# Development of evolutionary based techniques with applications to engineering.

Pelley, Carwyn Lloyd


SWANSEA UNIVERSITY

# Development of evolutionary based techniques with applications to engineering

*Author:*
Carwyn Lloyd PELLEY

*Supervisor:*
Dr. Mauro Sebastián INNOCENTE
Prof. Johann SIENZ

SUBMITTED TO SWANSEA UNIVERSITY IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF PHILOSOPHY

Year of Submission: 2013

# Summary

Every possible problem can be considered to have a set of possible states by which amongst them, some are considered better than others by some chosen measure. It is the intention of optimisation to discover such states that perform better than all others for any given problem. It is an important tool within an array of subject areas, arguably all, in particular engineering, which tackles such applications as shape optimisation and industrial scheduling to name but a few.

The aims of this work, are to increase the performance of the in-house general-purpose particle swarm optimiser designed at the department of engineering at Swansea University. This is to be achieved through its hybridisation with a local search, considering both solution refinement and early triggering mechanisms.

In the discrete domain, an ant colony algorithm is to be chosen and evaluated by way of a parameter study and comparison against other leading ant colony algorithms made for the purpose of development for the future application to scheduling problems.

Objectives are achieved through the increased refinement properties of the particle swarm optimiser with its hybridisation with local search. Additionally, an early switching mechanism is derived for the local search, resulting on average in a 20% reduction in the number of function evaluations required for constrained problems. With the highly unpredictable responses to unconstrained problems, only stagnation measures are derived. This study bridges the gap between the in-house optimiser and other hybrid particle swarm techniques available in the literature, resulting in competitive performance.

An extensive literature review of ant colony identified the population-based ant colony algorithm (PACO) for further investigation. A detailed parameter study is conducted, resulting in the realisation of the strongly coupled parameters present. Following this, a hybrid off-line tuning method is devised, hybridising a simple particle swarm optimiser with the ant colony algorithm, resulting in an overall better performing algorithm. This indicated clear strengths in some cases over the more popular of ant colony algorithms.

# Acknowledgements

I would first like to thank the EPSRC for their funding of this project together with my first supervisor Prof. Johann Sienz, for making this opportunity available to me. I owe my deepest gratitude to Dr. Mauro Innocente for our long discussions concerning this research and his continual guidance and support, for which this thesis would not have been possible without. I would also like to show my gratitude to Dr Jason W. Jones for his technological guidance throughout this project.

I would also like to show my gratitude to colleagues and friends Mr. Bruce Jones and Mr. Sean Walton for our frequent discussions concerning this work and my wife Mrs Rachel Pelley, for her continued support during this time, where again this thesis would not have been possible without them.

# Declaration and Statements

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: ..... (candidate)

Date (dd/mm/yyyy): 28/02/13

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed: ..... (candidate)

Date (dd/mm/yyyy): 28/02/13

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed: ..... (candidate)

Date (dd/mm/yyyy): 28/02/13

# Contents

# List of Figures

# List of Tables

# Nomenclature

ACO Ant Colony Optimisation

ANN Artificial Neural Networks

ATSP Asymmetric TSP

CC Credibility count

cg Centre of Gravity

COP Combinatorial Optimisation Problem

DE Differential evolution

DMS-PSO Dynamic Multi-swarm Optimiser

EA Evolutionary Algorithms

EC Evolutionary computation

EP Evolutionary programming

ES Evolutionary Strategies

ES Evolutionary strategies

FR Feasibility Ratio

GA Genetic algorithms

gbest Global best solution

GP Genetic programming

GP-PSO General-Purpose

GP-PSO-SQP General-Purpose Particle Swarm Optimisation, hybridised with SQP local search

IQR Interquartile Range

KISS Keep It Simple Stupid

lbest Neighbourhood best solution

LH Latin-Hypercube

MA Memetic algorithms

NN Neural Networks

OA Omicron ant colony optimisation

PACO Population-based ant colony optimisation

pbest A particles own personal position

PESO+ Particle Evolutionary Swarm Optimisation Plus

popbest Population of personal best solutions

popcur Current population solutions

PSO Particle Swarm Optimisation

RNG Random Number Generator

SA Simulated annealing

SACO Simple Ant Colony Optimiser

SQP Sequential Quadratic Programming

STSP Symmetric TSP

TS Tabu search

TSP Travelling Salesman Problem

# Chapter 1

# Introduction

## 1.1 Background

Optimisation is a multi-disciplinary field, receiving attention from mathematicians, engineers, social-biologists to economists, to name but a few [2].

The field of optimisation at the department of Engineering at Swansea university has developed somewhat recently. Over the last few years, a competitive in-house general-purpose particle swarm optimiser has been developed by Innocente [3] for the purpose of continuous optimisation problems.

Interest lies primarily in both the continuous and discrete domains. In the former, non-linear continuous function optimisation is of particular interest, allowing the effective solving of real-world engineering problems. Real-world problems quite readily become intractable and no longer adhere to easily obtainable solutions, often requiring a computational cost which exceeds reasonable limits. To overcome this, methods which are no longer deterministic and often take inspiration from natural or biological phenomenon are used, presenting a quick and efficient solution to the user through the sampling of only suggestible regions of the search space [4][1]. Since only a small fraction of the search space is sampled, we can no longer guarantee global optimality, however, optimal or near optimal solutions with reasonable computational costs result. Such methods are often called approximate or heuristic approaches, one of the main topics of this thesis [5].

Similarly, the discrete domain is important, in particular for routing and assignment problems. These are common to engineering, especially within manufacturing [5]. Tra-

[1]The search space consists of all possible $n$-dimensional solutions to a given $n$-dimensional problem.

ditional methods lend themselves to be powerful and effective for particular problems, however heuristic approaches allow the user to tackle a much greater range of problems within minimal intervention, whilst again only sampling a small subset of the search space [5].

It is to this end that this thesis is hoped to further develop.

## 1.2 Motivation

An overwhelming number of processes which are observable everyday are semi-static or clearly dynamic[2] optimisation problems (in fact all) [7]. Three examples are given here: the evolution of animal species or micro-organisms, who compete for survival in a changing environment; a species of ant, gathering food, minimising their path length towards this source; and the formulation of crystalline structures as the material cools, in search of the minimum energy state. All have in common, that they are decentralised[3] and for all intense and purpose, are non-deterministic [5][4].

For this reason, it is clearly advantageous to use non-deterministic decentralised systems, which optimise as a result of an emergent behaviour rather than a designed one. This enables them to encapsulate a much wider number of problem types with little to no intervention/adjustment from the user [7]. Such algorithms can generate any solution at any given time-step (iteration) with statistical likelihood that tends to favour the more likely better performing solution areas of the search space. Both particle swarm and ant colony paradigms have such characteristics, offering to succeed as general purpose optimising tools, where traditional methods clearly fail [3].

## 1.3 Objectives

This thesis is concerned with the perspective of learning and development within the optimisation field, in particular, the two most applicable of engineering domains are: continuous optimisation and combinatorial optimisation problem types.

This thesis intends to further enhance the in-house particle swarm optimiser through the incorporation of a local search (hybridisation), a natural progression of the algo-

[2]Dynamic optimisation problems are those where the search space changes over time [6](p.402).

[3]No central control mechanism, driving the task toward optimality

[4]Non-deterministic is described here as the use of such mechanisms which cannot yet be understood or predicted and as such, appear entirely random. A pseudo-random (stochastic) element is introduced to simulate such unpredictable responses.

rithm. Following this, a suitable method for early switch-over is to be derived, allowing an efficient transfer from global to local search.

Finally, this thesis intends to enhance an ant colony algorithm suitable for scheduling applications by performing a parameter study of this algorithm. Following this, the results of this parameter study are to allow the algorithm to be enhanced from its original set-up.

## 1.4 Layout of the thesis

The thesis is subdivided into five parts: part I concerns itself with the introduction to the field of optimisation; part II concerns itself with the background and development within the particle swarm paradigm; part III contains an extensive literature review of the ant colony optimisation paradigm, with a study of a particular ant colony algorithm; part IV draws overall conclusions; and part V is an appendix (see below for further details).

**Part I:** comprises of a chapter on the introduction and categorisation of the field of optimisation (chapter: 2). The remaining chapters describe common optimisation methods, from traditional (chapter: 3), trajectory (chapter: 4) to population-based (chapter: 5), with the intention of describing their original formulation, applied applications and developments since their original design. Finally a chapter on those algorithms with connection/similarity with ACO (chapter 6).

**Part II:** comprises of two main themes: chapter 7 is concerned with the detailed description of the 'particle swarm optimisation' paradigm. The application and hybridisation of the SQP local search with the particle swarm is then made in chapters 8-11.

**Part III:** comprises of: A chapter which consists of a extensive literature review of the 'Ant Colony Optimisation' paradigm (chapter 12); chapters 13 and 14 then describe the code verification and detailed parameter study of the chosen ant colony algorithm. Following this, an off-line parameter tuning of the algorithm is conducted, hybridising both particle swarm and ant colony algorithms in chapter 15.

**Part IV (chapter 16):** Comprises of a conclusion (section 16.1), contributions to the field of optimisation (section 16.2) and lastly a section on the future work to come from this thesis (section 16.3).

**Part V:** Is an appendix, comprising of additional results relating to the hybridisation of the PSO with local search (appendix A). Additionally, chapter B extends the literature review for ACO; chapter C describes the problems used in the ant colony study; chapter D includes additional results for the verification of the implemented coding of ACO; chapter E comprises of additional results relating to the parameter study conducted of the PACO algorithm; and finally, chapter F comprises of additional results relating to the comparison of the off-line tuned PACO algorithm with those available in the literature.

# Part I

# Background

# Chapter 2

# Introduction to Optimisation

It is considered here that the implications of the *no-free-lunch theorem* as stated by Wolpert and Macready [8] to be rather significant and as such, it is described as an introduction to the field of optimisation. Following this, a background to the categorisation of both algorithm and problem types are discussed.

## 2.1 The no-free-lunch theorem

An attempted understanding in the significance of the mathematical proof that is, 'the no-free-lunch theorem' by Wolpert and Macready is crucial when working in the field of optimisation. It is thought by the author of this document to signify the following: Any possible algorithmic design will not perform any better than any other (including a random search). This accounts for the fact that there are an infinite number of problem 'types' within the mathematical framework of the universe and so to design an algorithm better suited to a range of problems inevitably worsens its performance on others.

The implications of such a theorem are: Why design an algorithm which performs no better than a random search? The author of this thesis considers that the type of problems that are encountered in the real-world (where practical applications are concerned), are only a small subset of those possible. As such, algorithms can be designed to perform well over such a subset of problems.

With traditional optimisation algorithms, the limitations are clear (some of which are to be addressed), where knowledge of the problem leads to the design of efficient and fast algorithms for specific applications. These problem specific designs however inevitably confine the algorithm to effective application to a smaller range of problems (types). Indications of the no-free-lunch theorem perhaps.

However, evolutionary algorithms (EA) adapt to their environment, allowing them to perform well on problems for which they are not designed. It is for this reason that the author of this thesis considers that an adaptable algorithm does not necessarily conform to the implications of such a theorem. This is an important aspect in algorithm design considered by the author of this thesis throughout. For further information, see [8].

## 2.2 Optimisation problems and algorithm categorisation

Optimisation problems can be categorised in a number of ways, of which, the main categorisation considerations are shown in fig. 2.1. The categorisation of concern here is of discrete/non-discrete, single objective/multi-objective and constrained/unconstrained. Algorithms are then often also categorised by the types of problems for which they are designed, where a reduction of performance or an ability to tackle a problem may result when considering a problem outside its intended domain. Other categorisation types include: differentiable or non-differentiable equations; uni-modal or multi-modal (single or multiple basins within the problem search space); convex or non convex function; smooth or non smooth; dynamic (where the problem itself may change between successive iterations) or static; and linear or non-linear. With respect to non-linear functions, these can then be subdivided into function type, including quadratic, geometric or any number of other descriptive categorisation of their formulation. This is a non exhaustive list of categorisation types. Only static single objective problems are considered in this work.

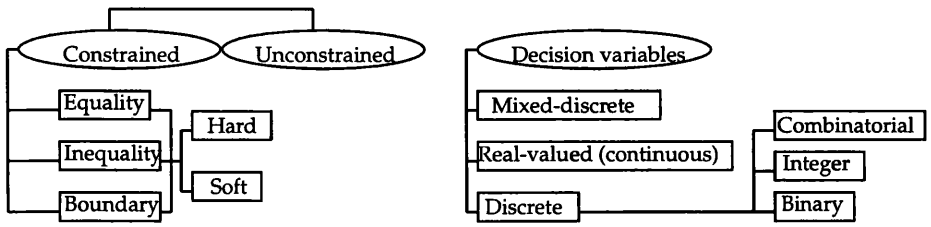



Figure 2.1: Main problem categorisation considered in this thesis.

Within the discrete domain, Combinatorial Optimisation Problems (COPs) are considered. Following the notation of [9], these are problems which consist of a finite set of objects $S$ (the search-space) and an objective function $f : S \rightarrow R+$ which assigns a cost to each object ($s \in S$).

The goal is then to find an object of minimal cost value. From here-in, the terms CO problem and discrete optimisation problem are used interchangeably.

Within the continuous domain, continuous optimisation problems are considered, where these are real-valued ($x \in \mathbb{R}$ so that a solution that can take any value $[-\infty, +\infty]$). However, the search-space may be restricted with the use of constraints. The conflict and constraint functions may also not necessarily be continuous, but there still exists an infinite number of solutions. From here-in, the terms continuous optimisation problems and non-discrete optimisation problem are also used interchangeably.

Since different problem types often result in different algorithmic approaches to achieve cutting edge performance, a number of available algorithms are discussed. This number is somewhat extensive and those described are limited to the most popular and/or have similarity with the two algorithms of interest (Ant Colony Optimisation (ACO) and Particle Swarm Optimisation (PSO)). The most popular are shown in table 2.1. In this table, problems are labelled as constructive or non-constructive, where constructive here refers to algorithms that constructively build partial solutions but also encapsulates those algorithms that deal with a complete solution to a reduced problem (i.e. subset of the original complete problem). This table also indicates whether the algorithm (in its original form) has a form of memory; is a local based algorithm[1]; its original intended problem domain; and finally whether it is population based. These forms of categorisation will be referred to in more detail in the upcoming chapter.

It should be noted that the traditional methods are generally not population-based.

| Traditional methods | | | | | |
|---|---|---|---|---|---|
| Algorithm | Constructive | Population | Local-search | Memory | Original domain |
| Exhaustive search | ☐ | ☐ | ☐ | ☐ | D |
| Local search | ☐ | ☐ | ☑ | ☐ | – |
| -Hill climbing | ☐ | ☐ | ☑ | ☐ | D |
| -Bracketing method | ☐ | ☐ | ☑ | ☐ | R |
| -Exchange algorithms | ☐ | ☐ | ☑ | ☐ | D |
| -Fixed-point methods | ☐ | ☐ | ☑ | ☐ | R |
| -Gradient-based methods | ☐ | ☐ | ☑ | ☐ | R |
| Simplex | ☐ | ☐ | ☑ | ☐ | R |
| Tabu search (TS) | ☐ | ☐ | ☑ | ☑ | D |
| Dynamic programming | ☑ | ☐ | ☐ | ☐ | D |
| Branch & bound | ☑ | ☐ | ☐ | ☐ | D |
| Greedy algorithms | ☑ | ☐ | ☐ | ☐ | D/R |
| Divide & conquer | ☑ | ☐ | ☐ | ☐ | D |
| Heuristic methods | | | | | |
| Algorithm | Constructive | Population | Local-search | Memory | Original domain |
| Simulated annealing (SA) | ☐ | ☐ | ☑ | ☐ | D |
| Ant colony optimisation | ☑ | ☑ | ☐ | ☑ | D |
| Particle swarm optimisation | ☐ | ☑ | ☐ | ☑ | R |
| Evolutionary computation (EC) | ☐ | ☑ | ☐ | ☐ | – |
| -Evolutionary strategies (ES) | ☐ | ☑ | ☐ | ☐ | R |
| -Evolutionary programming (EP) | ☐ | ☑ | ☐ | ☐ | – |
| -Genetic algorithms (GA) | ☐ | ☑ | ☐ | ☐ | D |
| -Genetic programming (GP) | ☐ | ☑ | ☐ | ☐ | – |
| -Differential evolution (DE) | ☐ | ☑ | ☐ | ☐ | R |

Table 2.1: Comparison table of popular optimisation algorithms. The column *original domain* is the broad type of problems that the algorithm was expected to tackle. $D$ refers to discrete problems and $R$ to real-valued problems.

