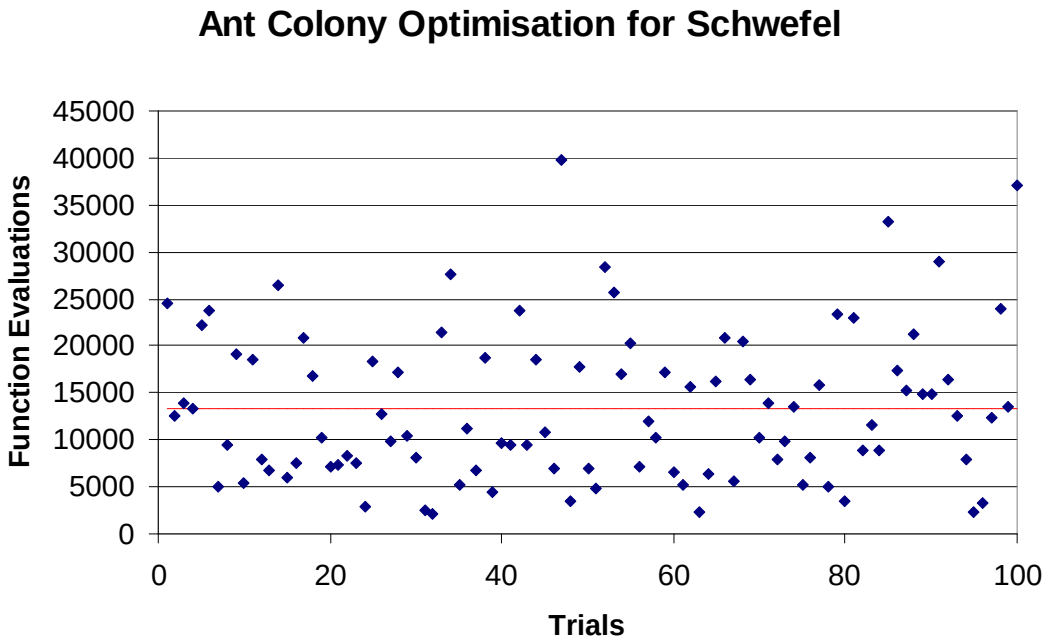



Figure 4.21 Trial results for the Ant Colony Optimisation algorithm, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

### 4.2.4 Griewangk

The Griewangk function is described by the expression –

$$F_4(\vec{x}) = 1 + .00025\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(x_i / \sqrt{i}\right) \qquad \text{for the range } [-500,500]^n,\ n \geq 1$$

The global solution to this function is defined as –

$$F_4(\vec{x}^*) = 0 \qquad \vec{x}^* = \{0,0,0....,0\}$$

Once again, this function calls for dynamic nest relocation, since the global optimum exists at the default point of nest initialisation (0, 0). Unlike the case with the Schwefel function, the placement of the nest did not drastically affect solution discovery time (or prevent discovery at all). However, it was found that the standard rate of radius restriction was too rapid, meaning that some trial runs did not find a solution within the accepted "value-to-reach" range before algorithm termination. To counter this problem and allow more time for ants to "home in" on the global solution, the convergence rate was relaxed from 0.99 to 0.999. This gave the ants the extra degree of precision necessary to find the global optimum, with an average of ~45000 cost function evaluations required by CACO, and roughly ~5.1 million by random search.

