Ants find the shortest path

Gareth Hunter

A dissertation submitted in partial fulfilment of the requirements of Dublin Institute of Technology for the degree of

M.Sc. in Computing (Knowledge Management)

**JULY 2012**

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Knowledge Management), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute’s guidelines for ethics in research.

***Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***Date: DD Month Year***

1. ABSTRACT

AAAA

*(approx. 250-300 words)*

**Key words:** *list 5 to 8 words*

**ACKNOWLEDGEMENTS**

I would like to express my sincere thanks ……….

*(thank all the people how have assisted you in completing your dissertation. Start with your supervisor, all DIT staff that may have helped, other people can include family and friends, industrial and academic staff from other institution, etc.)*

**TABLE OF CONTENTS**

[1 ABSTRACT ii](#_Toc393754977)

[TABLE OF FIGURES vii](#_Toc393754978)

[TABLE OF TABLES viii](#_Toc393754979)

[1. Introduction 1](#_Toc393754980)

[1.1 The Problem 3](#_Toc393754981)

[1.2 Background 4](#_Toc393754982)

[1.3 Research problem 5](#_Toc393754983)

[1.4 Intellectual challenge 5](#_Toc393754984)

[1.5 Research objectives 5](#_Toc393754985)

[1.6 Research methodology 6](#_Toc393754986)

[1.7 Resources 6](#_Toc393754987)

[1.8 Scope and limitations 6](#_Toc393754988)

[2 Swarm intelligence 8](#_Toc393754989)

[3 ANT COLONY OPTIMIZATION 9](#_Toc393754990)

[3.1 Introduction 9](#_Toc393754991)

[3.2 The ant system: 10](#_Toc393754992)

[3.3 global parameters 11](#_Toc393754993)

[3.4 Elitist ant’s algorithm: 13](#_Toc393754994)

[3.5 Rank-Based Ant System: 14](#_Toc393754995)

[3.6 Best-Worst Ant Algorithm 14](#_Toc393754996)

[3.7 Min-Max Ant System 15](#_Toc393754997)

[3.8 ANT COLONY SYSTEM 16](#_Toc393754998)

[3.9 LOCAL OPTIMAL 16](#_Toc393754999)

[3.10 STAGNATION MEASURES 16](#_Toc393755000)

[3.10.1 Variation Coefficient 16](#_Toc393755001)

[3.10.2 Branching Factor 16](#_Toc393755002)

[4 metaheuristic methods 18](#_Toc393755003)

[4.1 The Genetic algorithm 18](#_Toc393755004)

[4.1.1 Selection 19](#_Toc393755005)

[4.1.2 Roulette Wheel Selection 19](#_Toc393755006)

[4.1.3 Elitism 20](#_Toc393755007)

[4.1.4 Tournament selection 20](#_Toc393755008)

[4.2 Genetic operators 20](#_Toc393755009)

[4.2.1 Crossover 20](#_Toc393755010)

[4.2.2 Multi point crossover 21](#_Toc393755011)

[4.2.3 Mutation 22](#_Toc393755012)

[4.3 Limitations and conclusions 22](#_Toc393755013)

[5 Local Search 23](#_Toc393755014)

[5.1.1 2 2-opt 23](#_Toc393755015)

[5.1.2 3Opt 24](#_Toc393755016)

[5.1.1 Tabu search 24](#_Toc393755017)

[5.1.2 Local search Applied to ACO and GA 25](#_Toc393755018)

[5.2 Conclusion 25](#_Toc393755019)

[2. Np-complete 26](#_Toc393755020)

[1.1 Introduction 26](#_Toc393755021)

[5.3 O-notation 26](#_Toc393755022)

[5.4 TRAVELLING SALESMAN PROBLEM 27](#_Toc393755023)

[5.5 locally optimiser loop 28](#_Toc393755024)

[5.6 ANT COLONLY OPTIMIZATION APPLIED TO TSP 29](#_Toc393755025)

[5.7 Virtual ant’s 29](#_Toc393755026)

[6 SOFTWARE ENGineering 31](#_Toc393755027)

[6.1 Data Structures 31](#_Toc393755028)

[6.2 Antsystem classes 32](#_Toc393755029)

[6.3 Interface 32](#_Toc393755030)

[6.4 Factory 32](#_Toc393755031)

[6.5 Implementing Genetic Algorithm 32](#_Toc393755032)

[6.6 Conclusion 32](#_Toc393755033)

[7 ??? 33](#_Toc393755034)

[7.1 Introduction 33](#_Toc393755035)

[7.2 ??? 33](#_Toc393755036)

[7.3 Conclusion 33](#_Toc393755037)

[8 ??? 34](#_Toc393755038)

[8.1 Introduction 34](#_Toc393755039)

[8.2 ??? 34](#_Toc393755040)

[8.3 Conclusion 34](#_Toc393755041)

[9 Experimentation & Evaluation 35](#_Toc393755042)

[9.1 Introduction 35](#_Toc393755043)

[9.2 Experimentation 35](#_Toc393755044)

[9.2.1 Genetic algorithm configuration 35](#_Toc393755045)

[9.2.2 36](#_Toc393755046)

[9.3 Evaluation 36](#_Toc393755047)

[9.4 Conclusion 36](#_Toc393755048)

[10 Conclusion 37](#_Toc393755049)

[10.1 Introduction 37](#_Toc393755050)

[10.2 Research Definition & Research Overview 37](#_Toc393755051)

[10.3 Contributions to the Body of Knowledge 37](#_Toc393755052)

[10.4 Experimentation, Evaluation and Limitation 37](#_Toc393755053)

[10.5 Future Work & Research 37](#_Toc393755054)

[10.6 Conclusion 38](#_Toc393755055)

[BLIOGRAPHY 39](#_Toc393755056)

[Appendix A 43](#_Toc393755057)

TABLE OF FIGURES

[There is a question of when to apply global evaporation: immediately after all ants complete their trip, or after all pheromones on graph edges (paths between cities) are updated? Evaporation before updates is, essentially, increase of updating pheromones. (Dorigo and Socha, 2006b) to pseudo code for the above algorithm and elitist system is shown in Figure 3 9](#_Toc393580897)

[Figure 3 . Ant colony system pseudo code 10](#_Toc393580898)

[Figure 1 15](#_Toc393580899)

[Figure 2 15](#_Toc393580900)

TABLE OF TABLES

[Table 1 11](#_Toc393581233)

Key Terms

Metaheuristic: consists of a set of algorithmic concepts used to solve a general class of computational problems, which can be applied to different problems with only a few modifications.

# Introduction

Ants are social insects that evolved from wasps like ancestors more than a hundred million years ago ants live in colonies with each ant having a defined job, with these attributes, ants and are extremely successful, organised insects covering the surface of the planet except for the Antarctic and some remote islands. (Hölldobler, 1990). If you have ever watched ants, you ask the same question: How can ants get anything done? put it this way by Mark Twain “[As a thinker and planner the ant is the equal of any savage race of men; as a self-educated specialist in several arts she is the superior of any savage race of men; and in one or two high mental qualities she is above the reach of any man, savage or civilized](http://thinkexist.com/quotation/as_a_thinker_and_planner_the_ant_is_the_equal_of/325463.html)!”

Swarm intelligence (Beni and Wang, 1993) can be defined as the collective behaviour of decentralised self-organising system artificially or by nature. Ant optimisation (Dorigo et al., 2006a) (Goss et al., 1989) is a section of swarm intelligence in which researchers study the behaviour of social insects, including ants, bees, and termites. The ability that swarm insects have to survive and solve complex tasks, whereas each insect cannot survive on its own is of particular interest to computer algorithm designers. “Ants are able to discover the shortest path to a food source and share this information with other ants” (Sim and Sun, 2003) Swarm intelligence is of particular interest in the robotics domain, and other areas.