[1]The term local search is often but not limited to gradient based optimisation techniques and consists of applying local changes to the solution until some optimal criterion is reached.

Those algorithms that are constructive and population-based are likely to be considerably improved with implementation of a local search as described by Dorigo and Stützle [5]. This is so often the case, that many authors do not consider the combining of methods to be the hybridisation of two techniques. This gives rise to such terminology as Memetic algorithms (MAs), to which different authors interpret different meanings: some define it to be the hybridisation of a local search with evolutionary algorithms. Since the categorisation of what is considered an EA is unclear at times, it seems logical to use the phrase 'Memetic approach' when considering the incorporation of a local search to any population-based heuristic approach.

Regarding algorithms for tackling combinatorial optimisation, they can in general be classified within one of two different approaches, namely 'instance-based' and 'model-based' approaches [5], the former meaning that new candidate solutions are generated based only on the current solution (or current population of solutions). These include Evolutionary computation (EC) and Simulated Annealing (SA) for example, however Tabu Search (TS) is not since it also uses information through the use of tabu lists[2]. The later (Model-based approaches) are where solutions are generated using a parametrised probabilistic model, using the previously visited solutions in such a way that the search will concentrate on the regions containing high-quality solutions [5].

ACO and PSO belong to this model-based approach which is to be discussed in further detail. The interested reader is referred to appendix B.1.4.

Yet another classification of algorithms when it comes to combinatorial problems and indeed other problem types, is how the solutions are constructed. These can be split into either exact or approximate (heuristic) algorithms. Exact algorithms are guaranteed to find an optimal solution within a certain number of steps. However, traditional approaches are generally very problem specific and prone to suboptimal convergence, resulting in the increased popularity of approximate algorithms in use today. With the use of these approximate algorithms, the guarantee of finding the global optimal solution is lost, however, a much simpler approach is usually apparent with an optimal or near optimal solution achieved with minimal computational requirements by searching a small subset of the search space.

Following the categorisation of Michalewicz and Fogel [4], approximate algorithms for combinatorial optimisation problems can be categorised by their approach to construct and or modify a solution. They generally split into three classes: tour[3] construc-

[2]A tabu list is a list of those solutions which are excluded from further consideration.

[3]A tour is a single solution made up of a number of solution components (this is a Hamiltonian circuit

tion algorithms; tour improvement algorithms; and finally composite algorithms. Tour construction methods are where a tour is constructed by adding partial solutions to the final solution until it is complete. Tour improvement algorithms are where changes are made to complete solutions by making adjustments (exchanges for example in a permutation problem). Finally, composite algorithms combine the features of the above two algorithmic subsets. Consider the Travelling Salesman Problem (TSP), a CO problem, where a salesman must visit every city and only once, then return to his initial starting city (permutation of visited cities (vertices)). A nearest-neighbour algorithm is said to be a good example of a construction heuristic for the TSP, where the nearest feasible (unvisited) city is visited at every step, starting from a random city. A possible tour improvement algorithm is the 2-opt algorithm. This algorithm exchanges 2 arcs (edges) of the tour in order to achieve a new shorter tour.

All heuristic methods in table 2.1 can be described as *metaheuristics*, meaning a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems. This is by no means the only stated definition of the term, however this meaning appears to have the greatest clarity [5].

From the upcoming description of various metaheuristics, it becomes clear that a distinction between algorithmic ideas is no longer possible, since algorithms borrow ideas and features from one another. Hybridisation is now a common trend, allowing the algorithm to effectively tackle a wider range of problems (for example the Memetic approaches described previously). For this reason, an overview of the original framework, together with recent developments since their design are given.

Since many optimisation algorithms iteratively optimise until some criterion is met, it should be highlighted here that the terms *iteration* and *time-step* can be used interchangeably and the two generally refer to a complete single optimisation sequence.

in the popular travelling salesman problem for example).

# Chapter 3

# Traditional optimisation algorithms

Traditional optimisation methods are not very robust[1] and are often but not limited to being gradient-based. These are often described as 'exact methods' that guarantee convergence to locally optimal solutions. They are generally well suited to the task of a well studied problem, but often find great difficulty for those problems that deviate from the algorithms intended problem domain. An example is a gradient based technique tackling a discontinuous problem.

Combinatorial optimisation problems are considered very difficult, in particular, for $\mathcal{NP}$-hard problems, heuristics are generally approached in their solving. $\mathcal{NP}$-hard problems are defined under the $\mathcal{NP}$-completeness theory[2][5] for the case where no known algorithm exists that will solve the problem within a polynomial time bound, and that the time to optimise can be as bad as an exponential with instance size as a worst-case scenario. Computational complexity theory is a branch of the theory of computation in theoretical computer science and mathematics that focuses on

(for further information, see [9]).

A brief summary is provided here for the purpose of explaining the categorisation of the various mainstreams of algorithms. Much of what is discussed here is based on the book by Michalewicz and Fogel [4].

Michalewicz and Fogel highlighted that these traditional algorithms are either ones that evaluate complete solutions or ones that evaluate partially constructed solutions. This also ties in with the definition of the commonly categorised approximate methods for tackling CO problems described in the previous section. Such simple classification amongst traditional algorithms and or even CO problems may similarly be applied to

[1] robustness is defined here as the ability to tackle a number of different problems

[2] a classification of problems according to their inherent difficulty

heuristic methods. With regards to the construction of partial solutions, this can be in the form of an incomplete solution to the problem posed, or alternatively, a complete solution to a reduced problem (i.e. subset of the whole sequence of cities for the TSP, or perhaps a non-linear programming problem, limited to a reduced domain of fewer search variables).

Traditional optimisation algorithms are reviewed by Michalewicz and Fogel in the context of problems of different types: The TSP (discrete combinatorial); boolean satisfiability problem (logical optimisation problem)[3]; and finally the non-linear programming problem, which is to maximise or minimise a non-linear function (real-valued continuous optimisation problem).

## 3.1 Traditional optimisation algorithms for complete solutions

Working with complete solutions is particularly effective if early termination is chosen, as a complete working solution is always available. The most influential source for this section is by Michalewicz and Fogel [4].

### 3.1.1 Exhaustive search

An exhaustive search (also called enumerative search), evaluates every solution in the search-space, until the best global solution is found and where this is not known, a search of the entire search-space is then carried out. Each solution in the search-space is generated systematically (no heuristic aspects) though ways of reducing the computational cost are highlighted in [4] such as *backtracking*[4] (somewhat beyond the scope of this review). To put into perspective of other approaches, consider a random search, even though the exhaustive search evaluates every solution, it can still perform better than other approaches on occasion, since it does not re-sample solutions which can happen in the random search.

[3] "The task is to make a compound statement of Boolean variables evaluate to TRUE"[4]

[4] Backtracking is a method by which solution components are abandoned as soon as it is determined that it cannot be completed to a valid solution

### 3.1.2 Local search for non-linear programming problems

There are numerous Local search algorithms, where they may work on various principles such as heuristics, derivatives or be strictly local based and they all generally work with complete solutions. The reasoning behind such diverse approaches in local search methods, is that there is no single approach or algorithm within this category that is superior to any other [4]. That is, they are problem specific and by no means black-box methods to non-linear programming problem optimisation.

In general, it's impossible to develop a deterministic method for finding the best global solution that would be better than exhaustive search [4].

#### 3.1.2.1 Bracketing methods:

Bracketing methods are root finding methods where the root is located in the interval $(a, b)$. Two methods are to be discussed here: bisection; and the regula falsi method [10].

The *bisection method*, is where bounds are defined and optimisation results from bisecting the range in-between bracketing points $a$ and $b$. A midpoint $m$ is found and $f(m)$ evaluated at this point (where f() is the function being optimised). Depending on the value of $f(m)$, new ranges are defined. In the case where the function tends to 0 for the optimum ($f(x*) = 0$), this is easy, since either of the ranges are reset to $m$ depending on whether $f(m)$ is positive or not. This then works on an assumption that $f(a)$ and $f(b)$ lie on opposite sides of zero (i.e. $f(a)f(b) < 0$). It is obvious that as long as $f(m) \neq 0$, then the ranges are set according to alg. 1.

**Algorithm 1** Pseudo-code - Bisection method

```
while Termination condition not met do        ▷ Test how close to zero
    if f(a)f(m) < 0 then
        b ← m
    else if f(m)f(b) < 0 then
        a ← m
    end if
end while
```

where $m = \frac{(a+b)}{2}$

This process is iterated until a chosen tolerance (error) is reached. The number of iterations required is then determined by the terminating length of interval (accuracy).

This interval length decreases geometrically as a function of the iteration number $n$, given by:

$$L_n = \frac{(b-a)}{2^n} \tag{3.1}$$

An illustration of the bisection method is shown in fig. 3.1, there the half-way point between the interval $(a, b)$ is clearly shown.

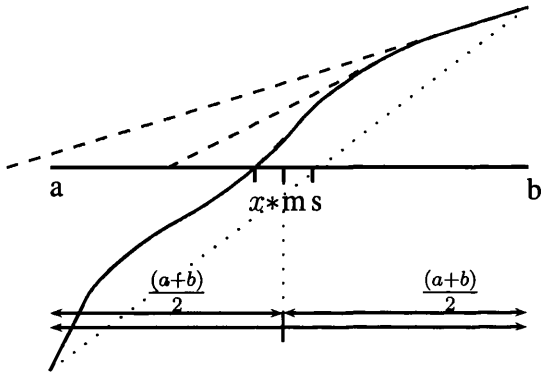



Figure 3.1: Illustration of both bracketing and fixed-point methods: green curve for the bisection method; blue curve for the Regula Falsi method; and finally the red curve indicating the Newton's method.

The *Regula Falsi method*, where a secant line[5] is constructed between the range, where the point at which this line intersects the x-axis is found. The secant line is then described by:

$$\frac{y - f(b)}{x - b} = \frac{f(a) - f(b)}{a - b} \tag{3.2}$$

so that

$$s = \frac{af(b) - bf(a)}{f(b) - f(a)} \tag{3.3}$$

where $s$ is the point of intersection.

Unlike the bisection method, the Regula Falsi method may never converge to zero as it is function dependent, though it is often much faster than the Bisection method. Further details can be found in [4, 10].

An illustration of this method is also shown in fig.3.1, showing the intersection with the axis of the line connecting points of the range.

[5]"A secant line, also simply called a secant, is a line passing through two points of a curve" taken from [11]

#### 3.1.2.2 Fixed-point methods:

These methods are described as faster than the previous methods stated (Bracketing methods), although not offering a guarantee of convergence to continuous functions. One such technique is *Newton's method.* Again, with the global optimum of $f(x) = 0$ ($f(x*)$), an initial 'guess' solution is made using some heuristic and the tangent line determined that intersects the x-axis. The intersection is then used as the next 'guess', until the distance between guesses is below a defined tolerance. The limitation of this method is clear with the requirement of a suitably good initial guess and the necessary computation of the derivative, though an approximation of this derivative may be used (then called the *secant method* [10]). This is then said to converge at a *superlinear* rate as opposed to a quadratic rate in Newton's method. The reader is referred to fig. 3.1 for an illustration of the difference between this method and the previous methods stated, where the tangent of a guess solution (Newton's method) is clearly shown.

#### 3.1.2.3 Gradient methods:

Many methods are said to utilise information about the gradient to search through the search-space. With use of the directional derivative, the search can be directed to the steepest ascent or descent (maximising or minimising respectively). However, the functions need be continuous and smooth for a gradient-based search to be possible.

These methods can be explained as follows: First, a solution is initialised randomly. A new solution at each step $i$ (iteration) is then generated by the following relation:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \Delta f(\mathbf{x}_i) \tag{3.4}$$

where $i \geq 0$, $\alpha_i$ the step size and $\Delta f(\mathbf{x}_i)$ is the gradient of $\mathbf{x}_i$. Incorporation of second-order information into this update rule then gives the following update:

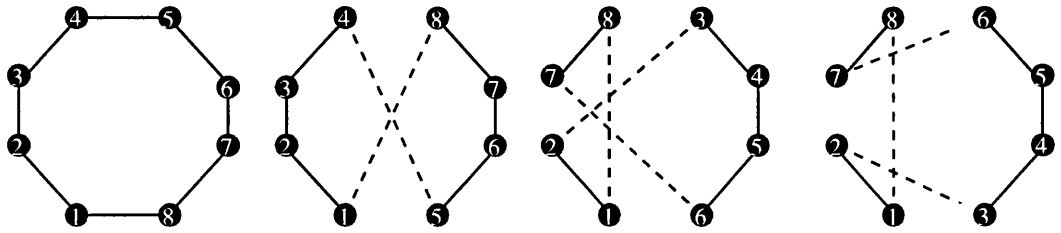
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{1}{H(f(\mathbf{x}_i))} \Delta f(\mathbf{x}_i) \tag{3.5}$$

where $H(f(\mathbf{x}_i))$ is the Hessian matrix. However, for difficult functions, the calculation of the second derivative is highly computationally expensive and so the use of *quasi-newton* methods are common, which is where an estimation of the Hessian is used instead. For a review of some common techniques, see [4].

### 3.1.3 Local search for permutation problems (Exchange algorithms)

A local search algorithm searches in the neighbourhood of a solution. Algorithms which fit into this category are deterministic, although they generally require some heuristic to initialise them, such as some greedy heuristic. Generally, exchanges are made amongst neighbours, producing new solutions until a locally optimal solution is found. These methods are particularly effective in such problems as the TSP, due to its fitness-distance correlation. The hybridisation between exchange algorithms and ACO is very common and in many cases considered necessary to reach state-of-the-art performance, as noted by countless authors [5, 12]. The most successful algorithm in dealing with the TSP (symmetric STSP and asymmetric ATSP), is the Lin-Kernighan heuristic, which is a local search exchange algorithm [13].

As described in [5], the 2-opt algorithm was formulated by G. A. Croes and the 3-opt by F. Bock and are said to be a special case of the $\lambda$-opt algorithm. At each step, $\lambda$ edges of the current tour are replaced by $\lambda$ edges, such that a shorter tour is found. As with the 2-opt and 3-opt, the time complexity of a $\lambda$-opt would then be $O(n^{\lambda})$ and since there is no bound on the number of exchanges, a 2 or 3 exchange is generally used. However, the Lin-Kernighan algorithm removes this limitation by the use of a variable $\lambda$-opt algorithm, where at each step, tests are made to determine whether $\lambda + 1$ exchanges might result in a shorter tour. An illustration of the 2-opt and 3-opt exchange is shown in fig. 3.2, where an exchange of two arcs is shown with the 2-opt and two possible combination of 3 arc exchanges shown for the 3-opt. The reader is referred to the following source by Helsgaun [14], for further information regarding the Lin-Kernighan method.



(a) original tour (b) 2-opt exchange (c) Two possible 3-opt exchanges

Figure 3.2: Popular tour exchange algorithms, where each solution component is marked by an ordering index.

A number of methods are used for speeding up such techniques: candidate lists (nearest-neighbour lists of limited length); the use of a fixed-radius nearest-neighbour

search; and the use of *don't look bits* as described in [5]. These techniques are then said to give a computation time that increases sub-quadratically with instance size. These techniques are self explanatory, other than the *don't look bits*, which are where vertices are discounted from further consideration in the exchange moves if no 'incidents' of the edge occur (no exchanges apparent using that particular vertex). These are just three standard speed-up techniques of a number available[15] (outside the scope of this review).

In [4], a 4-opt procedure is described as having been tested, but with little to no improvement observed over the 3-opt. However, an exchange only possible in the 4-opt is commonly implemented in other exchange schemes. This exchange is the combination of two 2-opt moves which themselves convert the tour into two disjoint cycles called the *double bridge* move. Another algorithm is the 2.5-opt, which is described in [5] as a heavily restricted version of the 3-opt procedure on top of a 2-opt procedure. An additional check on top of the 2-opt exchange is made to see whether inserting an edge between the one being looked at and its successor results in a better solution. This results in a small additional computational overhead compared to the 2-opt but with significantly better tours, though still falling short of the 3-opt.

## 3.2 Traditional optimisation algorithms for partial solutions

Again, following the insightful review by Michalewicz and Fogel [4], working with partial solutions is described as being particularly effective, as the structure of the problem can be used to advantage in the development of a solution. Working with partial solutions offers the possibility of decomposing the problem into sub-problems for significant speed-ups.