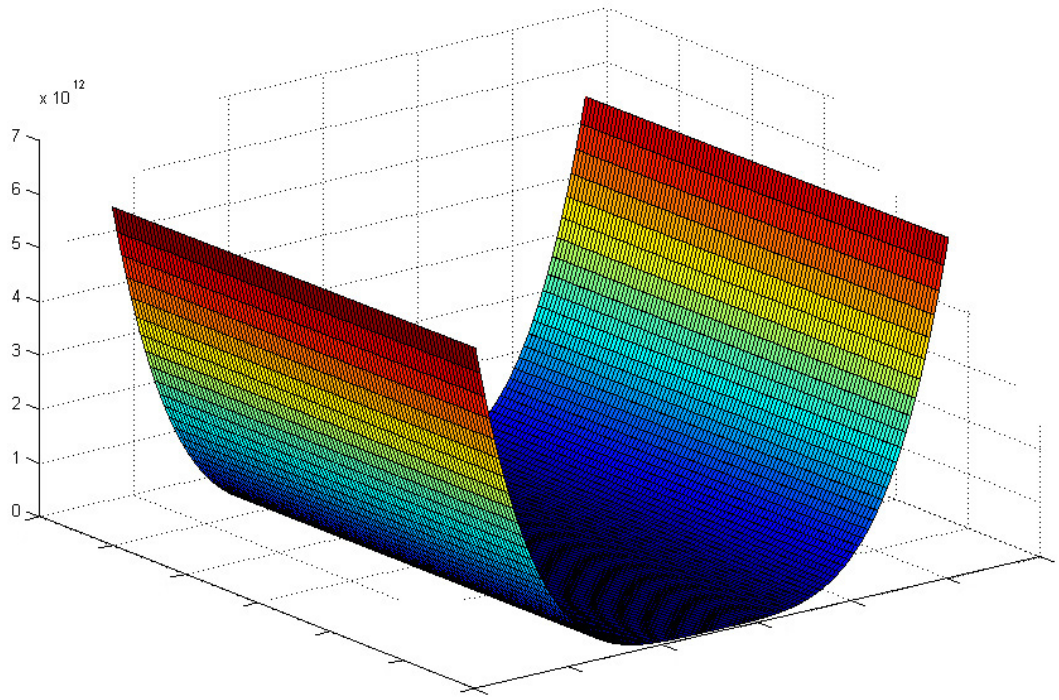



Figure 4.22 A plot of the Griewangk function

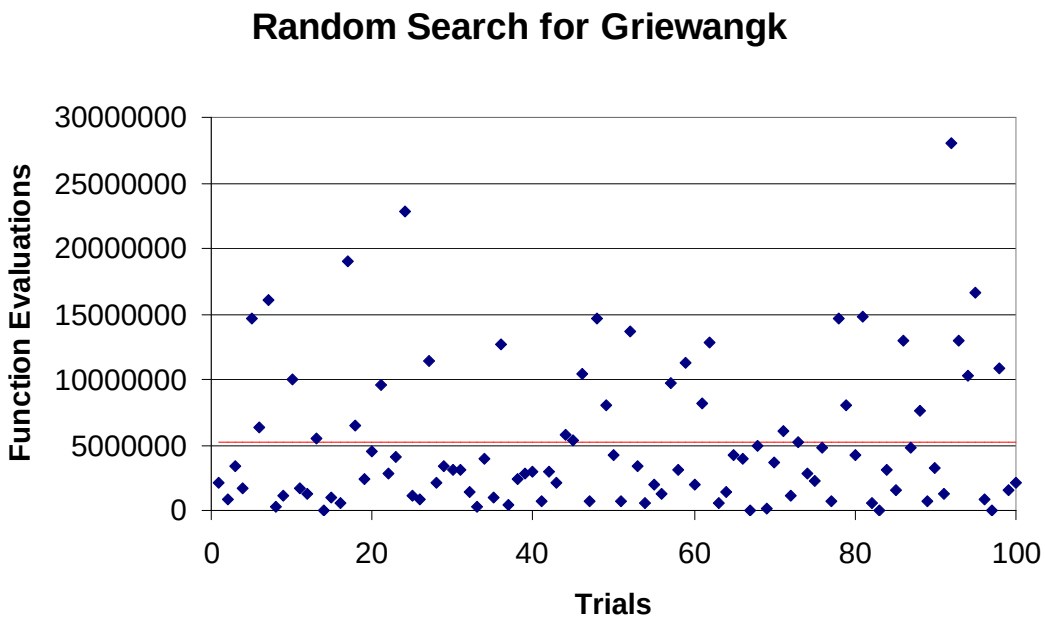



Figure 4.23 Trial results for random search, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