In this thesis we investigate five different variants of these Ant optimisation algorithms and the genetic algorithm for solving NP-HARD problems. When optimisation problems are difficult to solve and cannot be solved efficiently or quickly are said to be NP-Complete. Typically, the travelling salesman problem easy to understand but increasing in complexity exponentially as a new city is added. Informally a salesperson intends to visit a number of cities and can only visit a city once, and returning to the start point minimising the distance they have to travel. (Karp, 1977). In more formal terms, TSP can be defined as a “*Hamiltonian tour of minimum length on a fully connected graph*”.

Furthermore, we extend these implementations introducing local search methods and compare the heuristics that guide the ACO algorithms. Moreover, experimenting with different pheromone update strategies and present a framework called Antsolver, created using advanced C++ methodologies. In order to demonstrate this we present an ACO implementation for the travelling salesman problem based on this framework.

Furthermore, presented is a local search procedure also applied to the basic ACO and Elitist ant system to my research this hasn’t been researched yet and the results are surprising.

Computational results for selected instances of the TSP library show that the ACO metaheuristic gives results comparable to those of other methods such as branch and bound and tabu search for many problem instances.

The aim of this paper is to discuss theoretical foundations and examples of practical applications of ant algorithms. These algorithms represent a new promising

approach to solving optimization problems that is based on the simulation of the behavior of ant colonies. An ant colony can be regarded as a multi-agent system

where each agent (ant) is functioning independently by very simple rules. Unlike the nearly primitive behavior of the agents, the whole system happens to function in

an amazingly reasonable way: “… nests of many species of ants surprise us by their dimensions and complex and rational architectonics. There are paths and

tunnels scattered on the niche territory for arphides and cochineal, and mushroom gardens… There exist various ways of storing and stoking up with food as well as

a real domestication of some species of insects…” [1

An interesting result of the cooperative behavior of biologic ants is the way they locate the shortest path from food source to the nest. Optimization algorithms

imitating such a behavior of ants were proposed in early 1990s in Italy [2]. The first paper on ant algorithms was published in an international journal in 1996 [3], and, it

took only a few years after this for a new field of scientific research (Swarm Intelligence and Ant Algorithms) to appear. Currently, many European researchers have successfully been working in this field. Biennially, international workshops on ant colony optimization and swarm intelligence have been organized in Belgium.

Special “ant” sections and workshops have been organized in the framework of international congresses and big conferences, and special-purpose issues of international scientific journals have been published.

problems,such as the traveling salesman problem, the vehicle routing problem, the problem of graph coloring, the quadratic assignment problem, the problem of network-

traffic optimization, the problem of job-shop schedule planning, etc. A key event in recognizing promises of the ant optimization was the winning of the 50000-Euro Marie Curie Excellence Award by the inventor of ant algorithms Dr. Dorigo in 2003.

* 1. The Problem

During the last couple of decades we have used computers to help

us solve a variety of problems. While many of these problems can be

solved in a reasonable amount of time, other problems are intractable

excluding the most trivial of instances. For instance, sorting a list of

numbers can be accomplished very quickly by a computer independent

of the list’s size.1 On the other hand it is very computationally expensive

to determine the shortest round-trip given a number of cities.2 This

problem, commonly known as the Traveling Salesman Problem (TSP),

is proven to be **NP**-hard meaning that the time required to solve it

increases very quickly with the problem size. This is not the case with

problems belonging to the complexity class **P** that also contains the

list-sorting problem mentioned beforehand.

**NP**-hard problems are often approached by approximation algorithms

that give solutions that are probably suboptimal but can be

computed in a reasonable amount of time. One such approximation for

the TSP is to create a round-trip that visits the nearest not yet visited

city next. This method might create a decent solution but optimality is

neither guaranteed nor very likely in general.

A subset of these **NP**-hard problems, those that can be formulated

as constraint satisfaction problems, can be solved efficiently by representing

them as so called Tree- or Hypertree Decompositions. Each

decomposition has a characteristic called width and each problem can

be transformed to many different valid decompositions. The smaller

a decomposition’s width the faster the solution to the problem can

be computed. Unfortunately, the problem of finding the decomposi-

One of the main ideas of ant algorithms is the indirect communication of a colony of agents, called (arti¯cial) ants, based onpheromone trails (pheromones are also used by real ants for communication).

The (arti¯cial) pheromone trails are a kind of distributed numeric information

which is modi¯ed by the ants to re°ect their experience while solving

a particular problem. Recently, the Ant Colony Optimization (ACO) metaheuristic

has been proposed which provides a unifying framework for most

applications of ant algorithms [15, 16] to combinatorial optimization problems.

In particular, all the ant algorithms applied to the TSP ¯t perfectly

into the ACO m

The TSP is extensively studied in literature [29, 31, 45] and has attracted since

a long time a considerable amount of research e®ort. The TSP also plays an

important role in Ant Colony Optimization since the ¯rst ACO algorithm,

called Ant System [18, 14, 19], as well as many of the subsequently proposed

ACO algorithms [21, 17, 52, 53, 7] have initially been applied to the TSP. The

TSP was chosen for many reasons: (*i*) it is a problem to which ACO algorithms

are easily applied, (*ii*) it is an *NP*-hard [26] optimization problem, (*iii*) it is

a standard test-bed for new algorithmic ideas and a good performance on

the TSP is often taken as a proof of their usefulness, and (*iv*) it is easily

understandable, so that the algorithm behavior is not obscured by too many

technicalities. calculated using the Euclidean norm. All the TSP instances we use are taken

from the TSPLIB Benchmark library [44] which contains a large collection of

instances; these have been used in many other studies or stem from practical

applications of the TSP. TSPLIB is accessible on theWWWat the address

<http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html>

|  |
| --- |
|  |

TSP instance att532 from TSPLIB; it comprises 532 cities in the USA.

The given tour is of length 27686, which in fact is an optimal tour.

* 1. Background
  2. Research problem
  3. Intellectual challenge
  4. Research objectives

Swarm intelligence is generally simpler and more flexible than traditional algorithms at solving NP hard problems and can solve many real-world complex problems. Among the most popular swarm intelligence algorithms is ant colony optimisation (ACO). This algorithm has been well researched and variations developed such as Ant System, Elitist Ant System, Ranked Ant System, Best-Worst Ant System and Min-Max Ant System.

The aim of this research is to analyse the performance of each variation of ACO algorithms, using the well-researched and benchmarked travelling sales person library (TSPLIB) dataset as input. A framework will be presented, developed in C++ using OO paradigms including design patterns and the DRY principle, to completely decouple each part of the application to enable expandability and an open architecture, developed using test driven development methodology, keeping in context with my MSC in advanced software engineering being pursued.

Empirical analysis will be used to compare the developed algorithms with TSPLIB input dataset that already provides a complete set of peer-reviewed benchmarks of the performance of each algorithm and how they should compare to each other, with different input problems. The framework will calculate each algorithms best solution, average solution, the average time for the best solution and the maximum time allowed for a particular amount of iterations. Profiling software will be utilised to accurately evaluate each of these measurements. ACO will also be benchmarked with the genetic algorithm (GA) using results defined by other researchers via TSPLIB.

In addition, the framework will solve the travelling sales person problem for Irish cities/towns using ant colony optimisation.

The following objectives have been achieved throughout the dissertation and contributed to the overall outcome:

* 1. Research methodology
  2. Resources
  3. Scope and limitations
  4. Organisation of the dissertation

The first section presents the self-organization principles of social insects and explains the way the ants locate the shortest path. The second section brings an example of the traveling salesman problem to demonstrate how the cooperative behavior of ants can be used in algorithms of combinatorial optimization. The third section discusses

methods for improving ant algorithms; and the fourth section reviews the applications of the ant algorithms. The first and second sections are based on the works

Chapter ?? defines the basic terminology used in this thesis and gives(Shtovba, 2005)

an overview on some fields of knowledge relevant to ant colony optimisation and the travelling salesman problem

In Chapter ?? exact and heuristic methods that have already been applied to the problem of TSP.