### 3.2.1 Greedy algorithms

With Greedy algorithms (algorithms exhibiting 'greedy' behaviour), construction of a complete solution occurs in a number of steps, where simplicity makes this family of algorithms popular. Greedy algorithms generally work by making the best possible decision at each construction step. Michalewicz and Fogel [4] describes a heuristic being used in deciding the best move at each step. A notable comment on greedy algorithms, is that in the case of the continuous programming problem, an efficient

greedy algorithm cannot be made since all points cannot be evaluated and as such, a best decision at each step is not necessarily possible. For this reason, an algorithm that exhibits greedy characteristics is used, for example, a line search with the obvious limitation of ignoring the interaction between variables. This then gives the obvious difficulty of classification of what is considered a greedy algorithm or not. In general, a greedy algorithm is those algorithms which exhibit such greedy characteristics and not an algorithm of specific design. Overall, these algorithms are said to be highly simple, but for this very reason, they generally fail to provide good solutions in real-world problems, where variables are generally strongly coupled [4, 16].

### 3.2.2 Divide and conquer

The approach in divide and conquer, is to break up the original problem into much smaller and simpler ones, solving these sub-problems in the hope of solving the original (when assembled from each of these sub-problems). This method is obviously only useful if the computational cost is less than solving the original problem [4, 16].

### 3.2.3 Dynamic programming

The idea of dynamic programming is to find a solution to the problem by operating on the current step and the next step in a recursive manner. Dynamic programming is said to be coined dynamic due to its usefulness in problems where times and orders of operations are said to be crucial [4] (which is true of CO problems). Dynamic programming works by beginning at the goal (after the final decision making stage) and working backward to the current state (decision). Again, the problem specific application of this method is apparent. Like Divide and conquer, Dynamic programming works by dividing the problem into subproblems where unlike in Divide and conquer, the subproblems may overlap and subproblems are no longer solved only once.

### 3.2.4 Branch and bound

The Branch and bound algorithm deals with the elimination of areas of the search-space which are unlikely to constitute part of a good solution, which is particularly effective in problems of high complexity. This method is said to be based on an exhaustive search and works by defining a lower bound on the cost of a function (or upper bound in the case of a maximisation function). By defining this lower (upper) bound, a solution that

has larger cost than this, need not be computed [4, 16]. In a tree-like representation, this is equivalent to the pruning of branches, leaving a significantly reduced pruned tree. A heuristic is used to define this lower bound (for example a nearest neighbour heuristic for the TSP) and the more accurate this lower bound, the faster the algorithm will converge, since more solutions are eliminated from the search. The disadvantage of this method, is the necessary trade-off required between the computation of these bounds and the time saved in the pruning of these solutions.

One highly successful and state-of-the-art algorithm for many applications, is the hybrid beam search with ACO by Blum [12], a classical approximate tree search method, said to be an incomplete derivative of the branch and bound algorithm.

### 3.2.5 Summary

Most real-world engineering problems exhibit a vast number of locally optimal solutions (multimodality and often multiobjective) and differ in characteristics, such that traditional methods require problem specific implementation to solve. In some cases, convergence to the global minimum can be guaranteed, but normally, such a guarantee cannot be made as most real-world problems are restrictive. This leads to the algorithmic development of heuristics and metaheuristics. To further illustrate the problems faced with dealing with different types: the standard traditional gradient based methods require the function being optimised to be continuous and differentiable, where if not, the method fails.

Though traditional methods are generally less computationally expensive than the metaheuristic, the metaheuristic intends to solve a vastly wider range of problem types and does so by only searching a small subset of the 'suggestible' search-space. When considering the computational cost of some example combinatorial problems, the necessity of searching this subset becomes obvious: If all states were to be evaluated in a boolean satisfiability problem (exhaustive search), the number of states would be $2^n$; for the TSP there are $(n-1)!/2$ solutions; and finally, for non-linear programming problems there are no traditional methods which give satisfactory results to this search-space of infinite states.

# Chapter 4

# Trajectory methods

Again, the main source of inspiration for the details found within these sections are from a book by Michalewicz and Fogel [4], where the reader is recommended to turn to this for a more detailed review of various methods.

## 4.1 Simulated Annealing (SA)

SA is one of the oldest metaheuristics and is inspired by the analogy to the physical annealing of solids (crystals), designed for tackling combinatorial optimisation problems [2]. Michalewicz and Fogel [4] described it as having been first implemented by Kirkpatrick S. and colleagues in 1983, inspired by the works of N. Metropolis who developed a Monte Carlo method for calculating the properties of substances and defined the 'Metropolis' procedure. SA is an algorithm for global optimisation and is a local search method, where at each step in the process, a solution is randomly generated. If this solution is improved over the previous solution, then it is accepted with a probability that has analogy with temperature[1]. This 'temperature' decreases over time and so the probability of accepting a worsening solution also decreases over time [2]. SA is a local search algorithm, because like TS, it searches amongst its neighbourhood. However, differently to TS, these neighbours have random order, where the move made is probabilistically chosen.

The temperature reduction is slow (cooling schedule), since in a real substance a fast cooling would result in an amorphous solid (where the resulting structure lacks long-range order) and so does not reach its minimum energy state (i.e. crystalline form).

[1] As a substance cools, the mobility of the molecules reduces, where a tendency is apparent for the molecules to align in a crystalline structure.

This has the analogy to being trapped in a local minimum.

Through this method (acceptance of worsening solutions), local optima maybe escaped (through uphill moves). The probability to which the solution is accepted or not is often defined by the 'Metropolis distribution', which simulates the behaviour of a system of particles by considering the difference between energies associated with their states $l$ and $n$, given as:

$$p = exp\left(-\frac{E_l - E_n}{kT}\right) > 1 \tag{4.1}$$

where $p$ is the probability that state $l$ is accepted if $p \leq 1$. $T$ is the absolute temperature, $k$ the Boltzmann constant. A general pseudo-code for this algorithm is shown in alg. 2.

**Algorithm 2** Pseudo-code- Simulated Annealing (SA)

```
s ← Generate initial solution
s_best ← s                          ▷ Initialise parameters, s_best =best solution
n ← 0                               ▷ Initialise parameters, n=iteration counter
while Termination condition2 not met do   ▷ Maximum iterations not exceeded
    while Termination condition1 not met do   ▷ Solution not good enough
        s' ← Generate solution in the neighbourhood of solution s
        if p(f(s), f(s'), temp) > random() then
            s' ← s
            f(s) ← f(s')
        end if
        if f(s) < f(s_best) then s_best ← s
        end if
    end while
    n ← n + 1
end while
return s_best
```

SA has three main functions: Generation function; acceptance function; and the decrement function. Most SA algorithms differ from one another with respect to the generation and decrement functions. A guaranteed convergence version of the algorithm requires a considerably slow annealing schedule and theoretically requires an infinite number of states as described by Dorigo and Stützle [5], Weise [17] (asymptotic convergence to the global optimum is observed).

Representation of solutions is generally made as real-valued vectors, though integer vector representations are also used. Solutions are then generated by introducing small random changes to these solutions. A summary of how a solution is generated can be

found in [6].

Developments since the original algorithm as noted by Weise [17] include its implementation on multi-objective optimisation problems but most significantly, of the neighbourhood generation mechanism, acceptance probabilities and the cooling schedule (whether static or dynamic).

Multi-objective problems are tackled in much the same way as with TS or the hill climbing algorithm[17] which is to be later described. To do this, it borrows features from EAs, considering a number of solutions rather than an individual solution. A selection scheme is then normally applied to determine which individuals should be used as parents to generate the next offspring.

The annealing schedule in SA can be generated by a Genetic Algorithm (GA) for example. The basic form commonly uses an exponential or linear cooling, so escaping highly attractive suboptimal regions may be difficult. Blum and Roli [18] describes that the most successful of SA algorithms utilise a non-monotonic cooling procedure, where some oscillating procedure with a cooling then re-heating phase is advantageous. This results in a balance between diversification and intensification.

The standard SA algorithm is memory-less[18], however, Blum and Roli describes that implementations with memory have been designed and shown to be beneficial.

With regards to typical applications tackled by SA, those listed by Weise [17] include combinatorial optimisation (scheduling, routing, assignment etc.), function optimisation, image processing, economics and finance, circuit design, machine learning, networking and communication etc. One of the first applications to which SA tackled, was the TSP. Further information can be found in [6, 17–19]. SA is also described as being used in general as a component in metaheuristics rather than a stand-alone search algorithm.

## 4.2 Tabu Search (TS)

Michalewicz and Fogel [4] describes TS as having been designed by F. Glover in the mid 80s with the intended application of combinatorial optimisation problems and is considered a generalisation of iterative improvement algorithms like SA as described by Pirim et al. [19]. It differs from SA, in that TS is based on a type of neighbourhood search utilising a memory management system while SA can be considered to be a biased random search. TS can apply 'uphill' moves only when it becomes trapped in local minima whereas SA can probabilistically perform them at any point in the search.

A pseudo-code is shown in alg. 3 (taken from [5]).

TS is originally deterministic and chooses its solutions as such, unless confronted with two equally good neighbour solutions. It is another local search method which extends the hill climbing or steepest decent method with a candidate list. The hill climbing technique is a local search method for discrete spaces that evaluates all its neighbours, where the best of these solutions is chosen to compete with the current solution. The definition of this neighbourhood is where the main underlying variation between various implementations remains. Returning specifically to TS, candidate lists of neighbourhood solutions are common methods to increase the speed of the algorithm, allowing the search to choose from a subset of the available neighbourhood (typically a subset of prominent moves). This memory management allows it to avoid cycling (returning to previously visited solutions), but since this list length (*tabu tenure*) is finite, higher period cycles are still possible.

At each step, a neighbour solution is chosen that is the best (even if worse than the current solution). If a basic principle tabu list is used, then the system may become frozen as the list grows ever larger with each iteration. This is due to the increasing possibility of no feasible solution being available in the neighbourhood of the current solution. In addition, the computation cost and memory requirements would also increase. To avoid this, common developments include modifying the definition of the tabu memory. With fixed length, when it is full, the newly created candidate replaces the first one. Other methods described by Weise [17], are in reducing the size of this list through the use of clustering measures, where a distance measure is used and a perimeter around the listed solution candidates declared to be tabu. A small tabu list promotes the search toward a more concentrated area of the search-space, since the memory of visited solutions is quickly forgotten, resulting in the finding of better solutions amongst the nearby-by neighbours. On the other side, with a large tabu list, the memory of solutions in the near-by search-space will be remembered as being visited for a longer period and it is only once the search is distant enough from this area that this previously visited area of the search-space becomes not tabu once again, an obvious trade-off is apparent here.

One approach is to make use of a dynamic tabu list, as described in [17], a variation to the standard tabu search, perhaps based on some online approach, taking into account the progression of the search but also the cycling rate of the search.

One of the most significant developments since the simple TS is the implementation of different types of memory: short-term and long term memory. With regards to the

**Algorithm 3** Pseudo-code- Tabu search (TS)

```
s ← Generate initial solution
s_best ← s                                          ▷ Initialise parameters
while Termination condition not met do
    A ← Generate admissible solutions    ▷ Determine subset of the neighbourhood
solutions which are not tabu or are tabu but satisfy the aspiration criterion
    s ← Select the best admissible solution
    if f(s) < f(s_best) then
        s_best ← s                          ▷ Memory structures updated (tabu list etc.)
    end if
end while
return s_best
```

information stored by the short-term memory tabu, Blum and Roli [18] describes that attributes are stored rather than complete solutions, including perhaps partial solutions or differences between solutions. Some strategies are described as storing some complete solutions within memory (normally elite solutions). With the use of attributes to determine the tabu status of solutions, it is quite possible that more than one solution with not be accepted (since an attribute does not describe solutions in full). To overcome this, an 'aspiration' criterion is used, which enables a solution to be used even if forbidden by the tabu conditions. This aspiration criterion is normally defined as those solutions which are better than the current best one. This then defines the tabu search as a dynamic neighbourhood search technique [18].

A long term memory can also be used, which takes into account information since the beginning of the run. Long-term memory as discussed by Blum and Roli [18], may make use of such information as recency, frequency, quality and influence (four dimensional memory in TS). Regarding frequency memory, this may be the number of times a solution or attribute is found within a region of the search-space for example. Choices of the search that led to good solutions can also be recorded, allowing a better and more efficient search towards the global optimum. Finally, the recording of information to identify good attributes (high quality solutions) can be made.

Developments other than those listed above, are path re-linking and strategic boundary searching, dynamic parameter adjustment and probabilistic TS. Path re-linking is where two or more solutions are combined in the hope of finding another that is of better quality. Further details regarding these techniques are discussed by Gendreau and Potvin [20].

Typical applications of this technique include CO, machine learning and networking

and communication, operations research and biochemistry etc. as described by Weise [17]. Of the COPs, TS is said to perform particularly well in Job Shop Scheduling (JSS), Vehicle Routing, Quadratic Assignment and the MAXSAT problem[18]. TS is one of the most successful metaheuristics and most commonly cited metaheuristics[18], but it is not so efficient at escaping local minima, though it can indeed do so.

To compare with SA, memory management in the original TS is intended to avoid cycling behaviour but does give the possibility of escaping local minima. However, SA has an advantage, as very poor solutions than the current solution can be accepted based on the 'temperature' of the system, making it highly suitable to statistically escape local minima. The interested reader is referred a book by Pirim et al. [19] for further information.

## 4.3 Summary

The trajectory methods described in this chapter offer much in the way of improvement over traditional methods. They can generally be adapted to a great number of problems and specifically perform very well on CO problems (scheduling etc.). However, it is noticed that the number of parameters has increased from the traditional methods, with the inclusion of memory and temperature. These require further understanding by the user of the algorithm. All these algorithms up until now process only one solution at a time in their original form, which is an inherent limitation, where the use or interaction between solutions may lead to a better understanding of the search-space. This better understanding could also lead the search toward an eventual better quality solution and a reduced computational cost. This introduces the necessity of population-based methods, which have a set of competing and or cooperating solutions.

It is said that many ideas in TS are commonly in use by other metaheuristics, especially since many metaheuristics incorporate memory into their search and so benefit from a sophisticated memory management scheme which is TS.

# Chapter 5

# Population methods

## 5.1 Evolutionary Computing (EC)

This section is most heavily influenced by the following sources [2, 6, 7, 21, 22], where the interested reader is directed towards these sources for further reading. EC is a machine learning technique inspired by natural evolution, which has its origins in the ideas of Charles Darwin with his works in 1859 "The Origin of Species By Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life". The main ideas from Darwin are summarised by Affenzeller et al. [23], as:

- *Evolution, change in lineages, occurs and occurred over time.*
- *All creatures have common descent.*
- *Natural selection determines changes in nature.*
- *Gradual change, i.e., nature changes somehow successively.*
- *Speciation, i.e., Darwin claimed that the process of natural selection results in populations diverging enough to become separate species.*

The origins of EC as described in [23], began in the sixties in two separate locations (United States and Germany). Two different approaches were derived, GA and Evolutionary Strategies (ES) respectively. GA was developed by an American computer scientist and psychologist J.H. Holland, for the purpose of understanding self-adaption in biological processes (making it much closer to the biological model than ES, as described in [23]). ES on the other hand was developed by Rechenberg and Schwefel for

the purpose of optimisation. The major difference between these two approaches is the representation of the genotype and also the way in which the operators are used. In GAs, the mutation operator is said to have its main use in avoiding stagnation, though mutation is said to be the primary operator in ES.

EC splits into three main categorises as shown in fig. 5.1, where the many subsets of EAs arguably fit under one of these three categories, though only the most common metaheuristics are described within this review. It should be noted that Genetic Programming (GP) is considered by some to be a category in itself, however, it is described here as a branch of GAs.

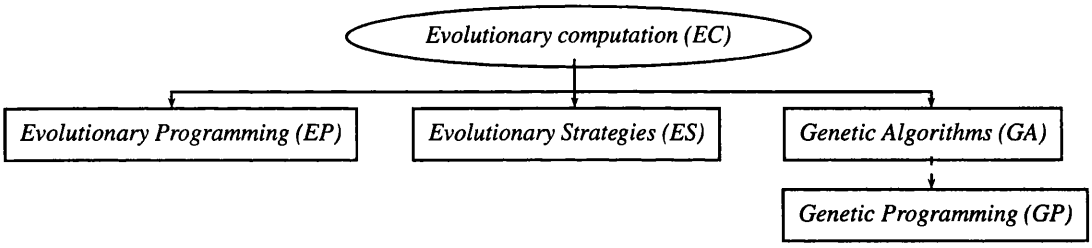



Figure 5.1: Categorisation within Evolutionary computation

All algorithms under this category are inspired by population genetics and typically consist of selection, mutation (modification) and recombination (crossover) operators. That is, survival of the fittest in order to refine a set of solution candidates iteratively. EAs are generally either geontypic-based (GAs) or phenotypic-based (EP and ES), where a genotypic-based approach is considered a bottom-up approach, where the systems behaviour emerges. The later is a top-down approach, where the focus is on observable features. Engelbrecht [6] describes the difference as being, that genotype describes the genetic composition of individuals (inherited from its parent(s)) and that the phenotype is the behavioural traits of the individual within its environment.