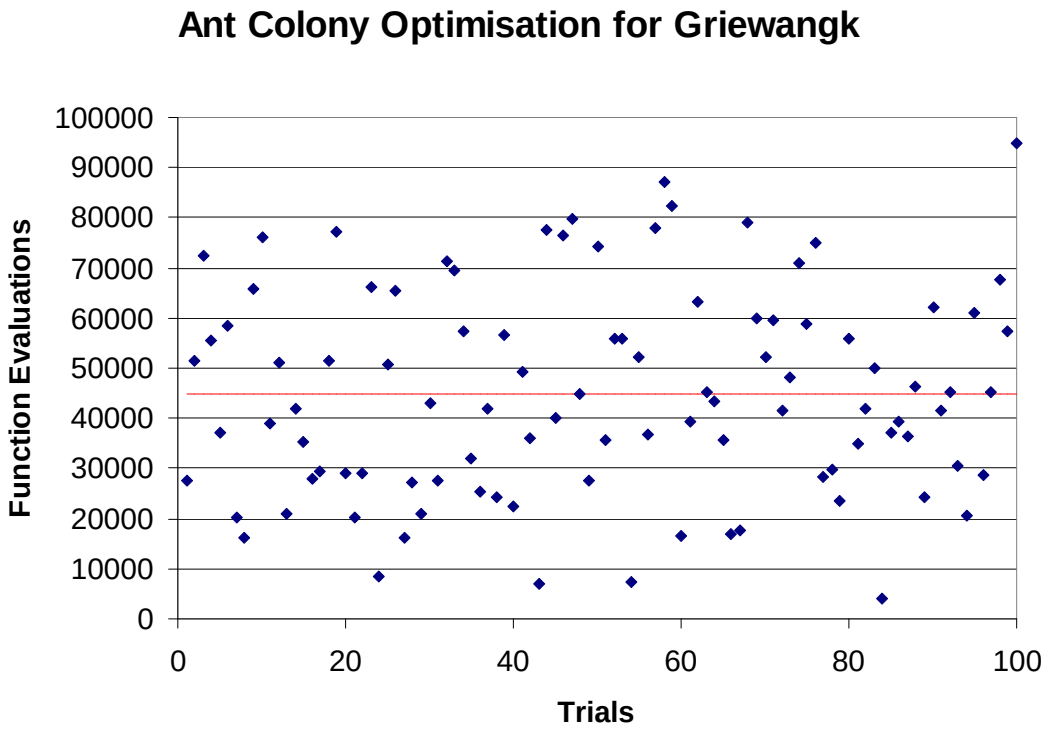



Figure 4.24 Trial results for the Ant Colony Optimisation algorithm, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

### 4.2.5 Salomon

The Salomon function is described by the expression –

$$F_5(\vec{x}) = 1 - \cos\left(2\pi \|\vec{x}\|\right) + .1\|\vec{x}\| \qquad \text{for the range } [-100,100]^n,\ n \geq 1$$

$$\|\vec{x}\| \equiv \sqrt{\sum_{i=1}^{n} x_i^2}$$

For this function, the global solution is defined as -

$$F_5(\vec{x}^*) = 0 \qquad \vec{x}^* = \{0,0,0,\ldots 0\}$$

The Salomon function is widely regarded a difficult optimisation problem due to the fact that any highly focussed local search is often “tricked” into thinking that the slope towards better solutions runs upwards towards the search boundaries. This is due to the number of small upward-jutting ridges covering the surface, seen in Figure 4.25.