Chapter ?? is all about the Ant Colony Optimization metaheuristic, its roots in nature and the differences between the variants. Furthermore, various real-world

problems are described that already have been solved using ACO.

In Chapter ?? we present our approach of applying ACO to the problem

OF tsp. We present different strategies of updating the pheromone matrix, measures to determine algorithm stagnation and local search algorithms in order to improve the solutions constructed by the ants.

Chapter ?? documents all implementation artefacts that were created in the course of this thesis. There is a focus on the Antsystem

In Chapter ?? we give the computational results we obtained by applying our implementation to examples taken from popular benchmark libraries.

Finally, Chapter ?? concludes and describes future work.

1. Swarm intelligence

Swarm intelligence (Beni and Wang, 1993) can be defined as the collective behaviour of decentralised self-organising system artificially or by nature researchers study the behaviour of social insects, including ants, bees, and termites. The ability that swarm insects have to survive and solve complex tasks, whereas each insect cannot survive on its own is of particular interest to computer algorithm designers. “*Ants are able to discover the shortest path to a food source and share this information with other ants*” (Sim and Sun, 2003)

Swarm intelligence is of particular interest in the robotics domain, some notable research applications include Sensorfly a swarm of autonomous helicopters that could be used to search for humans during disasters the idea is that the swarm is better than one (Purohit et al., 2011). Swarm based computer systems are highly fault-tolerant because the failure of one component in a swarm does not affect the overall system. This makes them particularly suitable for hazardous or remote environments, and the military is also interested in swarm research, developing a swarm of Spybot’s known as MAST (“Micro Autonomous System Technologies (MAST),” n.d.)

Self-organisation is the interaction between four distinct elements, renewal, randomness, positive feedback, and negative feedback. (Goss et al., 1989) states "*It is important to note that the selection of the shortest branch is not the result of individual ants comparing the different lengths of each branch, but is instead a collective and self-organizing process*"

1. ANT COLONY OPTIMIZATION
   1. Introduction

In nature, ants live in a colony in which the number may vary from thirty to tens of millions to super colonies (Browne, n.d.) No one ant in charge, each individual ant has a particular job that they do very well, even the Queens function is only to lay eggs. All ants work for the success of the colony a truly social insect. So successful at it they have withstood one hundred million years and in the process colonised the earth (except for Antarctica and a few small island). The efficiently of ants finding and transporting food, overcoming obstacles, building anthills, and other operations is truly optimal. Success does not arise from leadership or the intelligence of a few ants, but emerges from the trial and error of many non-intelligent individuals in fact a single ant cannot survive outside of the colony for long. However the ant colonies as a system themselves is extremely robust: a reduction of up to 40% of insects has practically no effect on the functioning of the whole society [8]. A mass destruction of ants (for example, resulting from a chemical treatment of their habitat) leads to the consolidation of insects from the neighbouring anthills into one family to save the society. (Goss et al., 1989) (Hölldobler, 1990) {Citation}Ants are said to be self-organisation as stated by (Prabhakar et al., 2012) “*Social insect colonies operate without any central control. Their collective behavior arises from local interactions among individuals*.” (Bonabeau, 1999)

* + 1. Stigmergy

When searching for food ants initially explore areas around their colony randomly. They communicate with one another via a chemical substance called a pheromone in which other ants can smell. As an ant moves around it deposits a constant amount of this pheromone that other ants, then follow. Each ant moves in a random fashion. As soon as a particular ant finds food it carries some food on its back and returns to their colony (nest) releasing stronger concentrations of the pheromone proportional to the quality and quantity of the food source found. When another ant smells this pheromone the other ant simply decides to follow the previous path depending on the strength of the pheromone if it chooses to do so it will reinforce the existing trail, so that the next, ant that stumbles upon this trail will almost certainly follow it. Therefore, the more ants that travel along the path more attractive the path will be for the next ants. This is known as stigmergy discovered by French entomologist Pierre-Paul Grassé within ant colonies in 1959. (Bonabeau, 1999) With time, these pheromones evaporate, which allows the ants to adapt to their environment. Therefore an ant using the shortest path will return to the nest quicker therefore reinforcing the path more overtime more ants are able to complete the shorter route and more pheromone accumulate on the shorter path. This continuous random movement of each ant helps the colony find different routes. This also ensures that the ant can find its way around any obstacles that may become present in their path. As (Clark, 1997) defines Stigmergy ‘‘the use of environmental conditions as instigators of action and the overall ability of the group to perform problem-solving activity that exceeds the knowledge and the computational scope of each individual member’’ this is a good analogy when applied to computer science.

|  |
| --- |
| Figure 1 |

The asymmetric bridge in figure 1 was an experiment carried out in a laboratory using a colony of Argentine ants that uses pheromones to communicate proposed by (Goss et al., 1989) demonstrates how the social operation of ants allow them to find the shortest path. A bridge A,B,C,D was constructed. Point A is a gate, and is opened the number of ants travelling along route A,B,D and the route A,B,D, to the food and back to the nest was counted. At early stages of the experiment both routes were selected evenly. But after some time, the ants preferred the shorter route ABD as the quantity of pheromone on the shorter path is greater than that of the longer path.

The concept of stigmergy and randomness are key element in the implementation of ant colony optimisation algorithms. (Manjurul Islam et al., 2006) (Dorigo and Gambardella, 1997) (Dorigo et al., 1996)

* 1. Artificial Ants

Marco Dorigo. founded the field of Ant Colony Optimisation (ACO) in 1991 (Dorigo et al., 2006a) (Goss *et al*., 1989). ACO is a section of swarm intelligence in which researchers study the social behaviour of real ants described earlier in this chapter.

Ant colony optimisation introduces a meta-heuristic algorithm that uses artificial ants to find solutions to problems. Based upon the behaviour of the real ants in nature in which an individual ant is unable to hunt for food or communicate by itself. As it is part of the collective the ants together can solve complicated problems and can successfully operate their Colony.

|  |  |  |  |
| --- | --- | --- | --- |
| Year | Algorithm | Authors | Reference |
| 1991 | Ant System |  |  |
| 1992 | Elitist ant system |  |  |
| 1995 | Ants-Q |  |  |
| 1996 | Ant colony system |  |  |
| 1997 | MAX-MIN ant system |  |  |
| 1999 |  |  |  |
| 2000 |  |  |  |
| 2001 |  |  |  |

* 1. The ant system:

This was the first algorithm of the ant colony optimisation algorithms proposed by Marko Dorigo. It is inferior to all other ant colony optimisation algorithms according to most research. As being the first this algorithm was just used to prove the concept and is the basis for all other ACO algorithms.

The ant system is a new member in the class of these meta-heuristics, with more

famous members being simulated annealing (cf. e.g. [16]), genetic algorithms (cf.

e.g. [13], [15]), tabu search (cf. e.g. [11], [12]), neural networks (cf. e.g. [14], [17])

and evolution strategies (cf. e.g [20]). Most of these methods (except tabu

search) have been derived from nature, and the same is true for the ant system,

that was developed more recently by Colorni, Dorigo and Maniezzo (cf. [4], [5],

[9])

compered with more famous members being simulated annealing (cf. e.g. [16]), genetic algorithms (cf.e.g. [13], [15]), tabu search (cf. e.g. [11], [12]), neural networks (cf. e.g. [14], [17]) and evolution strategies (cf. e.g [20]). Most of these methods (except tabu search) have been derived from nature, and the same is true for the ant system, that was developed more recently by Colorni, Dorigo and Maniezzo (cf. [4], [5],

[9])

The idea of the ant system is based on the following observation. A colony of

ants is able to succeed in a task (for instance to nd the shortest path between

the nest and the food source) whereas a single ant would probably fail, especially

as ants are almost blind. It was found that ants leave a trail of pheromone when

they move. This pheromone trail can be observed by other ants and motivates

them to follow the path, i.e. a randomly moving ant will follow the pheromone

trail with high probability. That is the way how the trail is reinforced and more

and more ants follow that trail. The following example (cf. [10]) shows how over

time, short paths are found through this self-reinforcing process.