The general form of EAs is shown in alg. 4, which is influenced by a book by Dréo et al. [21].

*Individuals* are those which are subject to evolution and represent a solution to the given problem being tackled. The grouping of these individuals is where the population is defined, where this population evolves each iteration with the use of various operators. This results in a new population (called a *population generation*) to be created. As described by Dréo et al. [21], a *parent* is one or more individuals used by an operator, with those that result from the application of such an operator are its *offspring*. Where

**Algorithm 4** Pseudo-code- Evolutionary Computing (EC)

```
Initialise n-dimensional population Gen(t=0)
while Termination condition(s) not met do
    Reproduction: selection of parents among a population of
        μ to generate λ offspring

    Variation operators used on the λ selected members,
        generating λ offspring                    ▷ Cross-over and/or mutation

    Fitness evaluation of the offspring

    Select μ individuals of the λ offspring and μ
        parents (or only among the λ offspring) to generate a new generation

    Iterate t=t+1
end while
```

more than one operator is used successively, the offspring generated from one, becomes the parent for the next operator.

### 5.1.1 Selection Operators

Selection for reproduction is called here *selection*, for consistency with [21]. This determines how many times an individual is to be reproduced in a generation. The second type of selection operator is called *replacement* selection, which decides which individuals are removed from the population so that the population size remains fixed from generation to generation.

The fitter an individual, the more often it should be selected to reproduce or survive. Since an overall bias toward the fitter individuals is necessary from the application of these operators, a fitness of these individuals must be defined which depends on some objective function. That is, the fitness of each of the offspring must be evaluated in each generation. This fitness function is described by [21] as both difficult to define and also often computationally expensive.

The various methods in which selection can occur are now outlined.

**Selection pressure:** *Selection pressure* as described in [21], is where the variation operators are not used and so better individuals should be able to reproduce faster than those of lesser fitness for evolution to occur, where its offspring end up dominating the

population over time (replacing the population). The higher the selection pressure, the greater the risk of premature convergence.

**Genetic drift:** *Genetic drift* is the random variation caused through random sampling.

**Proportional selection:** *Proportional selection* is where individuals are selected with probability proportional to their fitness. This method has the effect of allowing a particular solution to dominate (stagnate the search) when the differences between candidate solutions is small (which is likely later on in the search).

**Roulette wheel selection:** Here, fitness values are normalised and each divided by the maximum value, so that each different candidate solution can be thought of as a slice in the wheel corresponding to the selection probability of that particular individual. Again, the probability of a solution being selected is very similar to those who have similar fitness and so the algorithm may not tend toward the best solution.

**Rank based selection:** To avoid the limitations of the above, a *rank based selection* method can be used, where the ordering of the fitness values determines the probability of selection and not the fitness itself. In this way, a rank-based roulette wheel selection method may be used by each individual, ranked from worst to best and divided by the summation of all the positions, giving equal slices in the wheel to each of the solutions.

**Tournament selection:** This technique is where every solution is randomly paired, enabling the possibility for rather weak solutions to survive to future generations, which is highly beneficial in problems where many local minima are present. Furthermore, the best individual will not dominate the reproduction process.

**Truncation selection:** This method chooses the $n$ best members of the population, with its use in either reproduction or replacement selection. For reproduction, $\lambda$ offspring are generated from the $n$ selected parents and each of these will have $\frac{\lambda}{n}$ offspring. If used for the replacement selection, then a population of $\mu$ members are generated for the next generation and then $n = \mu$.

**Elitism method (replacement selection):** This method selects a set of individuals to survive to the next generation, where the number selected which survive without being mutated are called the generation gap.

### 5.1.2 Variation Operators

Variation operators which may also be called *alteration* operators[7], have the purpose of finding new solutions that are not in the current population. The number of possible operators are said to be numerous, but they generally fit in the category of either *mutation* or *crossover.*

Mutation is where an individual is modified to generate a new individual (add new genetic information to the genetic pool that does not necessarily exist in the current set of solutions). Mutation achieves exploration through increased diversification and mimics the changes seen in nature (a number of small changes resulting in significant differences). The *mutation-rate* determines the proportion of individuals which undergo this mutation (i.e. the probability of mutation).

Crossover operators are where parts of two or more individuals (parents) are combined to produce offspring, based on the concept of sexual reproduction. The way in which these variation operators are used is highly problem-type specific (dependent on the representation of these solutions). Dréo et al. [21] gives the example of tackling an $n$-dimensional continuous search-space, where an $n$-dimensional vector $\mathcal{R}^n$ is generally chosen to represent a solution. With regards to an example discrete-space application (specifically the TSP), an individual commonly represents a single tour (an integer vector) where the variation operators should only be allowed to build legal tours (unvisited). The *crossover-rate* determines the number of members of the population which are crossed among the offspring (probability of crossover). Since crossover is commonly known to achieve rather poor solutions at times, it is common to introduce some preference to crossing members that are close to one another (by distance in the search-space). This is then defined by the *restriction radius.*

In GAs, the mutation-rate is generally low and the crossover-rate rather high. In ESs however, there is no cross-over and a 100% mutation-rate is apparent. Mutation is important in generating a solution transformed near the solution it has transformed from, allowing each individual to perform a random local search [21]. On the other hand, the crossover is said to loose its importance once members are in proximity to the same valley (peak) due to the reproduction selection (with an individual being possibly selected more than once), since they undergo no crossover.

### 5.1.3 EC and the travelling salesman problem

There are four pre-requisites as defined by Fogel [22] for solving a problem with EC, these are:

- Solution representation
- Devising a random variation operator (used to generate offspring, mutation and recombination)
- Rule for solution survival
- Initialisation of the population

The chosen problem is the TSP due to its simplicity. With regards to representation, the simplest way in to represent the problem is to identify each different possible permutation [22]. This is the case for ACO for example, where the problem is represented by an $n \times n$ matrix. The cost function is then defined as a means to evaluate any candidate solution, which in the simplest case would be the tour length (as for ACO). With regards to the random variation operator(s) used to generate offspring, two possibilities are highlighted in [22], including sexual and asexual reproduction. The former is where two parents exchange information, then recombined to generate offspring. The later is essentially cloning, where mutations are introduced in the genetic information of the offspring. An illustration of these two methods are shown in table 5.1, where any number of parents may be used in limitless ways of recombination. In this illustration, a tour is combined from the two parents at a random point in the sequence, where each solution here is a sequence of numbers representing a single tour in the TSP.

### 5.1.4 Representation of solutions

#### 5.1.4.1 Binary representation:

Binary representations originated with GAs, where any discrete alphabet may be used, i.e. binary, integer, discretised real numbers etc. It is described in [21] that the genotype is generally represented by this discrete alphabet while the phenotype, which is the solution to the problem, generally represents it directly. The genotype is said to undergo the "action of the genetic operators" such as selections and variations, while the phenotype is said to be used only for the fitness evaluation. The example is then given of a solution which is expressed naturally as a vector of real numbers, where this would be

| Asexual reproduction: | | Offspring |
|---|---|---|
| Parent#1 | | |
| [123456] | → | [126453] |

| Sexual reproduction: | | Offspring |
|---|---|---|
| Parent#1 | | |
| [123\|456] | | |
| | → | [123623] |
| Parent#2 | | |
| [145\|623] | | |

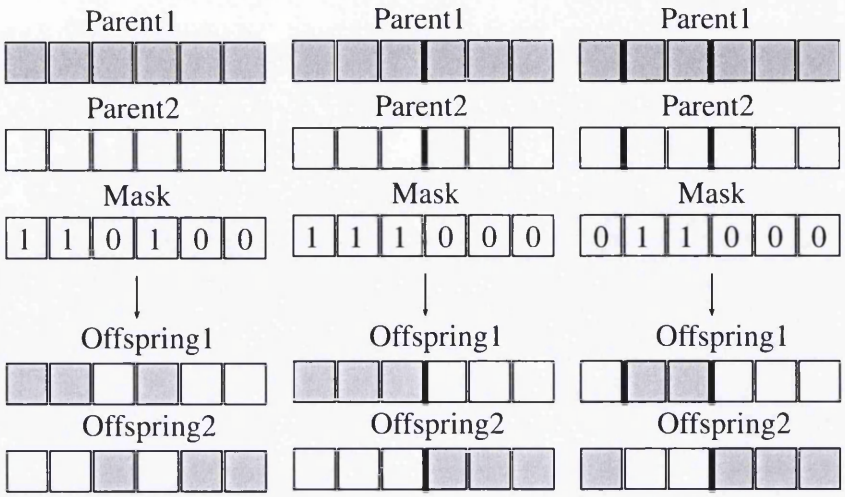
Table 5.1: Illustration of possible forms of reproduction within EAs; one parent (inversion) or two (or more) parents.

the phenotype. The genotype will then be the binary string which codes this vector (i.e. the representation of this solution but not the solution itself).

With binary representation, the most common methods for crossover include the *one-point crossover* or the more generalised *n-point cross-over* and *uniform crossover*, as identified in [2, 21] and illustrated in fig. 5.2. The former is where a point in the parents string is chosen at random and bits are then exchanged (the reader is referred to fig. 5.2). The n-point corr-over is an extension of this to $n$ points at which to cross-over. Uniform crossover considers a probability of swapping bits at each bit state of the string. An illustration of three different cross-over methods for the binary string representation is shown in fig. 5.2. This then results in two offspring from two parents, where one offspring of the two are chosen by random, if only one offspring is to be used.

Following the crossover of parents, a mutation is made by randomly flipping bits in each of the strings. Since the representation of one number to another may require a number of bit changes (called Hamming cliffs), a mutation can result in a significant jump in the phenotypic space, where a small jump is made in the genotypic space. This is especially important in the case of a bit string which represents a vector of integer or real numbers. To avoid this, a better representation is said to be when the hamming distance is one, where adjacent values differ by only one string bit (adjacency property). One such scheme is called 'binary-reflected Gray coding'. However, Dréo et al. [21] describes these Hamming cliffs as not too influential on the overall performance of the algorithm.

Adaptation of the algorithm in order to deal with continuous search-spaces from a binary string representation point of view is possible (GAs). If for example a variable




(a) Uniform cross-over (b) One-point cross-over (c) Two-point cross-over

Figure 5.2: Binary string cross-over operators in genetic algorithms.

in the range of 0.1 to 0.5 were to be chosen, then this can be represented as a bit-string according to some required decimal precision. This can be done as discussed by Kennedy et al. [2], Engelbrecht [6] by the following: If $s$ is the number of bits assigned to the objective variables, then the number of integers in the sub-string that can be generated range from 0 to $(2^s - 1)$. Using standard binary decoding, the conversion between a continuous variable encoded to a fixed length bit string is represented by the following:

$$(2^s - 1)\frac{z - z_{min}}{z_{max} - z_{min}} \tag{5.1}$$

where $z \in [z_{min} z_{max}]$, which is a real in these limits. The string length is determined by the chosen decimal precision ($prec$) and the dynamic range by the product of the two, so that:

$$\text{string length} = (z_{max} - z_{min}) * prec \tag{5.2}$$

In this way, a continuous space is represented in its discrete form, though this is obviously not ideal with the additional computational expense dependent on the required precision.

#### 5.1.4.2 Real representation:

With the real representation generally used in ESs, the search is conducted by a population of vectors in a bounded $n$-dimensional real valued search domain. The individuals of the population as described in [21] are then placed in the search-space according to a probability distribution, where this distribution has an expectation and a variance. By analogy, the expectation can be considered the centroid of the population and the variance, to be the moment of inertia of the population [21].

Regarding crossover, choosing two individuals to generate offspring occurs in a number of possible ways. One is like the binary representation with the exchange of components. Other methods include specific ways in which to cross parents including uniformly generating offspring within a hyper-rectangle, where the two parents define its longest diagonals (see [21]). The third option highlighted in [21] is the linear BLX-$\alpha$ crossover, which is where offspring are generated at random on a line segment passing through the two parents. For mutation, common methods include the uniform mutation (the simplest), where a random variable of a uniform distribution in a hyper-cube is added to an individual. However, the more popular method is a Gaussian mutation, where the value added to an individual is from a Gaussian random variable. Further details can be found in [21].

#### 5.1.4.3 Other representations:

Representation of individuals for permutation problems is also possible with ordinal, path or sequence representation. While EP was originally designed with an ordered list representation for evolving a Finite State Machine, other representations are also possible with real-valued representations being the most common for optimisation of continuous functions. Another application is with the training of a neural networks.

Individuals in GP are executable programs which are said to be represented as trees [2, 6] (also called parse or syntax trees). GP then evolves computer programs within a problem domain and measures the performance of these programs. Within this tree structure, a 'grammar' has to be defined, including a terminal set and function set. The former specifies the variables and constants, while the later specifies the functions applied to the element of the terminal set. An example problem is given here, in order to explain the above definitions, guided by those explained by Engelbrecht [6]:

$$y = cos(x) + c * sin(z)/ln(a) - 6. \tag{5.3}$$

In this example, the terminal set is { x,c,z,a,6. }, where $x, c, z, a \in \mathbb{R}$. The function set is $\{cos, +, sin, /, -\}$. The goal is to discover the true program within the space of potential computer programs (the program which gives the correct output for a given set of inputs). Once the terminal and function set are defined, a way in which to specify the fitness of individual programs is required which is problem-dependent. Fitness is normally inversely proportional to the error produced by the individuals (program) output. This can be done by taking a number of sample cases, to obtain an average performance measure. Each function within the function set requires a fixed number of arguments, which are called the functions arity, so that one of the specification requirements of the terminal set is in choosing a minimal set that will complete the task correctly.

For example, to create a boolean expression, a function set comprising $AND$, $OR$ and $NOT$ logic is sufficient to generate any boolean expression and so this is the minimal set. An example of an $XOR$ boolean expression from [6] is shown in fig. 5.3, where the function set consists of $\{AND, OR, NOT\}$ and the terminal set is $\{x1, x2\}$ where $x1, x2 \in \{0, 1\}$.

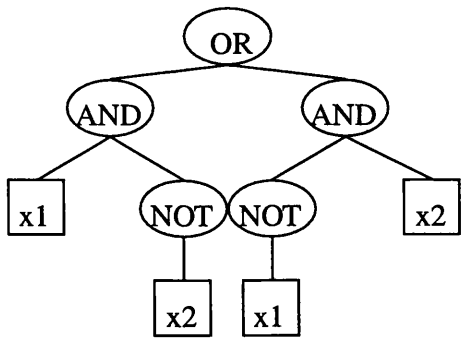



Figure 5.3: Syntax tree for an $XOR$ boolean expression

Variations on set-up include a fixed tree or varying tree structure, where the later is most common with restrictions on the maximum depth of a tree normally imposed (increasing maximum depth as a function of generation number is discussed in [6]). With regards to the nomenclature of the tree-based representations, the size of the tree generally refers to the depth of the tree, while the shape refers to the branching factor of nodes in the tree. Initialisation occurs randomly within the maximum depth restrictions. For each individual, a root is selected from the set of function elements. Each branch from the root and each non-terminal node are then determined by the arity of the selected function. For each of these non-root nodes, selection is made at random from the terminal or function set, where if one is chosen from the terminal set, the corresponding

node becomes a leaf (end of the branch, where no expansion occurs). Another method of initialisation as highlighted in [6] is to initialise to be as as simple as possible, and to grow them (increase their complexity if necessary during the evolutionary process). A more detailed explanation of the available initialisation methods are described in [2].

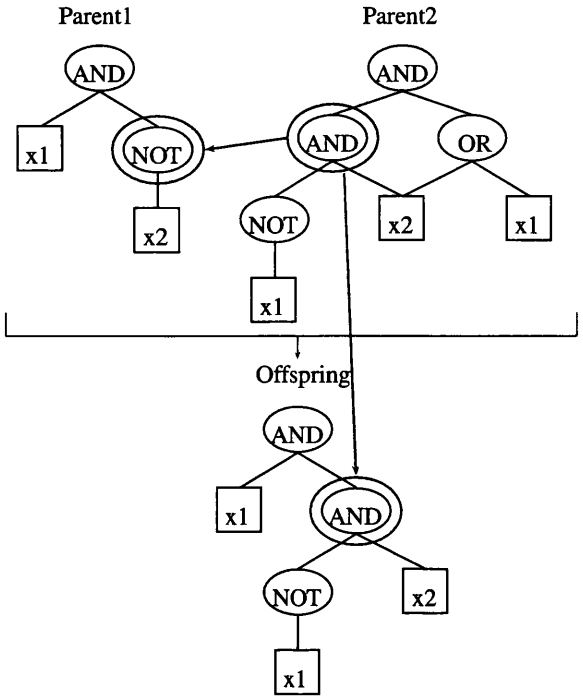
Crossover operations may occur by the generation of one or two offspring. In the former, a random node is selected within each parent, then cross-over occurs by replacing the sub-tree of the one parent with that of the other. The later is where random nodes are again chosen from the two parents, only this time the sub-trees are swapped to result in two offspring. An illustration of this is shown in fig. 5.4.