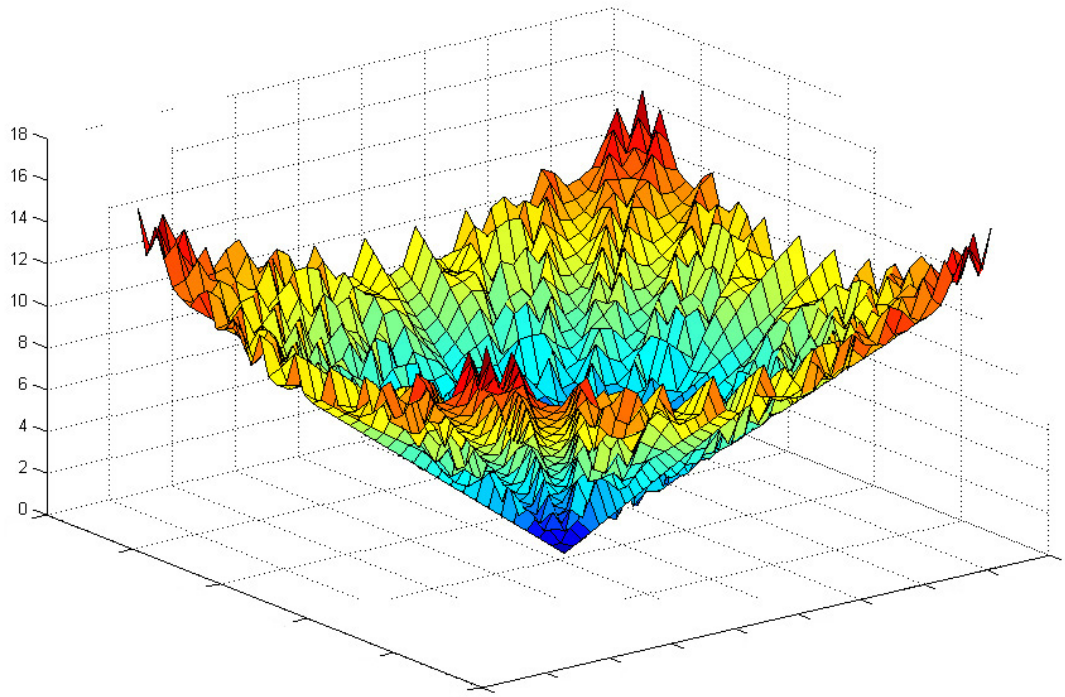



Figure 4.25 A plot of the Salomon function

Despite this, CACO successfully located the global optimum in each of the 100 trial runs, averaging ~66000 cost function evaluations before the solution was found. Comparatively, blind search took around ~29 million evaluations to achieve the same result.

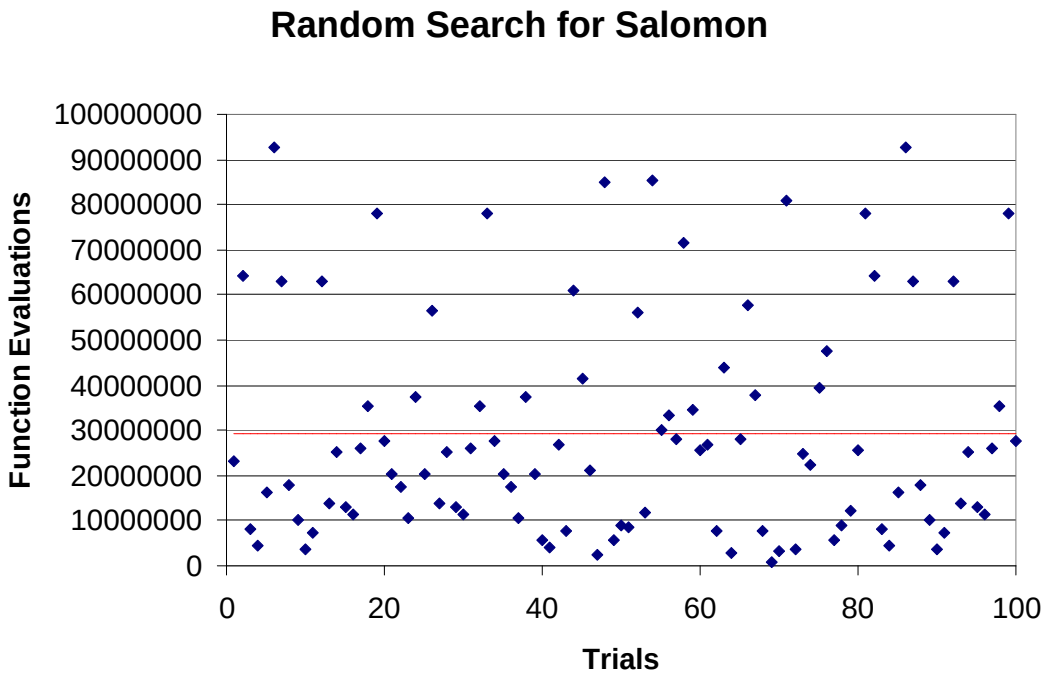



Figure 4.26 Trial results for random search, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