Suppose that the ants commute between food source A and nest E with

speed one, i.e. they need one time unit for a distance of length one (cf. Figure

In every time period 30 ants leave the nest and the food source, respectively.

The only difference in the preceding algorithms is the rate of evaporation, and how they handle pheromones. Initially all routes are initialised with some initial level of pheromone, and some heuristic that helps find a good solution at the beginning.

* 1. global parameters

ACO, in common with other meta-heuristics based on natural systems, is quite sensitive to your choice of global parameters—alpha, beta and so on as shown in the above (algorithm 1) Although there has been quite a bit of research on ACO parameters, the general consensus is that you must experiment a bit with free parameters to get the best combination of performance and solution quality.

|  |
| --- |
|  |
| Algorithm 1 |

Here

**sp -** partial solution #p

**N** - set of all paths from the city ito all adjacent cities still not visited by the ant

**cij** - path from the city **i** to the city **j**

**p -** probability

**tij** - amount of pheromone on the path **cij**

**hij** - some heuristic factor, usually Description: 644067/AntSysTrav_0.PNG, where **dij**is a distance between cities **i** and **j, Q** is some constant

**α** and **β** - algorithm parameters.

The ants update pheromones on paths connecting the cities according to the formula:

Description: 644067/AntSysTrav_5.PNG

where

**m** - number of ants,

Description: 644067/AntSysTrav_11.PNGif the ant **k** travelled the path Description: 644067/AntSysTrav_13.pngbetween cities i and j; **Q** is some constant, and **Lk** is the length of the **kth** ant's travel

Description: 644067/AntSysTrav_12.PNG0 otherwise.

After all ants complete their nth trip, evaporation is applied to all paths between cities:

Description: 644067/AntSysTrav_8.PNG(3)

Here **Description: AntSysTrav_9**is the evaporation factor.

There is a question of when to apply global evaporation: immediately after all ants complete their trip, or after all pheromones on graph edges (paths between cities) are updated? Evaporation before updates is, essentially, increase of updating pheromones. (Dorigo and Socha, 2006a) to pseudo code for the above algorithm and elitist system is shown in Figure 2

|  |
| --- |
| *Algorithm AS-TSP*  *initialize all edges to (small) initial pheromone level τ0;*  *place each ant on a randomly chosen city;*  *for t := 1 to t\_max do*  *for k := 1 to m do*  *build a tour T(k,t) for ant k by applying the probabilistic transition rule;*  *end;*  *if (best T(k,t) better than current solution) update current solution to T(k,t);*  *for every edge (i,j) do*  *apply pheromone update;*  *end;*  *// the next block simulates a special “elitist ant” and is optional*  *Let b be the ant with the best tour*  *for all edges egdes in best tour T(b,t) do*  *apply additional pheromone increment with Q/length(T(b,t));*  *end*  *end.* |

Figure 3 . Ant colony system pseudo code

* 1. Elitist ant’s algorithm:

The quality of the solutions produced by the ant system could be improved using

so-called elitist ants (cf. [10]). The idea of the elitist strategy in the context of

the ant system is to give extra emphasis to the best path found so far after every

iteration. When the trail levels are updated (cf. Formula (2) in Section 2.2),

this path is treated as if a certain number of ants, namely the elitist ants, had

chosen that path. As it is likely that some edges of that path are part of the

optimal solution, the aim is to guide the search in succeeding iterations. The

updating of trail levels therefore is done in the following way:

The concept of elitism can also be found in genetic algorithms. In general,

in a genetic algorithm the \_test individual of a generation (best solution of an

iteration) will with positive probability not be included in the next generation,

if the genetic operators selection, recombination and mutation are applied. In

3In [10] the best results for a 30 city problem were obtained by using eight elitist ants.

6

that case the genetic information of that individual would be lost. Therefore,

the idea of elitism is to preserve the \_ttest individual of a generation. As a

consequence local search aspects (exploitation) become more important while

global search aspects (exploration) become less important (cf. eg. [13]). This

potential drawback is also valid for the ant system.

On the other hand, in the ant system algorithm

Equivalent to the ant system except the pheromone is updated slightly different and each iteration

Description: ElSysTrav_0

Here

Description: 644067/ElSysTrav_1.PNG  if the path **ij** is from **Tbs, Tbs** is the best to date round trip, **e** is an algorithm's parameter.

* 1. Rank-Based Ant System:

Equivalent to the ant system except for the pheromone update as before

Description: 644067/RankAntsTrav_1.PNG

**Q** is a constant, LR the length of the **rth** ant trip, **e** is the additional multiplier.

Again, the best results are at **Description: 644067/RankAntsTrav_3.PNG**The member Description: 644067/RankAntsTrav_2.PNG is similar to the Elitist Ant Algorithm (Bullnheimer et al., 1997) (Dorigo and Socha, 2006a)

* 1. Best-Worst Ant Algorithm

Equivalent to the ant system except for the pheromone update and only the best ants to date get to update

Description: 644067/BWASTrav_0.PNG

Description: 644067/BWASTrav_1.PNG  if the path Description: 644067/BWASTrav_5.PNG **,   
Tbs** is the best to date round trip, **Cbs** - length of the trip.

In addition, the paths of the round trip of the worst ant for the current iteration that are not in the best to date solution are subject to additional evaporation:

Description: BWASTrav)4.png

Here **ρw**is an additional evaporation factor for all Description: BWAS_Trav_6.png and Description: 644067/BWASTrav_7.PNG, **Tw** is the worst solution for the given iteration, and **Tbs** is the best to date solution. (Dorigo and Socha, 2006a)

* 1. Min-Max Ant System

There are variants in the selection of the ants allowed to update pheromones: the best to date ant, or the best for current iteration, There are min and max pheromone limits to the quantity of pheromone on the paths between cities, τmin and τmax. It is believed that prevents local loops. So, evaporation is: The solution construction is according to the equation (1).

So, evaporation is:

Description: 644067/MMASTrav_0.PNG

The update is:

Description: 644067/MMASTrav_1.PNG

Here Description: 644067/MMASTrav_2.PNG  if the path Description: 644067/MMASTrav_3.PNG  **Tsel** is a selected ant's round trip, Csel - is the length of the trip.

Some authors initiate pheromones to **τmax**. (Dorigo and Socha, 2006a)

Table 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | α | β | r | T0 |
| AS | 1 | 2 to 5 | .5 |  |
| EAS | 1 | 2 to 5 | .5 |  |
| ASrank | 1 | 2 to 5 | .5 |  |
| MMAS | 1 | 2 to 5 | .002 |  |
|  |  |  |  |  |

Here, n is the number of cities in a TSP instance. All variants of AS also require some additional to small TSP instances with up to 200 cities, good results are obtained by using always the iteration- best pheromone update rule, while on larger instances it becomes increasingly important to alter- nate between the iteration-best and the best-so-far pheromone update rules.

* 1. ANT COLONY SYSTEM
  2. LOCAL OPTIMAL
  3. STAGNATION MEASURES

When the pheromones on the ant trails become too unbalanced that is some edges have large quantities of pheromones, while others have none, making the search space biased that is all ants will follow more or less the same path. In order for the algorithm to detect this case Dorigo and Stützle proposed countermeasures.

* + 1. Variation Coefficient

The Variation Coefficient in probability theory is as ratio of standard deviation to the mean. Applied to Ant colony optimisation it is the standard deviation of the constructed pheromones and there average pheromones. The disadvantage of this method is that’s some ants can generate totally different solutions but the variation coefficient can be the same. However it is efficient and easy to implement.