Regarding mutation, those summarised in [6] are function node mutation, terminal node mutation, swap mutation, grow mutation, Gaussian mutation and trunc mutation. Swap mutation is to randomly select a node, to which those connected to it are swapped. Grow mutation is where a node is randomly selected and replaced by a randomly generated tree restricted by a particular depth. Gaussian mutation, to quote Engelbrecht [6], is where a terminal node which represents a constant is selected at random and mutated using a Gaussian random value to its value. Trunc mutation is where a function is randomly selected and replaced by a random terminal node. Some asexual operators are also devised (the reader is referred to [6]).

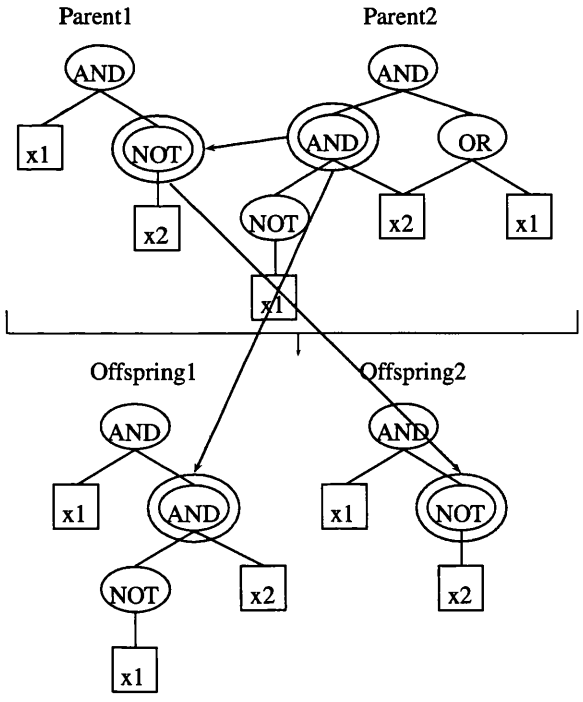
Lastly, the crossover and reproduction in GP is said to work in parallel with the fitness being determined. A probability is said to be assigned to the two, where the sum is 1 (that is, to perform one or the other). Reproduction is said to be done in a similar way to GA (roulette wheel) where over-selection is a method utilised if populations are said to be large (+1000). For this case, highly fit solutions are given a higher probability of selection.

## 5.2 Swarm Intelligence

Swarm intelligence is a branch of artificial intelligence that involves the study of the collective behaviour that emerges from decentralized and self-organized systems. The two most popular of methods are outlined here: particle swarm optimisation; and ant colony optimisation.




(a) One offspring cross-over




(b) Two offspring cross-over

Figure 5.4: GP example crossover operations.

### 5.2.1 Particle Swarm Optimisation (PSO)

Since the PSO is to be described in greater detail in Part II, only a summary of information regarding the paradigm is provided here. PSO was originally based on the analogy of bird flock simulations and consists of a swarm of particles, randomly initialised within feasible space with also randomly initialised velocities. The velocity of each of the n-dimensional particles are accelerated towards its own personal best position and the best of the whole swarm with stochastic weighting between the former and the later. PSO, like ACO, fits into the field of swarm intelligence and as discussed in the previous section, is considered to belong to the category of EAs by many authors. To re-iterate, it is not considered here to belong to this category, as it is not inspired by natural evolution. Other techniques may also fit into this category such as the artificial bee-hive colony by Karaboga [24] or more recently the Cuckoo search by Xin-She Yang and Suash Deb as discussed in [25]. However, the main emphasis here remains with PSO and ACO, since they are well developed over the last couple of decades.

PSO was originally inspired by bird flocks by Kennedy and Eberhart [26], with its original purpose for optimisation of unconstrained continuous search-spaces. Like most algorithms developed, it has been modified/hybridised to tackle a grater range of problems such as constrained problems, multi-objective optimisation, dynamic search-spaces and even CO problems. Since the constraint handling techniques of EC is highly developed, PSO often implements these ideas in an effective manner. Since EAs are also population based, the exchange of ideas is somewhat easily made. A not so recent review of these developments can be found in a paper by Banks et al. [27, 28].

Firstly, to put into perspective with ACO, EAs are generally competitive algorithms, while ACO and PSO have a cooperative strategy. Both algorithms are also said to share their difficulty in theoretical development or understanding, since the behaviour is emergent from interactions amongst the group. The algorithms differ with agents indirectly communicating in ACO by the use of pheromones, which act as a memory with stochastic element. In PSO, individuals follow either their own personal best or the best of the entire swarm or local neighbourhood with a stochastic element (with these locations stored in memory).

The most significant element to be discussed is in the implementation of CO problems with PSO, since this algorithm was originally designed to tackle unconstrained continuous search-spaces. These methods are outlined here as discussed by Banks et al. [28]. The main method outlined for tackling CO problems, is in defining a discrete PSO through the use of fuzzy matrices for the positions, where the values in the matrix define

the degree of membership to the corresponding element of the CO problem. The velocity equation is then modified to use matrix representations of velocities and positions. A solution can then be found to a discrete problem through a continuous search-space. Another method described, is in the introduction of some concepts of GAs, together with some local search hybrid or perhaps to hybridise with ACO or SA, with both being highly suitable for discrete problems.

Typical applications are generally to optimise functions with continuous-valued parameters and in the training of neural networks, however a number of other application are possible from clustering, design, scheduling (CO) to data mining as discussed by Engelbrecht [6]. Through the incorporation of ideas from EAs, it is possible to deal with constraints, discrete problems, dynamic problems and multi-objective problems for example.

The reader is referred to [3, 7] for an extensive investigation in the behaviour and development of the PSO algorithm together with constraint handling techniques.

### 5.2.2 Ant Colony Optimisation (ACO)

Since ACO is to be described in greater detail in Part III, only a summary of information regarding the paradigm is provided here. ACO is based on the analogy of the foraging behaviour of real ant colonies. ACO first initialises its population of ants randomly, then constructs solutions by incrementally and probabilistically adding solution components with the use of both pheromone information (historically good solution components) and heuristic information, until a complete solution is made. After all ants of the population have constructed their solutions independently, they deposit pheromone on the arcs of solution components corresponding to their solutions, proportional to the performance of their solution (original ACO). From this, following ants (ants of the next cycle) build their solutions with additional attraction toward solution components belonging to good solutions found by previous ants (due to their increased pheromone concentration). That way, information is exchanged indirectly through interaction with the environment called *stigmergy*. This way, a pheromone map of the solution space is constructed and well performing components are favoured over low performing components.

ACO is discussed briefly here with respect to its range of applications, as again this method is reviewed in some detail in chapter 7. The original ACO was originally intended for tackling discrete optimisation problems (for the most part, $\mathcal{NP}$-hard CO

problems), with recent trends to be found in the following sources [5, 9, 12, 29]. Like GAs, ACO has tackled all major CO problems from Routing, assignment, scheduling to subset problems. However, more recent trends are with parallel implementations and continuous optimisation implementations as described by Mullen et al. [29] and Blum [9, 12]. Other recent avenues of research include multi-objective optimisation, dynamic $\mathcal{NP}$-hard problems and even stochastic optimisation problems. The most recent overview of such developments along these lines can be found in [30]. Blum [9] goes into some detail into the way in which such types are tackled, however, since this deviates somewhat from the intended use of the algorithm, it is suffice to say, that like PSO and EAs etc. variants can be found which tackle a multitude of problem types. As with other population-based methods such as EAs and PSO, ACO shares the overwhelming advantage of performance increase with its hybridisation with other techniques. In ACO, this is particularly the case, where state-of-the-art performance is generally only reached with the hybridisation with other techniques (generally some form of local search technique) [9, 12].

## 5.3 Ant algorithms, other algorithms inspired by the behaviour of real ants

For completeness, other algorithms that fit under the banner of 'ant algorithms' are briefly discussed here with the main source of information taken from [2, 5] by Kennedy et al., Dorigo and Stützle. In addition, a review article by Dorigo et al. [31] also describes in some detail those algorithms which belong to the label of 'ant algorithms'.

Algorithms defined as belonging to this category are multi-agent systems which are inspired by the observation of real ant colonies (specifically its behaviour, with the exploitation of indirect communication called stigmergy). Stigmergy, originally defined by Grassél, used it to describe the indirect communication through modification of the environment in *Bellicositermes Natalensis* and *Cubitermes* species of termites and this is the generally accepted meaning of the term. The exact definition used by him is as follows:

Stigmergy: *Stimulation of workers*[1] *by the performance they have achieved.*

ACO is based on the foraging behaviour of ants, but other algorithms have been designed based on other aspects of the ant colony, such as those based on brood sorting,

[1] workers are one of the casts in termite colonies [31]

division of labour and cooperative transport. In these algorithms, Kennedy et al. [2] discusses that stigmergy not only has the effect of changing the way in which solutions are built (as is the case with ACO) but also directly affects the solution itself.

### 5.3.1 Inspiration from division of labour

Algorithms inspired by the division of labour of ants are generally highly suited to task allocation problems. In real ant colonies and indeed other social insect species, a division amongst reproductive and worker casts exists. Among these, there can then be a division of sub-casts who are assigned specific tasks. It should be noted that cooperation arises for two reasons. One is of a genetic reason, where differences between individuals make some suitable for some tasks while others not so suitable for that particular task (polymorphic ants[2] for example). The other reason is through self-organisation. That is, the complex collective behaviour that emerges from the interactions among individuals that exhibit simple behaviour (such as major worker[3] ants switching roles by performing minor worker[4] ant tasks). This might occur for example if not enough worker ants are available. In real ant colonies, triggers may range from reproductive success rates, food availability, predation, climatic conditions, phase of colony development, colony size and structure of the colony etc. as discussed by Kennedy et al. [2]. This aspect of division of labour is called plasticity. Specialisation allows a greater efficiency of individuals, since they know the task or are better equipped to deal with it.

With regards to the above, Kennedy et al. describes a study by E. Wilson on an ant species called *Pheidole genus*, which is subdivided into two morphological sub-castes: majors (soldiers) specialised for seed milling, abdominal food storage and defence; and minors for performing everyday tasks. Wilson is said to have noticed that majors began to perform tasks usually performed by minors after alteration of the minor/major ratio (i.e. replacing the missing minors). To this purpose, a response threshold model was said to have been developed by Bonabeau [2, 31]. This model is described as follows: $s$ is the intensity of the task specific stimulus (demand) and $\theta$, the response threshold (in units of stimulus intensity) which determines the tendency of an individual to perform a certain task according to $s$. One possible function for the probability by which an individual performs a task according to its stimulus and defined thresholds is given by:

[2]polymorphic ants are the worker and reproductive casts which have roles given to them according to their physical differences making one more optimised to follow a particular task

[3]worker ants who have large mandibles for cutting large pray and protecting the nest [2]

[4]worker ants who usually feed the brood or clean the nest [2]

$$T_\theta(s) = \frac{s^n}{s^n + \theta^n} \tag{5.4}$$

where $n$ is the 'steepness' of the threshold. For $s \ll \theta$, the probability by which an individual performs the task is low and where $s \gg \theta$, the probability is high. The stimulus intensity is then described by the following relation in the case of one task and $n = 2$,

$$s(t+1) = s(t) + \delta - \alpha n_{act} \tag{5.5}$$

where $\delta$ is the increase in stimulus (constant), $n_{act}$ the number of active individuals and $\alpha$ is the scale factor, which is the amount of decrease in intensity due to the activity of an individual (see [31]).

With the simulation of this threshold model, similar results to experimental observations of Wilson were shown with its application made easy to multiple tasks (see [2]). Kennedy et al. [2] describes methods with fixed thresholds to achieve emergent task succession (the reader is referred to [2]).

One of these models was successfully used by Krieger and Billeter, as discussed in [31] to organise a group of robots toward the task of puck-foraging, where this model is said to be successful in such a simple environment [32].

From the above discussion, the thresholds were fixed, which has many limitations. One of the most obvious, is that it does not account for the creation of task allocation thresholds, since individuals are 'pre-differentiated' and their roles pre-assigned, as discussed in [31]. With the modelling of real ants, only over short enough time-scales can the thresholds be considered constant. Dorigo et al. [31] then describes Theraulaz et al. as overcoming these limitations with the development of varying thresholds through a reinforcement process, where later, Bonabeau et al. successfully tackled the adaptive mail retrieval problem with it. Any task allocation problem is said to be suitable for its application.

### 5.3.2 Cemetery organisation and brood sorting

Another aspect of the ant algorithm, is its ability to cluster and sort, naturally resulting in researchers to follow this line of research for data analysis and graph partitioning.

With regards to cemetery organisation, the aggregation of dead bodies by workers is said to have been observed in a number of ant species[31], with the formation of clusters from initially randomly distributed items. An illustration of this is shown in fig. 5.5.

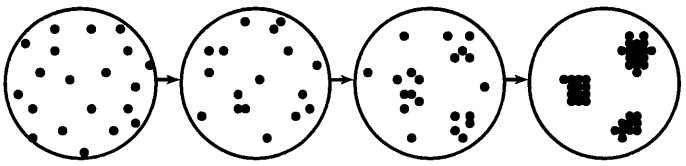


Figure 5.5: Tendency of clustering to occur in ant cemetery formation.

In this case, the stigmergic variable is the distribution of the clusters. That is, small clusters of items attract other ant workers, who in turn deposit their items, making the cluster grow and become ever more attractive to successive ants. This is then a positive feedback mechanism, resulting in the formation of large clusters. Again, Deneubourg et al. is said to have proposed a biologically inspired model based on plausible assumptions called the *Basic Model*. This model is based on the idea than isolated items are picked up and placed at a location where more of those items are located. In the case where only one type of item is in the environment, the following probability distribution for the picking up of an item is used:

$$p_p = \left(\frac{k_1}{k_1 + f}\right)^2 \tag{5.6}$$

where $f$ is the fraction of items in the neighbourhood of the ant (amount of clustering) and $k_1$ is the threshold constant. The threshold constant then determines whether the item is likely to be picked up ($f \ll k_1$) or not ($f \gg k_1$). Similarly, the probability of the ant depositing the currently held item is given by:

$$p_d = \left(\frac{f}{k_2 + f}\right)^2 \tag{5.7}$$

where $f$ is then said to be the crucially defined function, where Dorigo et al. [31] describes Deneubourg as having calculated $f$ according to a short-term memory by the number of recorded items found. This was not biologically inspired and was designed for the case of robotic implementation. With only one object type, $f = N/T$, where $T$ is the number of time-units in the short-term memory and $n$ the number of items found. This method is then said to be easily adapted to multiple item types.

Laumer and Faieta are said to have generalised the model by Deneubourg for the problem of data analysis, by defining a dissimilarity between objects in the space of object attributes (see [31]).

Lumer and Faieta randomly initialised four Gaussian distributed clusters in the at-

tribute space and found that the end spacial distribution of objects did not correspond to the number of clusters in the attribute space. For this reason a number of features were added. Ants (agents) moving at different speeds were incorporated, with fast ants said to form clusters over large scales while slow moving ants picking up and dropping with higher accuracy on the smaller scale. Secondly, a short-term memory was incorporated, where each ant is able to remember $m$ number of items dropped and their location, so that a comparison is made with its memory (this allowed the ant to move in a direction similar to previous deposits). Thirdly, a so-called behavioural switch is incorporated, where ants can 'destroy' clusters if no action has been performed by the ant over a number of time-steps.

The positions of the clusters are however arbitrary on the plane and so there is not a perfect correlation between the attribute space and the spacial distribution of items, though this is said to be unimportant in such applications as *textual databases* and so it is nonparametric. It is then said to belong to the partitioning approach since objects move from cluster to cluster. With regards to applications, the Lumer and Faieta algorithm is described by Dorigo et al. [31] to have been extended by Kuntz et al. to a wide range of graph drawing and graph partitioning problems.

### 5.3.3 Inspiration from the foraging and path parking of ants

Algorithms inspired by the foraging behaviour and or path marking of ants, other than ACO, include Edge Ant Walk and Vertex Ant Walk by Wagner et al. [33]. The artificial ants like in ACO, are said to deposit pheromone on visited arcs (or nodes), building their solutions according to the pheromone already deposited on these arcs. However, this implementation is said to be very different from ACO since the stigmergic variable (pheromone) is used to direct ants towards unexplored areas of the search-space.