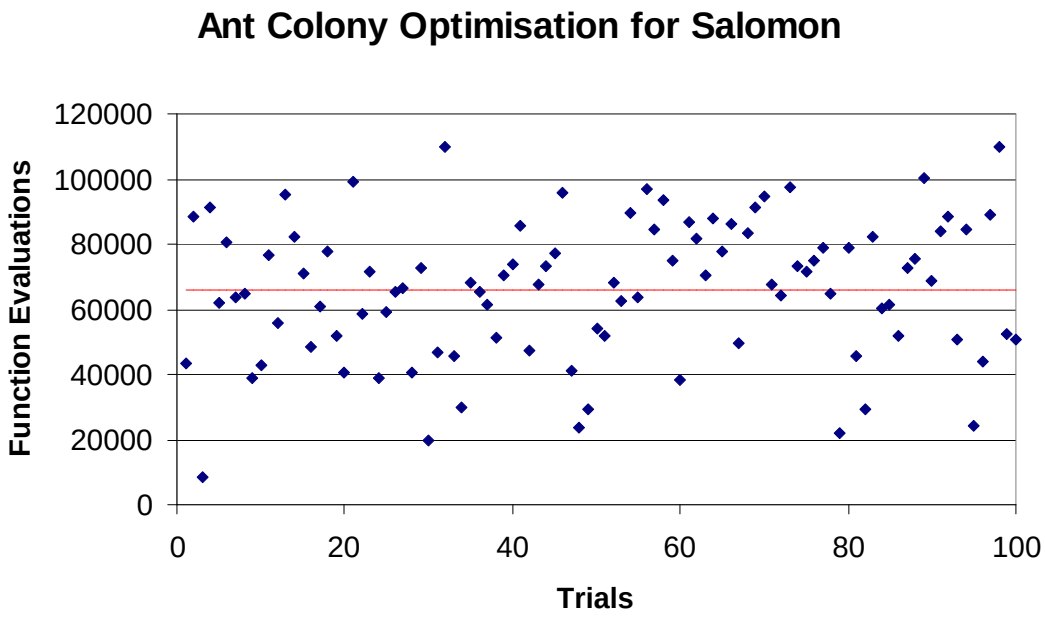



Figure 4.27 Trial results for the Ant Colony Optimisation algorithm, showing the number of function evaluations required before a value sufficiently close to the global optimum was found.

## 4.3 Parameter Setting

The trials in the previous section demonstrate that CACO is capable of solving a diverse range of optimisation problems, and in most cases can discover global solutions without too much difficulty. However, care must be taken to ensure that ants are given enough flexibility to roam all throughout the search space, or the risk of completely missing the best solution may arise. Unfortunately, for some functions, the nature of the search terrain is not known in advance, and so it is prudent to set a generously large initial search radius for these cases, to avoid the likelihood of problems.

Following these trials, some parameter variance experiments were conducted to determine how the algorithm behaves under different starting conditions. The first and most obvious of these experiments concerned the number of ants that are used to explore the space, and how this number affects the rate of convergence (Figure 4.28).

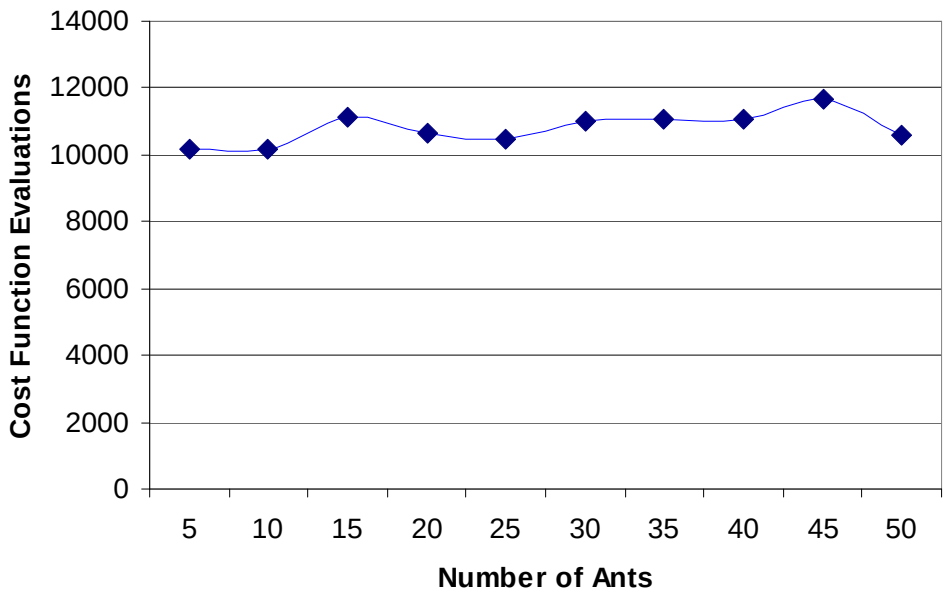



Figure 4.28 The number of ants used to explore a search space (Rastrigin), and the corresponding number of cost function evaluations needed to find the optimal solution.

Perhaps surprisingly, it was found that the number of ants the system is initialised with does not significantly affect the overall performance of the algorithm. Once again, this can be explained by considering the fixed rate of radius reduction, and the build up of pheromone. An increased ant colony population results in a higher average of visits per direction vector for each cycle or generation of the algorithm.