* + 1. Branching Factor

The Branching Factor evaluates the pheromone matrix looking at every node and determinants the pheromone value that fulfils Equation 1. The Branching Factor is defined as the average of all branching factors. Informally representing the average number of routes, the ant has a high probability of selecting.

Equation 1

|  |
| --- |
|  |

Is the minimum value of pheromones and is the maximum. Dorigo and Stützle use λ with a value of 0.05 can be defined in the interval [0,1] Antsystem proposed next will use the same.

1. metaheuristic methods

Metaheuristics are a very general class of algorithms that are applicable to a wide variety of problems. Such an algorithm tries to continually

find and improve feasible solutions to a given (often combinatorial) optimization problem with the guidance of an underlying problemspecific

heuristic – hence the name. Usually a metaheuristic runs in iterations and in each iteration one or more solutions are generated

using the knowledge about the search space acquired during the previous iterations.5

* 1. The Genetic algorithm

The genetic algorithm (GA) was developed by John Holmes his colleagues and students of the University of Michigan in the 1970s, inspired by Darwin’s theory of evolution. There has been a lot of research since then. However despite the GA age, these algorithms are still actively studied widening the problems that can be solved using theses algorithm. Based on Darwin’s principle of evolution selection, crossover, and mutation the (GA) utilises these to traverse large search spaces. (GA) algorithms are widely used in a wide range of real-world problems (CITE)

* resource allocation, John shop scheduling timetable planning the TSP
* network routing satellite orbit selection
* machine learning

The algorithm operates on a population of individuals called chromosomes; containing a representation of potential solutions to the problem that is being analysed. Chromosomes are represented by a fixed length data array (strings, integers, etc.). Each data bit in the array is called a gene. Initially the population is initialised with a set of candidate solutions (the population) either with random chromosomes or by a heuristic and the fitness is calculated. At each iteration a new generation (population) is created via selection and crossover from the old population. Producing better solutions as better solutions are likelier to be combined with each other to process better solutions. The algorithm then manipulates a percentage of the new populations chromosomes via mutation to explore the search space (CITE)

Steps are as follows:

1. Initialisation , A population of random solutions (chromosomes) is created at this point is very important that the population is diverse as possible (randomness)
2. Fitness, each chromosome is rated and how well the chromosome solves the given problem.
3. Selection, the fittest chromosomes are selected to create new generation.
4. Select a percentage of high fitness chromosomes from the new generation to undergo crossover and mutation, with the new modified chromosomes inserted back into the population.
5. Repeat step 2 until terminated.
   * 1. Selection

Not all chromosomes make it into the next population, only a select few chromosomes gets selected and several selection operators exist. The main idea is the selection of fitter individuals for the next generation, thus improving the algorithm. The choice of selection operators can greatly influence the solutions generated by the algorithm (CITE) are many different forms of selection operators. Most common are Roulette Wheel Selection, tournament selection and elitism.

* + 1. Roulette Wheel Selection

A selection technique for choosing a chromosome proportional to its fitness as defined by: , where is the probability of choosing chromosome i, is the fitness of chromosome i, over the sum of all chromosome fitness . As Buckland states, (CITE) “Imagine that the population’s total fitness score is represented by a pie chart or roulette wheel. Now you assign a slice of the wheel to each member of the population. The size of the slice is proportional to that chromosome’s fitness score: the fitter a member is, the bigger the slice of pie it gets. Now, to choose a chromosome, all you have to do is spin the wheel, toss in the ball and grab the chromosome that the ball stops on.”

* + 1. Elitism

In elitism the complete population of chromosomes is sorted by its fitness. The probability that an individual is selected is inversely portion to its position in the sorted list; the individual chromosome at the head of the list is more likely to be selected.

(PROBLEMS)(CITE)

* + 1. Tournament selection

A selection technique that chooses a chromosome via a tournament, individual’s chromosome are selected from the population randomly (uniform probability), and the winning chromosome (the best fitness) is selected. Tournament size can be adjusted (selection pressure) to only allow a greater probability of the fitter chromosome to be selected. This method has several benefits, it’s easy to implement and can be run in parallel. However, in large populations, the tournament size has to be equally large to ensure that the higher fitness chromosomes have greater chance of selection (CITE)

* 1. Genetic operators
     1. Crossover

There are many different methods of crossover simplest being one point. A random point (gene) of two parent’s chromosomes is selected and these chromosomes are exchanged with their consecutive counterparts. To create two new children chromosomes as illustrated in Table 2. The procedure for the rating crossover is as follows creating crossover is as follows:

* Select two parents chromosomes (selection)
* Select random positions of one or more genes to create crossover points
* Copy the genes outside of the crossover point of parent’s one chromosome into child one.
* Copy the genes outside of the crossover point of parents two chromosome into child one.
* New child is created
* repeat the same process with parent two, creating child two

Table 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Random crossover point | | | | | | |
|  |  |  |  |  |  |  |
| Parent Chromosome 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Child Chromosome 1 | 1 | 0 | 1 | 1 | 1 | 1 |
|  |  |  |  |  |  |  |
| Parent Chromosome 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| Child Chromosome 2 | 0 | 0 | 0 | 1 | 0 | 0 |
|  |  |  |  |  |  |  |

* + 1. Multi point crossover

Similar to one point crossover except two or more random points are selected parents (Genes) as illustrated in

Table 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Random crossover points | | | | | | |
|  |  |  |  |  |  |  |
| Parent Chromosome 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Child Chromosome 1 | 1 | 0 | 1 | 1 | 0 | 0 |
|  |  |  |  |  |  |  |
| Parent Chromosome 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| Child Chromosome 2 | 0 | 0 | 0 | 1 | 1 | 1 |
|  |  |  |  |  |  |  |

These two variants of crossover are the most popular, but there are many different variants of this crossover operation that depend on the problem being analysed. Later we will analyse a special form of crossover to solve the problem of the travelling salesman. (INSERT CHAPTER NO)

* + 1. Mutation

Here we discuss the crossover and mutation operators as they are the core of the genetic algorithm, however, there are many different operators (CITE). A certain proportion μ (a low value usually less than 1%) of newly selected chromosomes created by the selection process earlier is selected randomly for mutation in uniform probability.

Then each of these chromosomes is mutated, a gene from the chromosome is selected randomly and swapped with its counterpart. As illustrated:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Mutation |  |
| Chromosome: | 11101001000 |  | 11111011000 |

The main function of the operators is to inject diversity into new population allowing the genetic algorithm for new solutions lowering the risk of being trapped in a local optimal.

* 1. Limitations and conclusions

As the genetic algorithm can converge to find an optimum solution can also converge to find a bad solution. Bad solutions can be caused by many factors, premature convergence or poor parameter settings or simply down to luck with randomness. However the methods of crossover and mutation try to address these problems. However, as most problems analysed by the generic algorithms are NP-complete we can properly never know the exact answer to the problem being solved.

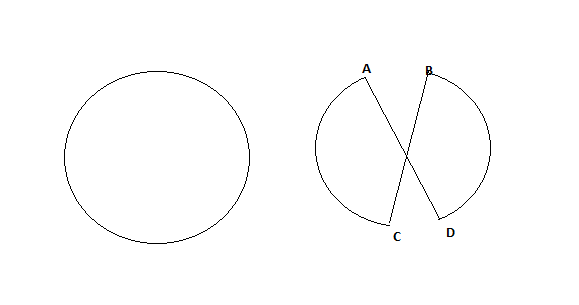
However generic algorithms can be great in solving problems when it is not possible to compute all possible solutions are we do not know the ways to reasonably solve the problem space like the travelling salesman problem.