Vertex Ant Walk is described as follows. Ants are initialised on random vertices, where ants visit arcs according to the number of marks deposited on the edge, together with the time at which the most recent mark was left. It then behaves as a steepest-descent algorithm, as it chooses a vertex to move to with the lowest number of marks on it (with the analogy to odour). Upon visiting an edge, this vertex is updated by incrementing the mark counter for that particular edge. Regarding the exit of a vertex (moving to a neighbouring vertex), this may cause conflict with other ants. To overcome this, Wagner et al. describes the use of a hard-coded I.D. on the ants or a difference of phase on the ants clocks or even a random phase. This is then designed to stop phase

collisions between ants. This method is said to use an optional 'dynamically altering cost function' to avoid local minima. In the edge ant walk, edges are marked rather than vertices, though the vertex ant walk was found to be more efficient, since the later can cover the graph in a shorter number of steps. Applications suggested by Wagner et al., are robotics and internet search, since this algorithm is a graph search algorithm or more precisely a graph covering algorithm. With regards to robotics, such applications as floor cleaning, wall painting or de-mining mine-fields are discussed possible avenues.

### 5.3.4 Inspiration from cooperative transport

Algorithms in this area tend to have applications primarily within robotics, though research is somewhat less extensive compared to other characteristics of the ant colony. These include the cooperation of agents for pushing or pulling objects, where a single agent alone cannot perform on its own. Again, a recruitment process is used which may use a chemical marking or direct contact. See [5] for further details.

## 5.4 Summary

EC is a well developed field, where typical applications include data mining, combinatorial optimisation, fault diagnosis, classification, clustering, scheduling, and time series approximation for example. With regards to their description, each of the individual algorithms under the banner of EC are no longer distinct since common trends are to borrow ideas from one another. With regards to representation of solutions, this also is not distinct between the various algorithms since they have drifted away from their original representations. Furthermore, this hybridisation of ideas between algorithms takes a further step with the hybridisation of algorithms themselves. In particular, with local search algorithms, then called Memetic algorithms (as discussed previously).

With regards to EAs similarity with ACO and PSO, Mullen et al. [29] describes their similarity of being population based and that all three incrementally build better solutions by building on previous solutions. EC however, in general contains only information on the current population (i.e. no form of memory), though forms of EC that incorporate direct memory do exist. Also, both PSO and ACO evolve their populations, though they are not inspired by the process of natural evolution. For this reason they are not considered here truly EAs.

As with EAs, common trends within swarm intelligence techniques are apparent,

with increasingly popular hybrids, indicating that no one technique can dominate any other, perhaps indicating the statements made by the no-free-lunch theorem (see section 2.1). This indicates that no particular algorithm can be claimed to be most suitable for any one particular problem type. All algorithms share ideas and mechanisms and the distinctions are only clear when comparing the original versions. That being said, PSO is predominantly still better known for its application to the optimisation of real-valued functions, while ACO leading in performance in particular COPs.

To conclude on the above discussion ant algorithms, these are defined as multi-agent systems inspired by real ant colonies, exploiting some stigmergic behaviour. Dorigo et al. discussed that this stigmergic variable can take various forms (as already highlighted). These stigmergic variables come in various forms depending on the behaviour that the algorithm is trying to mimic. Algorithms inspired by the foraging and path marking generally use an artificial pheromone of sorts as a stigmergic variable, where models inspired by brood sorting generally use the physical distribution of items to drive communication. However, all these methods are either not directly suitable for COPs, or require a more difficult approach when compared to ACO as discussed by Dorigo and Stützle [5]. These algorithms are all said to have in common, the property of flexibility and robustness and are decentralised systems. Another property common to all these algorithms is that optimisation occurs due to the self-organisation of the algorithm (i.e. the solutions result from an emergent behaviour to optimise rather than a specified one).

# Chapter 6

# Other paradigms with connection to ACO

## 6.1 Artificial Neural Networks (ANN)

> *...although there are conceptual similarities between neurons in living brains and logic gates in computers, the firing rates of biological neurons are much slower than computer logic gates (on the order of milliseconds for neurons versus nanoseconds for computers).*[4]

This quote is particularly insightful, since the approach in recent computer development and of algorithmic development to solve optimisation problems is to take advantage of the parallel implementations, where the power of the human brain is derived.

Artificial Neural Networks (ANN) or shortened to just Neural Networks (NN) are inspired by the interconnected parallel structure and functional aspects of the human brain and so it is biologically inspired. They are essentially algorithms which map a non-linear function from an $I$ dimensional input to a $K$ dimensional output ($f_{NN}$ : $f^I \rightarrow f^K$) and are machine learning techniques as described by Engelbrecht [6]. $f_{NN}$ is described as being usually a complex set of non-linear functions (one for each neuron in the network), where a neuron (perceptron) forms the building blocks of the NN. There are two types of learning: supervised learning; and unsupervised learning approach. The former is where training data is required (given a set of input-output pairs, the error that the network makes can be calculated). The later is where the network is to self organise into some configuration as described in Michalewicz and Fogel [4].

### 6.1.1 Theory regarding the workings of ANNs

#### 6.1.1.1 The Perceptron

The artificial neuron functions by receiving a vector of $n$ input signals, $\mathbf{x} = (x_1, x_2, ..., x_n)$ from the environment or other neurons. Each input signal $x_i$ has an associated weight $w_i^k$ corresponding to neuron $k$ to strengthen or deplete the input signal (i.e. the neurons are connected to one another by 'synapses' which amplify or diminish the signal being forwarded). The neuron then determines the net input signal and uses an activation function to compute the output signal with this given input signal. The output signal is then influenced by the threshold (bias) value $w_0^k$. An illustration diagram of the simplest single artificial neuron based on the biological equivalent called the perceptron is shown in fig. 6.1.

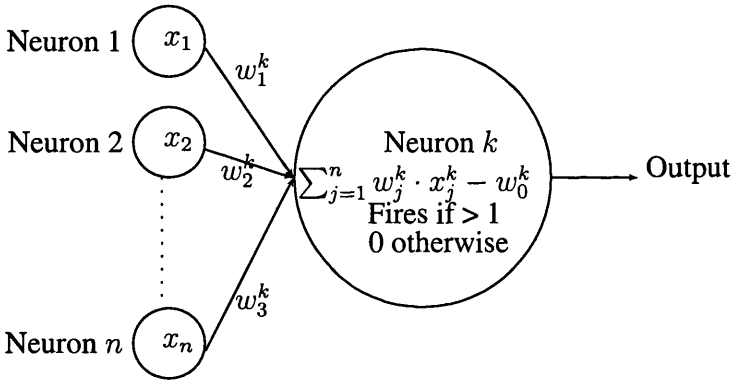



Figure 6.1: Single perceptron shown, where the net input signal is shown to be the weighted sum ($w_0^k$ is the threshold of output neuron $k$).

An activation function as described in [6] determines the output signal based on the input signal and bias (threshold).

Considering the functioning of two neurons connected to a third (simplest case), the neuron will fire when $w_1x_1 + w_2x_2 + w_0 \geq 0$ (the output of the neuron is 1 when fired and 0 when it is not). This is the equation for a line and so the output neuron fires when the input neurons fall on one side of the line, which is defined by $w_1$, $w_2$ and $w_0$ as shown in fig. 6.2.

This is then a pattern recognition system called a linear discriminant function or step threshold function (i.e. for classification). Determining these weights can be achieved in a number of ways. One method is to make use of training sets with a given input matrix and corresponding output patterns. The weights are adjusted to achieve the

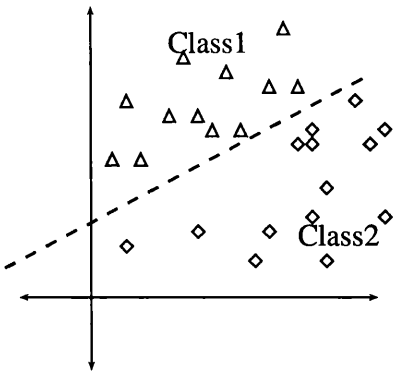



Figure 6.2: Two-dimensional feature-space for a perceptron with linear threshold function.

correct output of this training set. Weights can be adjusted by a constant step size for example, each time the corresponding output does not match the required class.

It is apparent, that where data from the classes are not linearly separable, this method fails to converge to the solution and the use of more neurons is required (using multiple linear boundaries etc.).

#### 6.1.1.2 The linear artificial neuron

The simplest case of function approximation could be achieved by the linear artificial neuron as described by Innocente [7], where rather than a linear threshold function, a transfer function is used. This is explained by the following illustration shown in fig. 6.3.

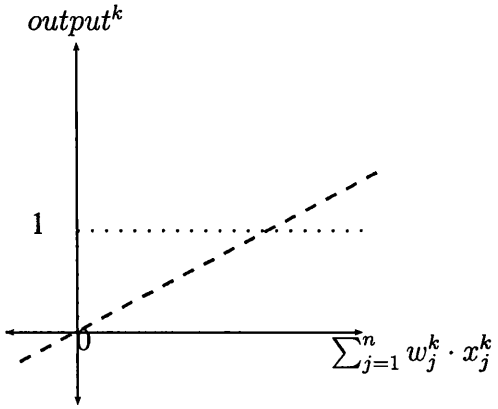



Figure 6.3: Illustration of a threshold (dotted) and transfer function (dashed) for a perception.

The linear artificial neuron then results in an output that corresponds to the linear

combination of its inputs, making it suitable for function approximation rather than for classification (perceptron). The weights are adjusted by determining the difference between the neurons $k$'th output ($\beta_k^p$) for pattern $p$ and the desired output $t_k^p$.

Each time a training set is provided, the weights are then adjusted and the 'mistake' (error) corrected in a single step, where a neuron given a new pattern $(p+1)$, forgets the previous learning. An alternative as described by Innocente [7], is the $\delta$-rule (a neutron with this correction rule is called ADALINE), which updates with controlled learning speed. With this method, the mistake (error) is not corrected immediately but instead iteratively improved with newer input patterns. However, this is said to only converge to a local minima.

#### 6.1.1.3 The nonlinear artificial neuron

Since both linear threshold and transfer functions are somewhat limited, often, nonlinear artificial neurons are utilised for classification of non-linearly separable functions. In terms of approximation, the linear artificial neurons are said to be limited to approximating hyper-planes, thus, again the use of nonlinear artificial neurons is clearly required. A popular non-linear transfer function as described in [4, 7] is the sigmoid function. Again, the weights can be adjusted according to minimising the sum squared error of the model (NN) with the training sets or through some gradient-descent strategy [4, 7]. The limitations of using a single artificial neuron are apparent, with the simple example of XOR boolean function representation (see [6, 7]). More than one neuron is then necessary to solve this problem.

#### 6.1.1.4 The multi-layer perceptron

A multilayer perceptron (that is a NN of multiple perceptrons with threshold functions) is a *feed forward* network, since information flows in only one direction from the input nodes to hidden nodes and finally to the output node(s). Other transfer functions include Gaussian functions or sigmoid functions, though many others are possible.

For function approximation, the number of neurons required in the hidden layer tends to infinity for arbitrary functions, though most require only a few. For this reason, most implementations for various applications use only a single hidden layer, though Michalewicz and Fogel describes that the number of required nodes to be reduced if the number of layers were to increase. This topology (network structure) is said to be rather underdeveloped [4]. With regards to the number of neurons in a network, too few result

in the bad approximation of the function, though with increasing neuron numbers, the tendency to overfit becomes apparent (analogy to over-fitting polynomials to data of too higher order). With the use of the gradient-based training of weights (such as back propagation), it is then possible in difficult problems to become trapped in a locally optimal set of weights which do not correspond to the desired level of error. This will then be indicated with the implementation of a new training set(s) (i.e. optimal on the training sets used so far, but a large error is apparent on any additional training sets used on average). A method in which to overcome this inherent problem is to utilise different methods for training the weights, including the use of SA, EA or PSO for example.

#### 6.1.1.5 Applications

Other than pattern classification and function approximation, further possibilities are open to NNs with the use of recurrent or extended networks, where either the output(s) can be fed back into the inputs (meaning that the output is a function of the input and also the previous output) or where feedback loops are used between layers of neurons, resulting in a form of memory. This extends the applications of ANNs to prediction (standard recurrent networks) and applications involving associative memories (associates new input patters to ones already seen and stored- Hopfield networks) and pattern discovery (data mining, through the clustering subsets of available data according to a set of rules).

CO problems are also possible with NNs as highlighted by Michalewicz and Fogel, where the TSP is tackled from an unsupervised NN (no training data). However, Michalewicz and Fogel describes that NN methods are generally not very competitive compared to other heuristics for this purpose.

### 6.1.2 Closing remarks

To summarise, the classes of applications as described in [6] primarily consist of classification and function approximation, though other types include pattern matching, pattern completion, optimisation, control, times series modelling and data mining (pattern discovery) etc.

Mullen et al. [29] describes ANN as having some similarity with ACO, since ANNs like ACO are biologically inspired. Similarity is also met between these algorithms with the common graphical representation by set(s) of connecting nodes. It is described in [29] that states in ACO are analogous to neurons in NNs and the local neighbourhood

around this state to be analogous to the synaptic links exiting the neuron. Ants are then described as being the input signals that propagate through the NN. The pheromone trails (reinforcement of solutions) then reinforce the synapse used.

For further details regarding the specifics of the ANN paradigm, the reader is referred to [4, 6, 7].

# Part II

# Particle Swarm Optimisation

# Chapter 7

# Background to particle swarm optimisation

The in-house particle swarm optimiser at the department of Engineering at Swansea university is an advanced global optimiser. One possibility and natural progression to improve both its effectiveness and performance is for its hybridisation with other successful techniques. To this end, the author of this thesis seeks to hybridise this global search with a local search sequential quadratic programming (SQP) algorithm.

Combining the PSO algorithm with SQP[1], by providing the latter with 'good' initial solutions, enables the feature of guaranteed local optima convergence which the PSO alone does not have. The SQP is contained within the optimisation toolbox of Matlab (fmincon) and is used as a black box method.

This chapter consists of a detailed background to Particle Swarm Optimisation (PSO), with respects to its origins and common formalisation section 7.1, major developments and trends apparent in the literature section 7.2 and finally the local search implementations available in the literature section 7.3.

## 7.1 Origins

In the last two decades, research into the field of swarm intelligence has resulted in clear benefits in its use for the purpose of optimisation. swarm intelligence fits into the category of modern heuristics, as defined previously as an algorithm that intends to find a solution to a problem within a suitable computational time without guarantee of

[1] SQP is an iterative nonlinear optimisation method, which can handle constraints

optimality (see section 2.2). Modern heuristics have a clear advantage over traditional methods, in that they are not problem specific.

The original formalisation of the PSO paradigm and its place amongst others was first described by Kennedy and Eberhart [34]. PSO fits into the above categories and can also be considered to have roots in evolutionary algorithms and more specifically genetic algorithms, with its stochastic processes giving it similarity to the former. The ability to follow its personal best and it local neighbourhood best[2] then gives it similarity to the crossover operator in the later as described by Kennedy and Eberhart.

The most notable of simulations to bird flocks and inspiration for PSO as described by Kennedy and Eberhart [26] are by Reynolds [35] and Heppner and Grenander [36]. Reynolds created a well-known bird flock simulation using agents called "*boids*", independently modelling agents trying to stick together (attraction) while avoiding collisions with one another and other objects within the environment. This simulation resulted in a 'flock-like motion' and as such, was deemed a success. The principle idea of the model, was that intuitively, a collision avoidance is necessary but also that sticking together makes sense for a number of reasons, from protection from predators to finding food through larger search patterns. Three basic behaviours are said to have driven this model [35];

1. Collision Avoidance: avoid collisions with nearby flockmates
2. Velocity Matching: attempt to match velocity with nearby flockmates
3. Flock Centering: attempt to stay close to nearby flockmates

The work by Heppner and Grenander was also highly inspirational to the creation/ development of the original PSO with it being quoted by Kennedy and Eberhart [26]. Heppner and Grenander similarly modelled bird flocks as Reynolds, however, they introduced a 'dynamic force' into the simulation, where the birds are attracted to a 'roost'. Using this method, no 'craziness' operator was required for the desired behaviour. This craziness operator was used to mimic the random movement of birds from the flock and is somewhat similar to the mutation operator in GAs, having the effect of increased diversification. Where the nearest neighbour velocity matching was removed, the behaviour was slightly changed, but a 'swarm' was now observed rather than a 'flock', as described by Millonas [37]. Kennedy and Eberhart [34] then defined PSO as belonging

[2]The neighbourhood best is the individual with the best quality solution amongst a predefined group of other individuals.

to the field of swarm intelligence according to the five quoted principles of Millonas [37]:

1. The population should be able to carry out simple space and time computations.
2. The population should be able to respond to quality factors in the environment.
3. The population should not commit its activities along excessively narrow channels.
4. The population should not change its mode of behaviour every time the environment changes.
5. The population must be able to change behaviour mode when it's worth the computational price.