For systems with more ants than vectors, this means that some vectors will be visited more than once. This increased ant density, although not necessarily contributing directly to detailed local exploration, results in more extensive terrain sweeping, and ultimately more rapid pheromone build up. The colony is therefore able to quickly establish the best areas in the terrain, and concentrate search in these areas.

Because of the fact that pheromone weightings associated with direction vectors are updated on an ant-by-ant basis (i.e., each ant updates the vector it is exploring after evaluating its own fitness), some vectors may get a double (or triple, or higher) doses of pheromone (depending on the ant : vector ratio) – and these vectors will therefore gain much higher selection probabilities. This, in part, helps to account for the results described above – the catch being that the chance of stumbling across a solution increases when the number of ants in the area is higher.

However, it was noted that for ant populations significantly higher than the number of direction vectors (10 times or more higher), the pheromone system begins to break down, and ants are led to believe that any area not showing immediate improvement must not be worth examining at all. This defeats the purpose of pheromone-based path selection and exploration, and really isn't an efficient or reliable way of running the algorithm.

It is therefore recommended, from the findings of these trials, that an ant : vector ratio somewhere in the vicinity of 2 : 1 is used, ensuring an adequate amount of vector end-point exploration, while at the same time disallowing enormously fast pheromone build-up, and premature "ruling out" of areas of the search that haven't been properly considered.

Similarly, it was ascertained that the number of direction vectors (representative of the amount of parallelism in the system), should be set to a value likely to encourage the consideration of many different regions at once (12 was a value that seemed to work well, however it was values below 4 and above 50 that resulted in noticeably degraded performance). Without a reasonable number of direction vectors (more than 4), the system finds it much harder to distinguish between good and bad solutions. Too many vectors, on the other hand, results in a gross amount of exploration over-lap, as all the vectors are drawn to the same regions, where ants repeatedly cover territory that other ants have already evaluated, without being the wiser.

## 4.4 Evaluation of Pheromone Usefulness

In section 4.2, CACO was compared to blind search for each of the 5 functions in the test suite. These trials clearly demonstrated that CACO was capable of out-performing random search, however little was said about how the use of pheromone trails was actually influencing the search. To investigate this, a further set of trials was conducted, examining the performance of CACO with and without the use of pheromone trails.

In order to conduct this comparison, the Rastrigin function was again used (Figure 4.14). This function was chosen because the surface it described is reasonably complex, and can be considered a non-trivial optimisation problem due to the large number of local optima it contains.

To determine the usefulness of pheromone trails in guiding the search, a batch of trials was run in which all pheromone-related functionality in the CACO algorithm was disabled. This is explained in the revised CACO pseudo code below, showing which components of the algorithm were removed. These modifications leave what is essentially a kind of population-based hill climbing algorithm; however there is no kind of information exchange between ants in the system.

```
initialize colony()
evaluate(t)
while (not termination_condition){
      time ← time + 1
      add_trail(t)                    ← Disabled
      send_ants(t)
      evaluate(t)
      evaporate(t)                    ← Disabled
}
end
```

Removing pheromone trails (and all associated guidance devices, including ant starvation) results in a system that is entirely devoid of any search direction management. Ants are free to roam around anywhere (provided they move no further than the maximum search radius), and all direction vectors are just as likely as each other to be selected for exploration. As before, ants can only update vector positions

if they move to a location that is an improvement over the last point they previously occupied - however no information is left to guide subsequent ants as to the "goodness" of the solutions encountered. This adaptation of CACO without pheromone communication effectively amounts to a radius-bound distributed hill climbing algorithm.

During the trials, run-time information was collected for certain parameters. Most importantly, the "best solution so far" was recorded for each generation (after every 50 cost function evaluations). The results are shown in Figure 4.29, which shows data from a typical trial run.