1. Local Search

Once an initial solution is found to the problem space top researchers on meta-heuristics tell us that the best approach to obtaining a high quality solution is to combine with a local search.(CITE) All meta-heuristic algorithm defined earlier can optionally be extended with local search methods; this can greatly improve the solutions constructed by the algorithm’s even guiding the algorithm to the optimal solution more efficiently. The most common heuristics are 2-opt and 3-opt, and also meta-heuristic approaches like tabu search (Johnson and McGeoch, 1997). Local search has been applied to a large number of hard computational problems in computer science. Local search is implemented by changing certain positions on the vertices of the problem being analysed or in the case of the genetic algorithm the genes to find a more local solution.

* + 1. 2 2-opt

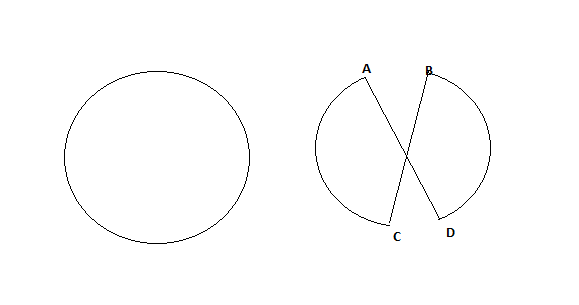
2-opt is a simple local search algorithm proposed by Croes in 1958 given a set of edges from the initial solution then delete any two edges, thus breaking the tour into two paths, and then reconnects those paths in the other possible way so that the route is still valid (Johnson and McGeoch, 1997).In the case of TSP if the reinsertion is done only if the new route is shorter. This is continued until no further optimisation is valid. And the final solution called 2 optimal as shown:



* + 1. 3Opt

The principle of 3-opt is the same as 2opt except instead of two edges removed there are three until there are no three possible moves left. ((Johnson and McGeoch, 1997)

(Johnson, 1990)



* + 1. [Tabu search](http://en.wikipedia.org/wiki/Tabu_search)

[Tabu search](http://en.wikipedia.org/wiki/Tabu_search) proposed by (Glover, 1989) uses both 2-opt and 3-opt searches together

best solution is selected. However, the algorithm maintains a certain number of previous moves and adds them that the so-called tabu-list. Therefore a previous move will not be repeated thus preventing the algorithm for moving in a circular pattern.

* + 1. Local search Applied to ACO and GA

Local search meta-heuristic suffer from the problem off converging at local minima if used by themselves however the solutions for local search in our context are generated via meta-heuristic algorithms, informally local searches are bad creating initial solutions and the quality of the final solution depends upon the initial solution. However and colony optimisation and the genetic algorithm are great creating initial solutions. (CITE) and combined with local search can greatly enhance there performance.

In the case of the algorithm descried in chapter XXXX there are different methods of applying local search, that is we can apply to all solutions in a iteration, or only the iteration best solution, or to the global best solution (the best so far solution). There is also the question of Lamarckian versus Darwinian updates.

This states that given a set of solutions *Sp*that has been applied to a local search procedure becomes *Slp* which solutions get introduced into the next irritation of the algorithm. This can be explained easily in the case of ACO there are two ways of updating the pheromones after a local search that is either with *Sp* (Darwinian the original) or *Slp .* (Lamarckian local search applied). There are arguments for and against each. (CITE). Lamarckian seem to have the better argument that is if there is a better tour then it would be as Diagro states “it would be stupid to use the worst tour”. Darwinian view is that meta-heuristic algorithms with local search with Lamarckian updates just learn how to generate good solutions for the local search procedure. The researched consensus that Lamarckian out preforms Darwinian. Experimental results of these approaches by the antSolver is also defined in (CHAPTERXX)

However in ACO with local search the use of acoustics values are not that important, (CITE ACO HEURTICS)

* 1. Conclusion

This chapter …..

# Np-complete

* 1. Introduction

As legend has it, the man that invented chess, the king at the time loved it and asked the man. How can I repay you? The inventor just replied I’m not a greedy man can you can place on the first square of my chessboard one grain of wheat and keep doubling that for each sequential square so on square two there will be two pieces of grain and on square three four pieces. The king was happy enough with this however when it gets to square 64 on the chessboard that properly accounts for more grain than is ever been in existence. 264 grains.

When optimisation problems are difficult to solve and cannot be solved efficiently or quickly are said to be NP-Complete. A problem is defined NP if it is solvable in polynomial time as the number of combinations increases exponentially as the problem increases. The theory of NP-Complete classifies problems based upon difficulty. NP hard is defined as given the worst possible input (N) “the worst case” the effort to find a solution increases exponentially.

* 1. O-notation

Algorithms are designed to work with an input *(N).* And usually the efficiency is defined as a function of *N f(N)* in either time complexity , space complexity or both. Efficiency analysis is mostly interested in the average case, the amount of resources expected an algorithm takes on typical input *(N)* , and in the worst case, the amount of resources an algorithm would use based upon the worst possible input *(N)*. Most algorithms are well researched to the point that accurate mathematical models are available to define the average case and worst possible case. The worst case efficiency is defined as computational complexity. That is, we can say given an input *(N)* the efficiency is proportional to. It does not matter what kind of computer the algorithm is executed on whether it is a supercomputer or laptop. The performance is proportional to *(N),* this is called O-notation. Defined as:

*A function g(n) is said to be O(f(N) if their exists Constants Co and No such that g(N) is less than cof(N) for all N > No.*

Most algorithms can be sensitive based on the input *(N)* and the average case can vary in real-world implementations and the worst case scenario may never appear.

* 1. TRAVELLING SALESMAN PROBLEM

The travelling salesperson problem (TSP) is the most famous kinds of NP hard problem (Cai, 2008). A salesperson intends to visit a number of cities and can only visit a city once, and returning to the start point minimising the distance they have to travel. (Karp, 1977). In more formal terms, TSP can be defined as a “*Hamiltonian tour of minimum length on a fully connected graph*”.

In the case of the travelling salesperson problem in which adding one more city grows exponentially. As demonstrated in “table 1” TSP for four cities would require five-fold more steps just to add an additional city or 50 city’s would require a thousand-fold steps would require to add an extra ten cities clearly a brute force calculation becomes impossible as a number of cities increase this problem is said to be NP-hard (Karp, 1977) that is no algorithm exists that can find the solution quickly and comparing each data must be implemented “exhaustive search “ to find a solution. NP-hard meaning that the time required solving it increases very quickly with the problem size. “*An exponential time algorithm cannot be guaranteed to work for all problems of size of 200 (say) are greater because no one can wait for an algorithm to take 2200 steps to complete, regardless of the speed of the computer exponentially growth the dwarfs technology changes*” (Sedgewick, 1998) Clearly a better solution is needed and that where evelouency algorthmins come in. These algorathms provides an optimisation approach to “best guess” a solution. These approximation algorithms “*trade optimality for efficiency*” (Dorigo and Socha, 2006b) But have the advantage of finding a good solution quickly. However, the problem with optimisation algorithms is to be stuck in locally optimiser loop discussed later.