PSO also takes its inspiration from the statement by sociobiologist Wilson O. that the sharing of information can be an evolutionary advantage, outweighing competitiveness [26].

The paradigm known as PSO first came about by the works of Kennedy and Eberhart [34]. PSO is based on the intelligence that emerges from the social interactions amongst individuals of the swarm. Kennedy and Eberhart also defined its place amongst other paradigms including the previously discussed EAs.

A general description of the original PSO by Kennedy and Eberhart [26, 34] is as a randomly initialised swarm within feasible space[3] with randomly initialised velocities. The velocity of each of the n-dimensional particles is accelerated towards its own personal best position (pbest) and the best of the whole swarm up until the current timestep (gbest) with stochastic weighting between the former and the later (this was called the GBEST model).
Each particle 'i' in dimension 'j' has its velocity updated according to:

$$\begin{aligned} v_{ij}{}^{(t)} = v_{ij}{}^{(t-1)} &+ iw \cdot U_{(0,1)} \cdot \left(pbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right) \\ &+ sw \cdot U_{(0,1)} \cdot \left(gbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right) \end{aligned} \tag{7.1}$$

The solution coordinate of the i'th particles of the j'th dimension is then updated according to eq. (7.2):

[3]The feasible area of the search space is where all constraints are satisfied (i.e. $\leq 0$)

$$x_{ij}^{(t)} = x_{ij}^{(t-1)} + v_{ij}^{(t)} \tag{7.2}$$

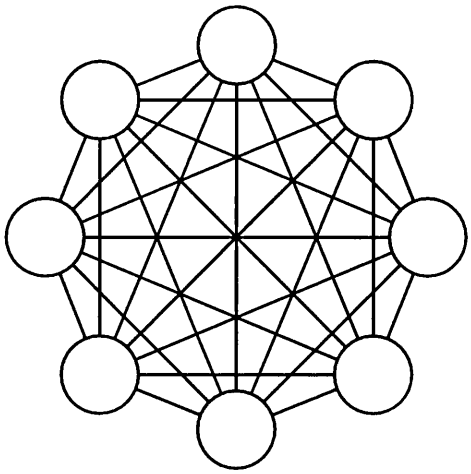
Where $U_{(0,1)}$ is a random number from a uniform random distribution in the interval [0,1], re-sampled for each time it is called; $iw$ and $sw$ are the individuality and social weights respectively.

In its original form, the social weights were both set to 2.0 (constants) "so that agents would "overfly" the target about half the time" [26] and velocities are often clamped so as to "prevent overflow". This overshooting produced better results with improved swarm dynamics.

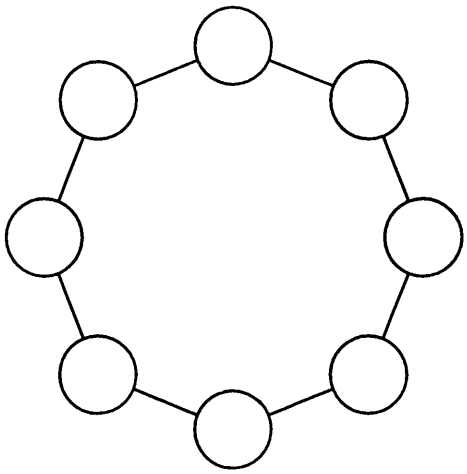
Kennedy and Eberhart also introduces the investigation of the local model (LBEST) as well as the global model (GBEST) with the manipulation of the 'neighbourhood' (where the neighbourhood is defined here as the group of particles within the swarm for which each individual is able to communicate with). eq. 7.1 is then represented by eq. 7.3, with lbest being the neighbourhood best solution. The two neighbourhood types described (local in its simplest form) are shown in fig. 7.1a and fig. 7.1b respectively. fig. 7.1a shows the global topology, where each and every particle is able to communicate their solution with one another. fig. 7.1b then shows the case where only two other particles are able to communicate with each particle, having the effect of delaying clustering by slowing the rate by which information is exchanged through the entire swarm.

$$\begin{aligned} {v_{ij}}^{(t)} = {v_{ij}}^{(t-1)} &+ iw \cdot U_{(0,1)} \cdot \left(pbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right) \\ &+ sw \cdot U_{(0,1)} \cdot \left(lbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right) \end{aligned} \tag{7.3}$$

Three neighbourhoods were tested by Kennedy and Eberhart: the GBEST model (where information about the best conflict found for each particle was available to all particle); a local model (LBEST) model with 2 neighbours; and another LBEST model with 6 neighbours. It was concluded by Kennedy and Eberhart that the GBEST model performed the best in terms of speed (time-steps to convergence). However, the LBEST models were much less prone to local optima convergence. From this early development and introduction of the PSO paradigm, Kennedy et al. succeeded in creating a robust and simple global optimisation algorithm. The pseudo-code for this algorithm is shown below;



(a) Particle swarm neighbourhood: global topology.



(b) Particle swarm neighbourhood: local ring topology.

Figure 7.1: The two most common neighbourhood topologies used for the PSO in the literature.

**Algorithm 5** Pseudo-code - PSO original

```
for each time step t do
    for each particle i do
        Update x̄_i^(t) & v̄_i^(t)          ▷ using eq.(7.1) or eq.(7.3) and eq. (7.2)
        Calculate conflict of particle_i^(t)
        Update pbest_i & gbest/lbest      ▷ gbest or lbest depending on neighborhood
    end for
end for
```

## 7.2 Development since the original algorithm

A number of important developments in the PSO paradigm have been made over the last decade and a half, many of which have been discussed by Bratton and Kennedy [38]. The most significant advancements to the basic algorithm are highlighted here, where the reader is referred to [38] for the source of much of this discussion.

Important advances in the definition of the Neighbourhood have been made with the original, defined according to euclidean distances, like the biological models in bird flocks, where birds communicate only with their immediate neighbours (defined by their distance from one another) as described by Reynolds [35] and Heppner and Grenander [36]. In all but early experimentation however, a pre-indexing of particles determining the neighbourhood of each is made, regardless of any distance measure,

since no observable advantage of using a distance measure was demonstrated.

Such features as the clamping of the velocity ($v_{max}$) are also discussed from the original formulation, with an explosion of the particles occurring [7]. This article also brings together definitions of the common terminology used within the literature such as local topology, commonly misinterpreted amongst the literature as meaning any neighbourhood other than the global topology. It is also stated that the use of swarms using global topology is incomplete at best [38]. This is likely due to its likelihood for premature convergence with the fastest possible exchange of information.

Particularly important advances in the formulation of the PSO paradigm are highlighted in [38], including the introduction of the '*Inertia Weight*' and that of the '*Constriction factor*', where both were introduced to remove the need to clamp the velocities (stop the apparent explosion of the swarm). Detailed analysis of this behaviour can be found in the thesis by Innocente [3, 16]. The Inertia Weight $w$, as defined by Clerc and Kennedy [39], was designed to adjust the influence of the previous particles velocities and thus eliminate the need for $v_{max}$ to be used. The *Inertia Weight* allows the control of the swarm to be more or less constricted. This resulted in the reformulation of eq. 7.3 to eq. 7.4;

$$\begin{aligned} {v_{ij}}^{(t)} = w \cdot {v_{ij}}^{(t-1)} &+ iw \cdot U_{(0,1)} \cdot \left(pbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right) \\ &+ sw \cdot U_{(0,1)} \cdot \left(lbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right) \end{aligned} \tag{7.4}$$

This inertia weight is suggested in [38] to be a dynamic parameter, where a larger value could encourage early exploration and a reduction, encouraging the swarm to converge, being analogous to friction. Generally this is set as a linearly changing value, but numerous possible options are available.

The constriction factor $\chi$ was defined with a similar intention of removing the need for velocity clamping. This was formalised and proved stable by Clerc and Kennedy [39] as shown in eq. (7.5);

$$\chi = \frac{2}{\left|2 - \rho - \sqrt{\rho^2 - 4\rho}\right|}, \rho = iw + sw \tag{7.5}$$

The constriction method is a particular case of the inertia weight method. In the formulation of $\chi$, a balance between the social and individuality weights with $\rho$ set to 4.1, $iw$ and $sw$ being set to 2.05, results in $\chi = 0.72984$.

The implementation of the constriction factor into eq. 7.3 is then shown in eq. 7.6;

$$v_{ij}{}^{(t)} = \chi \cdot (v_{ij}{}^{(t-1)} + iw \cdot U_{(0,1)} \cdot \left(pbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right) + sw \cdot U_{(0,1)} \cdot \left(lbest_{ij}^{(t-1)} - x_{ij}^{(t-1)}\right)) \quad (7.6)$$

Bratton and Kennedy [38] have also discussed a number of subsequent advances and interpretations of the swarm dynamics. Such features as to negate the effect of biases toward artificially good solutions are implemented. Since this presents difficulty when comparing with algorithms, such methods as the *centre offset* method are used, which negates any central bias, by moving the optimum from the point at which the algorithm is biased toward. Another method discussed, is the *region scaling* method, which initialises the swarm within a small region of the search space to which the global optima is not located within. The swarm is forced to expand. Boundary effects are also discussed, with central bias being identified as caused in part by the artificial limiting of particle solutions when reaching a boundary edge. The most common and easy solution to this problem is described as leaving the particles velocity and solution coordinates unchanged when passing the boundary so that it is statistically very likely to return to the search-space, due to the attraction of its own personal best and that of the global/local best. It is also recommended that a generous $v_{max}$ still be used since particles may be subject to high velocities beyond the boundary.

The number of particles used within a swarm was also identified by Bratton and Kennedy [38] as being highly problem specific and those chosen by various authors are different, based on the particular test suite being tackled. These developments were tested by Bratton and Kennedy with a benchmark suite of 14 problems, comparing the original formulation of PSO with that of the GBEST constricted formulation, together with the LBEST constricted model. It is conclusively shown that there is a significant improvement in using the constricted swarm over the original, furthermore there is a significant advantage of using the LBEST formalisation over the GBEST model, particularly in the case of multimodal problems.

It is concluded here that there have been many advancements of the basic algorithm, mainly with respects to the behaviour of the swarm in terms of improving the dynamics to exhibit a more convergent behaviour.

The PSO was originally intended for unconstrained problems, however, over the years various approaches have been devised for handling constraints (taking inspira-

tion from EAs) [3, 7]. These approaches generally though not exclusively fit into the category of preserving feasibility methods,

## 7.3 Local search implementation

To restate the definition in section 2.2, there is no consensus to the meaning of the term memetic algorithm, however, it is taken to be any population-based heuristic approach which is combined with a local search (i.e. to apply individual learning to a population-based heuristic approach). To quote Petalas et al. [40], memetic algorithms (MAs) were first proposed in 1989 by Moscato, where it was said to have been inspired by the notion of Memes, as defined by Richard Dawkins as a unit of cultural evolution. Simulated Annealing was used for local search refinement with a hybrid population-based GA by Petalas et al. to tackle combinatorial optimisation problems, including the travelling salesman problem. This method gained wide acceptance, due to its ability to solve difficult problems. A comparative study was made by Petalas et al. [40], coming to the conclusion that memetic algorithms to be highly superior in effectiveness and speed when compared to a purely global algorithm. This paper compared two variants on a range of problems from unconstrained, constrained, to integer programming problems: One variant employed the global PSO algorithm; and the other employed a memetic PSO with random walk with direction exploitation.

Implementations of hybrid based methods are far from uncommon amongst the optimisation community. Examples include the training of artificial neural networks for function approximation [41] or the hybrid SQP-PSO created by Victoire and Jeyakumar [42] for the economic dispatch problem. The latter is of high interest with its implementation of SQP to gbest, triggered each time the solution has improved. It was noticed by Victoire and Jeyakumar, that early on in the PSO search, particles are statistically likely to be in proximity to the global best but then move away from these areas. For this reason the local search hybrid was implemented.

The success of memetic algorithms (the combining of a stochastic search with local search) in genetic algorithms, has resulted in numerous methods of local search implementations with PSO to be considered by various authors. This includes the use of such techniques as the Hill Climbing method [43], Nelder-Mead-Simplex [44], Simulated Annealing [45] and Tabu search [46] for example. It is observed that there is much variation in the method of approach adopted by individual authors: Tandem search set-ups are where the entire population of particles is subdivided into two sub-sets, one

performing the stochastic search and the other performing the local search after which the subsets are merged; Cascade searches perform stochastic searches of all particles and a further improvement is achieved by use of a local search method for final solution refinement. Both methods of investigation and results are varied, but overall, conclusions indicate that memetic algorithms achieve highly accurate and less computationally expensive results.

A wide range of stopping criteria are considered in the literature though few are driven towards its research. Reduction of particle velocity components to a certain threshold is a method considered by Vaz et al. [47] indicating that no further improvement is likely. Another method is that used by Gimmler et al. [48] in the determination of the Euclidean distance between each particle and the best particles position, where a tolerance of clustering measure triggers a local search. Some methods of local search implementation do not involve clustering criteria at all and are triggered by a certain iteration number.

One of the most successful algorithms developed for a general purpose optimiser is by Liang and Suganthan [49], combining and coupling the SQP method with its Dynamic Multi-swarm Optimiser (DMS-PSO). Their method describes sub-populations solving their own objectives, being assigned adaptively and their assignment being periodically changed according to difficulty. For this reason the number of sub-populations is not necessarily equal to the number of objectives or constraints. This algorithm is to be further discussed in the following section as it is chosen as a highly suitable algorithm for comparison with the implementation of this thesis.

To summarise, the implementation of a local search algorithm is commonly known to be beneficial to solution refinement and to again quote Dorigo and Stützle [5], population-based heuristic approaches are likely to be considerably improved with implementation of a local search (see section 2.2). The hybridisation of a local search with the global search PSO also allows a guarantee of local optimal convergence, which the PSO alone does not achieve. Finally, termination measures and methods are rather uncommon amongst the literature, in that very little research is made in this area, offering to be a significant research opportunity.

# Chapter 8

# Benchmark and algorithm description

As described previously, an in-house particle swarm optimiser (GP-PSO) is to be enhanced (further discused in section 8.5). In this investigation, two stages are implemented: one is to determine whether solution refinement occurs by applying the local search with the pso (GP-PSO + SQP); secondly, an early triggering mechanism is to be investigated in order to save computational time in finding the global optimum (GP-PSO-SQP). That is, the later is to hybridise the two algorithms in effect by switching through some mechanism between the GP-PSO and the SQP.

This chapter concerns itself with the introduction of tools necessary to conduct this investigation. This includes a summary of the algorithms used in comparison, allowing a performance measure to be attained for the results of the investigation as shown in section 8.3. The set-up of the general-purpose PSO is highlighted in section 8.5 and regarding an early triggering mechanism, the swarm dynamics are utilised to indicate some level of stagnation as detailed in section 8.4. Since it is necessary to consider various properties of the PSO algorithm, three benchmark suits are chosen as described in section 8.1. These benchmarks consist of the unconstrained CEC05 suite (Real Parameter single objective optimisation) by Suganthan et al. [50], a constrained suite (constrained real parameter optimisation) by [51] and finally an engineering benchmark from [52],[3].

## 8.1 Benchmark description and formulation

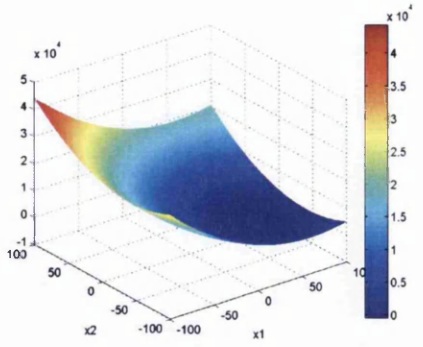
For the purpose of early termination to the local search for refinement of the solution (Memetic approach), it is necessary to consider various properties of the PSO algorithm. For this purpose, three benchmark suites are chosen for implementation, development

and testing with the combined general-purpose PSO with SQP local search (GP-PSO-SQP). One consideration is that the GP-PSO in effect behaves differently with unconstrained problems as to constrained problems due to its constraint handling technique (the conflict and constraint functions handled separately) and so the two popular test suites CEC05 (Suganthan et al. [50], unconstrained problems) and CEC06 (Liang et al. [51], constrained problems) are chosen. Finally, testing is made on a number of typical engineering problems, to which those tackled by Hu et al. [52] and Innocente [3] are chosen for consistency with the work by Innocente [3].