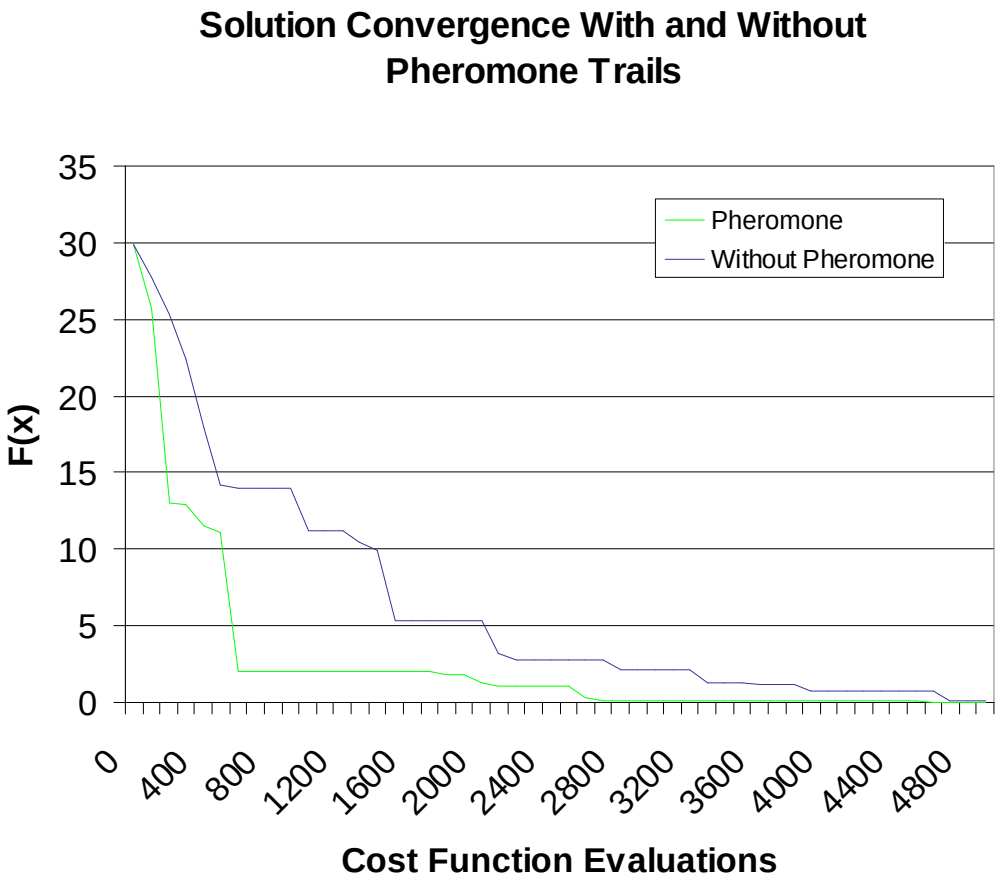



Figure 4.29 Performance of CACO with pheromone trails, and without pheromone trails.

This comparison clearly shows that the inclusion of pheromone mediated communication allows the ants to locate promising regions much faster than without pheromone trails. In this test, the pheromone system allowed good solutions to be located much more quickly than the distributed hill-climbing method.

An interesting (and unexpected) result in the above set of trials was the revelation that CACO was able to narrow down its search very close to the optimum solution in *well under* the average number of cost function evaluations required to converge on a “sufficiently exact” solution. CACO was successfully finding results evaluating to within 0.1 of the global optimum in just under ~2700 evaluations, when it was earlier shown that the average number of evaluations to solve the Rastrigin function was somewhere in the vicinity of ~9900 evaluations. This indicates that the algorithm is locating the “good” areas much quicker than expected, but ants are drifting distances too large to result in any local improvement. To enhance the performance of the algorithm under these circumstances, a faster rate of radius restriction could be used. Alternatively, a slightly different model of radius restriction could be adopted (such as the one described in Figure 3.14), that allows more flexibility in the initial phase of the algorithm, but an exponentially decreasing radius towards the end. This would ensure that less time is wasted with broader exploration when the areas of high solution fitness are already well-established.

# Chapter 5 : Conclusion

This thesis has demonstrated that the Ant Colony Optimisation algorithm, originally used exclusively for discrete optimisation problems, can be successfully adapted to continuous spaces, offering an excellent trade-off between exploration and exploitation. The work outlined in this document has shown that a population of co-operating artificial ants using pheromone trails as a method of information sharing is capable of solving both simple and reasonably difficult optimisation problems, with consistently encouraging results.

## 5.1 Summary of Results

The CACO algorithm implemented in this thesis was generally proven to be extremely versatile, being able to satisfactorily solve 4 of the 5 functions in the test suite with no modification at all to the default run-time parameters. However, it was found that for certain problems (such as the Schwefel function), the choice of parameters can directly hinder the algorithm's ability to locate the global optimum. The problem stems from the maximum search radius value limiting the search to specific areas, from which it is impossible to ever reach the global solution. This can be overcome by setting the initial search radius to some suitably large value, thus allowing ants the range and flexibility they require to reach all areas in the search scape.