The travelling salesperson problem (TSP) was first formulated as a mathematical problem in 1920's some people credit Karl Menger with popularising the problem to the mathematical community (StreetChicago, n.d.). The travelling salesperson problem is one of the most studied problems in combinational optimisation and is often used for testing optimisation algorithms. TSP can be applied to many fields from transport to genetics (“TSP Gallery,” n.d.). The TSP has several applications as stated by (Geetha et al., 2009) “*planning, logistics, and the manufacture of microchips, scheduling of service calls at cable firms, the delivery of meals to home bound persons, the scheduling of stacker cranes in warehouses, the routing of trucks for parcel post pickup, and a host of others.*” So solving the TSP problem is off academic interest and importance

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **City Number** | **Number of steps to compute** | | **1** | 1 | | **2** | 1 | | **3** | 6 | | **4** | 24 | | **5** | 120 | | **10** | 3,628,800 | | **20** | 243,290,200,817,664,000 |   Figure 4  **TSP Example** | Figure 5  TSP 8,246 towns/cities in Ireland (“8,246 populated locations in Ireland,” n.d.) |

* 1. locally optimiser loop
  2. ANT COLONLY OPTIMIZATION APPLIED TO TSP

In the case of ant colony optimisation (ACO) was the first problem solved by Dorigo inventor of the ACO algorithm (Dorigo and Gambardella, 1997) . Each virtual ant starts at a randomly city and travels from city to city until all cities are visited exactly one. At the last visited city the virtual ant returns to its start city. At each choice the ants select its next city via the following algorithm. (Dorigo et al., 1996) The problem can be easily applied to ants as the movements of ants and the traveling of the salesperson are similar to that of ants. The ant colony optimisation algorithm is well researched solving the travelling salesperson problem as detailed in the projects background section of this document. The travelling salesperson problem is defined as “*Given a list of cities and the distance between each city what is the shortest possible route that visits each city exactly once, and returns to the starting city*”. (Karp, 1977)

* 1. Virtual ant’s

Virtual ant’s agents are considered as autonomous in the sense an independent travelling salesperson solving his own problem and cannot see the overall problem. Firstly ACO is iteration algorithm, during one algorithm iteration each virtual ant completes one route in the travelling salesperson problem that is one city to another, but the ant is not allowed to visit a city already visited. Virtual ant chooses the next city to visit based upon the particular algorithm implemented. Each virtual ant deposits pheromone on this route. The amount dropped depends upon the quality of the route and like real ants this pheromone evaporates based to upon the rules of the algorithm implemented. (Dorigo et al., 2006)

The Ant colony optimisation algorithm (ACO) is well suited in finding the shortest path as discussed. TSP was the first problem solved by ant algorithms, the act colony optimisation program can generate good solutions to this the TSP NP-Hard problem. Some other NP problems that have been solved with ACO include, hyper-spectral imaging with good results (Zhang et al., 2013) , also applied to the n-Queen problem (Khan et al., 2009). The Vehicle Routing Problem a ﬂeet of vehicles that have to deliver items to customers from a depot such that the total delivery time is minimized. (Narasimha et al., 2012) ACO combined with a neural networks can be used to solve continuous optimisation problems, that is that the inputs constantly change (Pour et al., 2008) this continuous approach can also be applied to network routing as discussed by (Sim and Sun, 2003) TSP can be applied to many fields from transport to genetics (“TSP Gallery,” n.d.)

1. SOFTWARE ENGineering

There are many design methodologies for estimating the quality of software, when implemented correctly within software and design patterns can completely enhance the reusability, modularity, abstract, understandability and maintainability, design patterns guarantee reusable objects by creating abstract objects avoiding dependencies on any part of the system and thus avoiding tight coupling between these objects (Gamma, 1995) the bridge pattern is a classic example . “*it is useful to design programs with design patterns even if the actual design problem is simpler than that solved by the design patterns*” (Prechelt et al., 2001) The abstracts factory pattern is also a good example is each object can be called the same way but the object construction depends on the run time of the system. New factories can be added that are completely decoupled from the client code. That is an abstract factory can be used to implement different graphical user interfaces on the same system using the same client but displaying a completely different GUI that is detected at run-time. (Gamma, 1995) In addition, the decorator pattern is also a good example on how an object can be decorated as such; fundamentally the decorator pattern wraps the image object with additional functionality and can remove functionality at run time without changing the structure of the class. (Huaxin and Shuai, 2011) Although can also be error prone as you can add the same functionality more than once. Implementing design patterns makes for a more flexible modular design. Various researchers analyse software quality by the level of design patterns implemented in source code (Khaer et al., 2007) (Zhang and Budgen, 2012).

* 1. Data Structures
  2. Antsystem classes
  3. Interface
  4. Factory
  5. Implementing Genetic Algorithm
  6. Conclusion

This chapter ….

1. ???
   1. Introduction

This chapter will …..

* 1. ???
  2. Conclusion

This chapter ….

1. ???
   1. Introduction

The focus of this chapter will be …..

* 1. ???
  2. Conclusion

This chapter ……

1. Experimentation & Evaluation
   1. Introduction

To analyze their quality, the three versions of the ant system algorithm presented in this paper, i.e. AS, ASelite and ASrank, were applied to ve dierent TSP instances: a 30 city instance from literature5 and four real-life problems from an industrial application with 57, 80, 96 and 132 cities, respectively. For reasons of comparability, we furthermore applied simulated annealing, probably the most classical meta-heuristic, and a genetic algorithm, another population based method, to the ve test problems. In literature, a wide variety of theoretical as well as applied research on simulated annealing (cf. eg. [1], [18]) and genetic algorithms (cf. eg. [8], [13]) can be found. For that reason we only brie present the details of the implemented algorithms in the following two sections.

* 1. Experimentation
     1. Genetic algorithm configuration

The population size of the genetic algorithm was 10 individuals. The initial

population was generated with the nearest neighbor heuristic, beginning with

a randomly chosen city for each individual. A generation replacement scheme

with an elitist strategy (the best member of each generation always survived)

was used, and individuals were selected through rank selection. In cases of

mutation (pmut = 0:85), the same operators as in simulated annealing were

used, in cases of recombination (prec = 0:1), partially mapped crossover and

uniform order-based crossover were used with equal probability. In 5% of the

cases the individual was copied into the next generation without being changed

|  |
| --- |
| for all members of population  sum += fitness of this individual  end for  for all members of population  probability = sum of probabilities + (fitness / sum)  sum of probabilities += probability  end for  loop until new population is full  do this twice  number = Random between 0 and 1  for all members of population  if number > probability but less than next probability  then you have been selected  end for  end  create offspring  end loop |

* 1. Evaluation
  2. Conclusion

This chapter …..

1. Conclusion
   1. Introduction

As discussed earlier the primary goal of this research, an empirical analysis on different ant colony optimisation algorithms in solving the travelling salesperson problem. (Lissovoi and Witt, 2013) (Fejzagic and Oputic, 2013) (Song et al., 2006) (Pour et al., 2008) The travelling salesperson problem was chosen because of the following characteristics, it is a very difficult NP problem, and it has been particularly well studied by researchers. In addition TSP is easy to understand. It is a traditional benchmarking problem for combinational optimisation methods and there is a big library of travelling salesperson problems and methods including solutions which make it possible to compare efficiency of the ant’s algorithms with other approaches.

* 1. Research Definition & Research Overview
  2. Contributions to the Body of Knowledge
  3. Experimentation, Evaluation and Limitation
  4. Future Work & Research
  5. Conclusion

???

BLIOGRAPHY

BI8,246 populated locations in Ireland [WWW Document], n.d. URL http://www.math.uwaterloo.ca/tsp/world/eipoints.html (accessed 5.6.14).

Beni, G., Wang, J., 1993. Swarm Intelligence in Cellular Robotic Systems, in: Dario, P., Sandini, G., Aebischer, P. (Eds.), Robots and Biological Systems: Towards a New Bionics?, NATO ASI Series. Springer Berlin Heidelberg, pp. 703–712.

Bonabeau, E., 1999. Editor’s Introduction: Stigmergy. Artif. Life 5, 95–96. doi:10.1162/106454699568692

Browne, G., n.d. Sydney eScholarship Repository [WWW Document]. URL http://ses.library.usyd.edu.au/handle/2123/11091 (accessed 7.22.14).

Bullnheimer, B., Hartl, R.F., Strauss, C., 1997. A new rank based version of the Ant System. A computational study.