CACO was shown to out-perform blind search by several orders of magnitude for all test functions, and has the unique feature that it considers many search directions in parallel. The use of pheromone trails as a method of information sharing between ants exploring the system allows for better solutions to be attained faster, but also allows independent local search to progress virtually unhindered. For some multi-minima problems (for example, the Rastrigin function), the distributed nature of CACO led to the unusual situation in which the ants discovered not only the global solution, but several other "next best" solutions discovered at nearby local optima.

It was also demonstrated that CACO was capable of out-performing a distributed hill-climbing method (quite similar to local search) for spaces that contained numerous basins of attraction.

## 5.2 Review of Project Plan

For the most part, all of the goals of this thesis project were accomplished, with the evaluation results indicating a very pleasing level of performance. Unfortunately, due to the sheer amount of time required to run batches of trials, collect, tabulate, and analyse results, examining the intricacies of all of the run-time parameters in the system was not possible. In spite of this, a great deal of useful information was obtained, which clearly indicated the usefulness of the CACO system. However, there are still aspects of CACO which could be further investigated and better understood in order to further refine future versions of this algorithm.

## 5.3 Future Work / Research

This paper has demonstrated that CACO is a viable option for continuos optimisation problems; however there are several areas in which further research could be conducted. These include:

*1) Extension of CACO to handle higher-dimensional problems.*

This version of the CACO algorithm was designed to exclusively handle 2-dimensional problems for two main reasons: ease of testing and debugging, and simplicity of observing and visualising search results. However, most of the optimisation problems that would be considered non-trivial involve a large number of variables. Fortunately, CACO could be quite easily adapted to handle N-dimensional search without an absurd blow-out in data structure storage requirements. Implementing and evaluating a version of CACO able to handle an arbitrary number of variables would be an interesting area for future research.

*2) Comparison of CACO to other "good" algorithms.*

In this paper, CACO was compared to blind search, and to a lesser extent, distributed hill-climbing. However, it would be worthwhile to compare CACO to a number of more advanced optimisation algorithms. It would be of particular interest to see how CACO performs when compared to other nature-inspired systems, such as Simulated Annealing, or Genetic Algorithms.

*3) Pheromone trail dissipation*

One idea that surfaced during the evaluation phase of this project was the possibility of implementing a form of pheromone trail dissipation. This would be useful for cases when several different direction vectors are all drawn to the same basin of attraction. It might be possible to take an average of two or more vectors in the same vicinity, and create a new vector pointing to some intermediate point. This could be particularly useful in the synthesis of new solutions, and would allow ants to access global information via the diffusion of pheromone trails. This "pheromone trail sharing" would be, in a sense, similar to cross-over in a genetic algorithm.

*4) Replacement of the nest with localised "interest regions"*

As a final suggestion for possible future research, removing the nest as a centralised starting point for all ants and replacing it instead with a distribution mechanism which starts ants dispersed evenly throughout the search space might be another option to consider. While this is a slight departure from the ant colony metaphor, the individual search agents (ants) would be unchanged, and pheromone trails could be replaced with pheromone rings; regions of interest which ants may decide to visit. Obviously this is a fairly different idea, but one that could be implemented reasonably easily from the existing CACO framework without too much adaptation. This option could be the topic of future research in continuous space optimisation.

# References

[1] M. Dorigo & A. Colorni, The Ant System: Optimization by a colony of cooperating agents , 1996.

[2] M. Dorigo & L. M. Gambardella, Ant colonies for the travelling salesman problem, 1997.

[3] N. Meuleau & M. Dorigo, Ant Colony Optimization and Stochastic Gradient Decent, December 2000.

[4] G. D. Caro & M Dorigo, Ant Algorithms for Discrete Optimization, 1999

[5] T. Stutzle & M. Dorigo, An Experimental Study of the Simple Ant Colony Optimization Algorithm

[6] M. Dorigo & T. Stutzle, ACO Algorithms for the Quadratic Assignment Problem

[7] Ant Colony Optimization Publications Index
http://iridia.ulb.ac.be/~mdorigo/ACO/publications.html
Last modified: Tue Mar 6 2001, Accessed: October 16, 2002.

[8] Ant Colony Optimization Website : About ACO
http://iridia.ulb.ac.be/~mdorigo/ACO/about.html
Last modified: Tue Mar 6 2001, Accessed: October 16, 2002.

[9] G. Bilchev & I.C. Parmee, The Ant Colony Metaphor for Searching Continuous Design Spaces.