Cai, Z., 2008. Multi-Direction Searching Ant Colony Optimization for Traveling Salesman Problems, in: International Conference on Computational Intelligence and Security, 2008. CIS ’08. Presented at the International Conference on Computational Intelligence and Security, 2008. CIS ’08, pp. 220–223. doi:10.1109/CIS.2008.151

Clark, A., 1997. Being There: Putting Brain, Body, and World Together Again. MIT Press.

Dorigo, M., Birattari, M., Stutzle, T., 2006a. Ant colony optimization. IEEE Comput. Intell. Mag. 1, 28–39. doi:10.1109/MCI.2006.329691

Dorigo, M., Birattari, M., Stutzle, T., 2006b. Ant colony optimization. IEEE Comput. Intell. Mag. 1, 28–39. doi:10.1109/MCI.2006.329691

Dorigo, M., Gambardella, L.M., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 1, 53–66. doi:10.1109/4235.585892

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B Cybern. 26, 29–41. doi:10.1109/3477.484436

Dorigo, M., Socha, K., 2006a. An introduction to ant colony optimization.

Dorigo, M., Socha, K., 2006b. An introduction to ant colony optimization.

Fejzagic, E., Oputic, A., 2013. Performance comparison of sequential and parallel execution of the Ant Colony Optimization algorithm for solving the traveling salesman problem, in: 2013 36th International Convention on Information Communication Technology Electronics Microelectronics (MIPRO). Presented at the 2013 36th International Convention on Information Communication Technology Electronics Microelectronics (MIPRO), pp. 1301–1305.

Gamma, E., 1995. Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, Mass.

Geetha, R.R., Bouvanasilan, N., Seenuvasan, V., 2009. A perspective view on Travelling Salesman Problem using genetic algorithm, in: World Congress on Nature Biologically Inspired Computing, 2009. NaBIC 2009. Presented at the World Congress on Nature Biologically Inspired Computing, 2009. NaBIC 2009, pp. 356–361. doi:10.1109/NABIC.2009.5393321

Glover, F., 1989. Tabu Search—Part I. ORSA J. Comput. 1, 190–206. doi:10.1287/ijoc.1.3.190

Goss, S., Aron, S., Deneubourg, J.L., Pasteels, J.M., 1989. Self-organized shortcuts in the Argentine ant. Naturwissenschaften 76, 579–581. doi:10.1007/BF00462870

Hölldobler, B., 1990. The Ants. Harvard University Press.

Huaxin, M., Shuai, J., 2011. Design patterns in software development, in: 2011 IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS). Presented at the 2011 IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS), pp. 322–325. doi:10.1109/ICSESS.2011.5982228

Johnson, D.S., 1990. Local optimization and the Traveling Salesman Problem, in: Paterson, M.S. (Ed.), Automata, Languages and Programming, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 446–461.

Johnson, D.S., McGeoch, L.A., 1997. The traveling salesman problem: A case study in local optimization. Local Search Comb. Optim. 1, 215–310.

Karp, R.M., 1977. Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane. Math. Oper. Res. 2, 209–224. doi:10.1287/moor.2.3.209

Khaer, M.A., Hashem, M.M.A., Masud, M.R., 2007. An empirical analysis of software systems for measurement of design quality level based on design patterns, in: 10th International Conference on Computer and Information Technology, 2007. Iccit 2007. Presented at the 10th international conference on Computer and information technology, 2007. iccit 2007, pp. 1–6. doi:10.1109/ICCITECHN.2007.4579432

Khan, S., Bilal, M., Sharif, M., Sajid, M., Baig, R., 2009. Solution of n-Queen problem using ACO, in: Multitopic Conference, 2009. INMIC 2009. IEEE 13th International. Presented at the Multitopic Conference, 2009. INMIC 2009. IEEE 13th International, pp. 1–5. doi:10.1109/INMIC.2009.5383157

Lissovoi, A., Witt, C., 2013. Runtime Analysis of Ant Colony Optimization on Dynamic Shortest Path Problems, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO ’13. ACM, New York, NY, USA, pp. 1605–1612. doi:10.1145/2463372.2463567

Manjurul Islam, M.M., Waselul Hague Sadid, M., Mamun Ar Rashid, S.M., Kabir, M.J., 2006. An Implementation of ACO System for Solving NP-Complete Problem; TSP, in: International Conference on Electrical and Computer Engineering, 2006. ICECE ’06. Presented at the International Conference on Electrical and Computer Engineering, 2006. ICECE ’06, pp. 304–307. doi:10.1109/ICECE.2006.355632

Micro Autonomous System Technologies (MAST) [WWW Document], n.d. URL http://www.mast-cta.org/ (accessed 5.6.14).

Narasimha, K.S.V., Kivelevitch, E., Kumar, M., 2012. Ant Colony optimization technique to Solve the min-max multi depot vehicle routing problem, in: American Control Conference (ACC), 2012. Presented at the American Control Conference (ACC), 2012, pp. 3980–3985.

Pour, H.M., Atlasbaf, Z., Mirzaee, A., Hakkak, M., 2008. A hybrid approach involving artificial neural network and ant colony optimization for direction of arrival estimation, in: Canadian Conference on Electrical and Computer Engineering, 2008. CCECE 2008. Presented at the Canadian Conference on Electrical and Computer Engineering, 2008. CCECE 2008, pp. 001059–001064. doi:10.1109/CCECE.2008.4564699

Prabhakar, B., Dektar, K.N., Gordon, D.M., 2012. The Regulation of Ant Colony Foraging Activity without Spatial Information. PLoS Comput Biol 8, e1002670. doi:10.1371/journal.pcbi.1002670

Prechelt, L., Unger, B., Tichy, W.F., Brossler, P., Votta, L.G., 2001. A controlled experiment in maintenance: comparing design patterns to simpler solutions. IEEE Trans. Softw. Eng. 27, 1134–1144. doi:10.1109/32.988711

Purohit, A., Sun, Z., Mokaya, F., Zhang, P., 2011. SensorFly: Controlled-mobile sensing platform for indoor emergency response applications, in: 2011 10th International Conference on Information Processing in Sensor Networks (IPSN). Presented at the 2011 10th International Conference on Information Processing in Sensor Networks (IPSN), pp. 223–234.

Sedgewick, R., 1998. Algorithms in C++, Parts 1-4: Fundamentals, Data Structure, Sorting, Searching, Third Edition, 3 edition. ed. Addison-Wesley Professional, Reading, Mass.

Shtovba, S.D., 2005. Ant algorithms: theory and applications. Program. Comput. Softw. 31, 167–178.

Sim, K.M., Sun, W.H., 2003. Ant colony optimization for routing and load-balancing: survey and new directions. IEEE Trans. Syst. Man Cybern. Part Syst. Hum. 33, 560–572. doi:10.1109/TSMCA.2003.817391

Song, X., Li, B., Yang, H., 2006. Improved Ant Colony Algorithm and its Applications in TSP, in: Sixth International Conference on Intelligent Systems Design and Applications, 2006. ISDA ’06. Presented at the Sixth International Conference on Intelligent Systems Design and Applications, 2006. ISDA ’06, pp. 1145–1148. doi:10.1109/ISDA.2006.253773

StreetChicago, I.I. of T.S.F., n.d. About Karl Menger [WWW Document]. URL http://science.iit.edu/applied-mathematics/about/about-karl-menger (accessed 5.3.14).

TSP Gallery [WWW Document], n.d. URL http://www.math.uwaterloo.ca/tsp/gallery/index.html (accessed 5.6.14).

Zhang, B., Gao, J., Gao, L., Sun, X., 2013. Improvements in the Ant Colony Optimization Algorithm for Endmember Extraction From Hyperspectral Images. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 6, 522–530. doi:10.1109/JSTARS.2012.2236821

Zhang, C., Budgen, D., 2012. What Do We Know about the Effectiveness of Software Design Patterns? IEEE Trans. Softw. Eng. 38, 1213–1231. doi:10.1109/TSE.2011.79

Appendix A