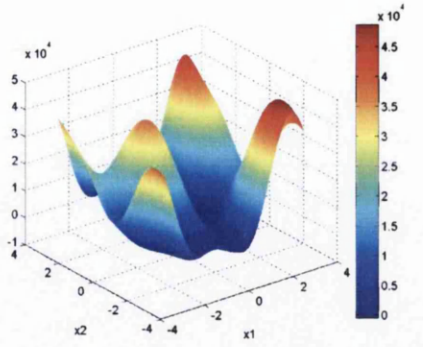
### 8.1.1 CEC05 Basic benchmark description

Within the benchmark set CEC05, five problems are chosen based on some similarity with those chosen by Innocente [3]. These include F1, F6, F7, F9 and F12 in 2-dimensions, 10-dimension and 30-dimensions (2D,10D,30D). The 50D versions are not chosen, since neither the GP-PSO nor any particle swarm optimiser that tackled this benchmark resulted in significant success rates on these problems. A plot of these functions within the imposed limits are shown in fig.8.3-8.7 together with the formulation of these problems (eqn. 8.1-8.8). For a full set of plots and formulations of the problems, the reader is referred to [50]. These are those chosen for initial testing of the algorithm (section 11.1.1 -11.1.2). For the final testing of the investigation using the unconstrained suite, the full 14 problems are investigated (see section 11.1.3). More information regarding these first 14 problems is now presented. Functions 1-5 are unimodal and functions 6-12 are multimodal, while functions 13-14 are what's described as expanded functions. An illustration of this difference is shown in fig. 8.1, with the common unimodal sphere function and the highly multimodal Schwefel's Problem.

These functions differ from their traditional formulation, in that the function is 'expanded' in such a way that variables are paired. Function 7 is a function with a global optimum outside the initialisation range (see fig. 8.5a-8.5d). Function 5 and 8 have their global optima on the bounds. Another important feature of one of the problems, is the random noise on function 'F4' (having the effect of moving the global optimum on each time-step). It should be noted that rotations and shifting occurs on some problems so as to eliminate bias toward the centre of the search space, and remove bias toward a zero value global optimum. However, this also greatly increases the difficulty of the functions with respect to their traditional formulations or at least removes the common artificial advantages that many algorithms have toward certain biases mentioned




(a) Surface plot of function1(2D) from the CEC05 test suite.




(b) Surface plot of function12(2D) from the CEC05 test suite.

Figure 8.1: Illustrative plots of the unimodal F1(2D) fig.8.1a and multimodal F12(2D) fig. 8.1b functions

previously. This restricts comparisons to be made on those algorithm's tackling this particular formulation of the problems. An example plot of the shifted Rastrigin's function is shown with and without rotation in fig.8.2. This serves to avoid bias toward the centre of the search-space.

F6(2D) is actually unimodal, (which can be observed in fig. 8.4, where it is only in the higher dimensions that is becomes multimodal. F7 (Griewank's function) and F9 (Rastrigin's function) are highly difficult multimodal problems. Finally, F12 appears to be a function that contains not just its difficulty in the number of local optima but also in the similarity of fitness between local optima and the depth of each of these valleys.

It should be noted that each of the problems is designed to offer its own type of difficulty to the algorithm applied, giving indication to the overall performance of an algorithm applied across this range of problems.

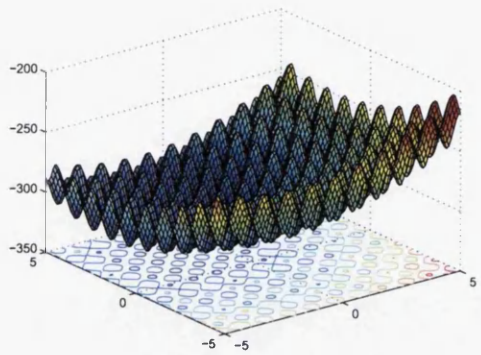
**Function1 F1: Shifted Sphere Function**

$$F1(x) = \sum_{i=1}^{n} z_i^2 + fbias, \mathbf{z} = \mathbf{x} - \mathbf{o} \tag{8.1}$$

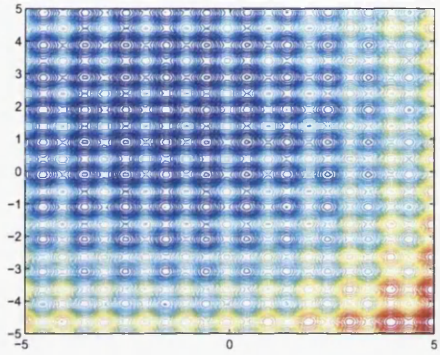
where $\mathbf{o}$ is an $n$-dimensional vector for shifting the solution coordinates, $\mathbf{x}$ the unshifted $n$-dimensional solution coordinates and $fbias$ is the shifted conflict.

$$\text{Global optimum } \mathbf{x}* = \mathbf{o}$$

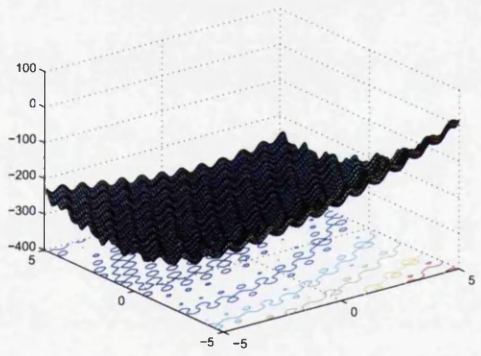
$$F1(\mathbf{x}*) = fbias = -450$$



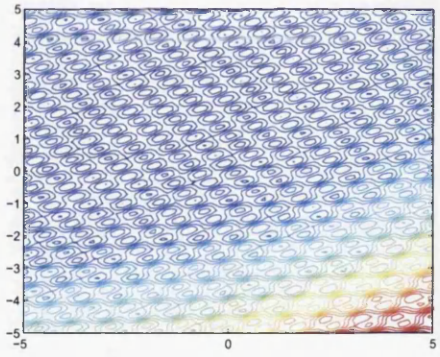

(a) 3D plot of Rastrigin's function(2D)




(b) Contour plot of Rastrigin's function(2D)




(c) 3D plot of rotated Rastrigin's function(2D)




(d) Contour plot of rotated Rastrigin's function(2D)

Figure 8.2: Rotation amongst the CEC05 suite to remove central bias.

with bounds $x \in [-100, 100]^n$

**Function1 F6: Shifted Rosenbrock's Function**

$$F6(x) = \sum_{i=1}^{n}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + fbias,\ \mathbf{z} = \mathbf{x} - \mathbf{o} + \mathbf{1} \tag{8.2}$$

$$fbias = 390 \tag{8.3}$$

with bounds $x \in [-100, 100]^n$

**Function1 F7: Shifted Rotated Griewank's Function - no bounds**

$$F7(x) = \sum_{i=1}^{n} \frac{z_i^2}{4000} - \prod_{i=1}^{n} COS(\frac{z_i}{\sqrt{i}}) + 1 + fbias,\ \mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M} \tag{8.4}$$

where M' is the linear transformation matrix used to rotate the function (see CEC05 for further details).

$$fbias = -180 \tag{8.5}$$

initialised population $x \in [0, 600]^n$ with problem artificially bound to $[-600, 600]^n$

**Function1 F9: Shifted Rastrigin's Function**

$$F9(x) = \sum_{i=1}^{n}(z_i^2 - 10COS(2\pi z_i) + 10) + fbias,\ \mathbf{z} = (\mathbf{x} - \mathbf{o}) \tag{8.6}$$

$$fbias = -330 \tag{8.7}$$

with bounds $[-5, 5]^n$

**Function1 F12: Schwefel's Problem**

$$F12(x) = \sum_{i=1}^{n}(\mathbf{A_i} - \mathbf{B_i}(\mathbf{x}))^2 + fbias,\ \mathbf{z} = (\mathbf{x} - \mathbf{o}) \tag{8.8}$$

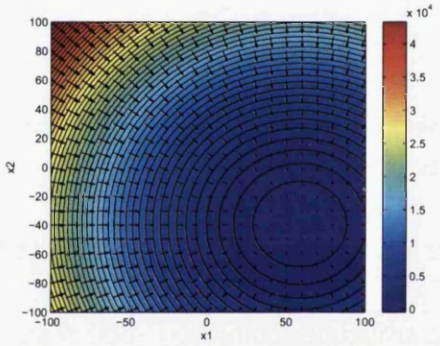
where:

$$\mathbf{A_i} = \sum_{j=1}^{n}(a_{ij}SIN\alpha_j + b_{ij}COS\alpha_j),\ \mathbf{B_i}(x) = \sum_{j=1}^{n}(a_{ij}SINx_j + b_{ij}COSx_j),\ i = 1, .., n \tag{8.9}$$

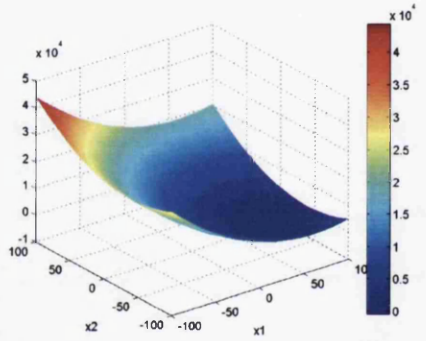
where $\alpha$ is an $n$-dimensional vector of random numbers in the range $[-\pi, \pi]$, $\mathbf{A}$ and $\mathbf{B}$ are a n-by-n dimensional matrix with $a_{ij}$ and $b_{ij}$ being integer random numbers in the range $[-100, 100]$.

$$fbias = -460 \tag{8.10}$$

bounds $[-\pi, \pi]^n$



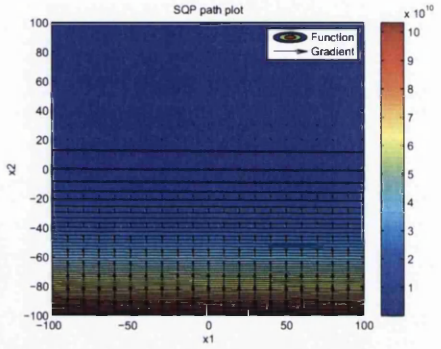

(a) Contour plot of function1(2D) from the CEC05 test suite.



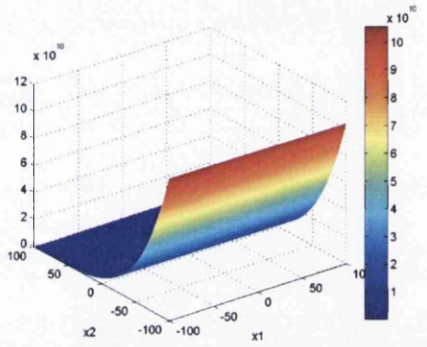

(b) Surface plot of function1(2D) from the CEC05 test suite.

Figure 8.3: Surface and contour plots of the chosen problems from the CEC05 benchmark suite.

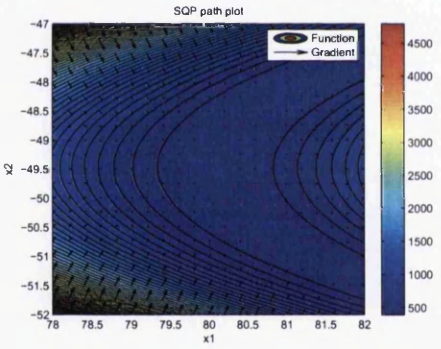
Looking at the coordinates at the end of the GP-PSO for the 2dimensional problems (for which further details can be found in appendix A.3.1), it becomes clear that problem 7 suffers from multiple attractions close to the global optimum due to the sheer number of suboptima surrounding the global minimum (close in position and conflict). Problem 12 suffers from both attractions to regions close in conflict to the global optimum, thus increasing likelihood for slowing convergence.



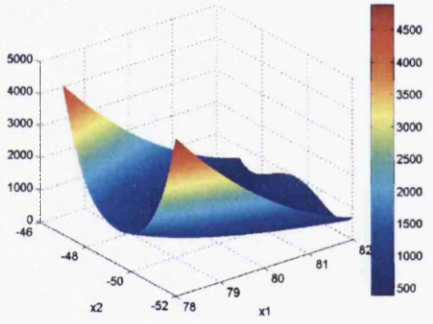

(a) Contour plot of function6(2D) from the CEC05 test suite.




(b) Surface plot of function6(2D) from the CEC05 test suite.

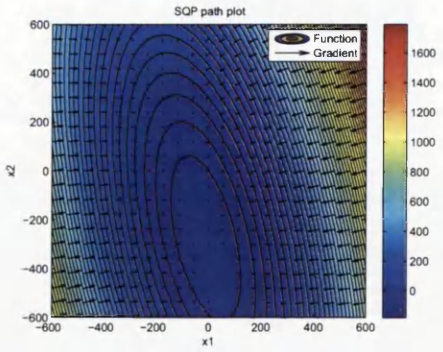



(c) Contour plot of function6(2D) from the CEC05 test suite.

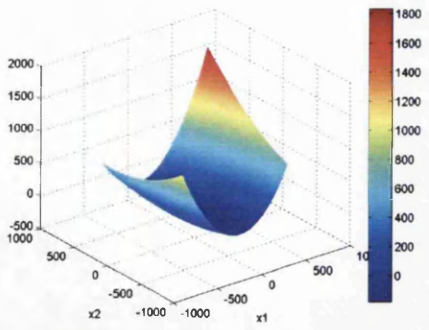



(d) Surface plot of function6(2D) from the CEC05 test suite.

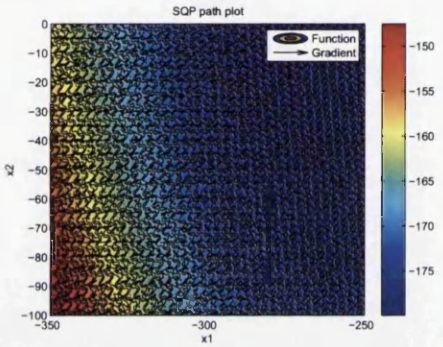
Figure 8.4: Surface and contour plots of the chosen problems from the CEC05 benchmark suite.



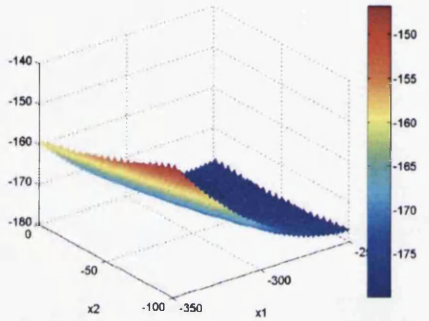

(a) Contour plot of function7(2D) from the CEC05 test suite.




(b) Surface plot of function7(2D) from the CEC05 test suite.

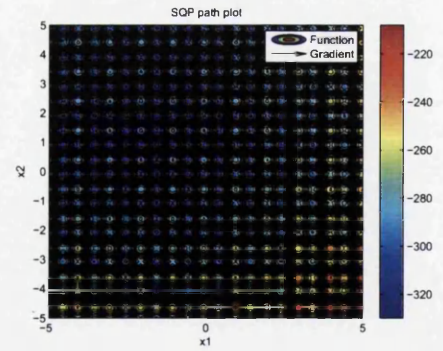



(c) Contour plot of function7(2D) from the CEC05 test suite.

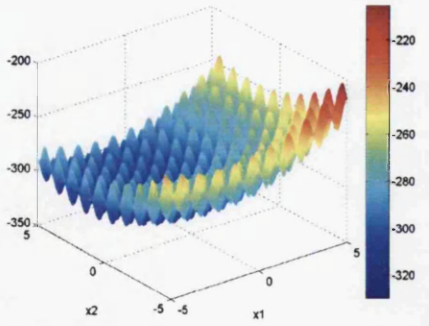



(d) Surface plot of function7(2D) from the CEC05 test suite.

Figure 8.5: Surface and contour plots of the chosen problems from the CEC05 benchmark suite.

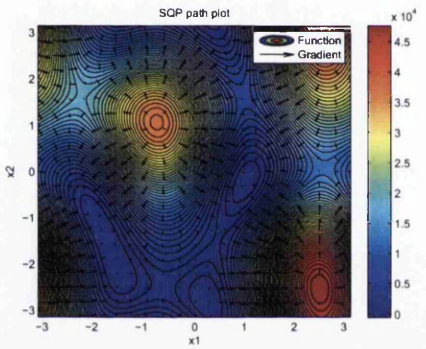



(a) Contour plot of function9(2D) from the CEC05 test suite.

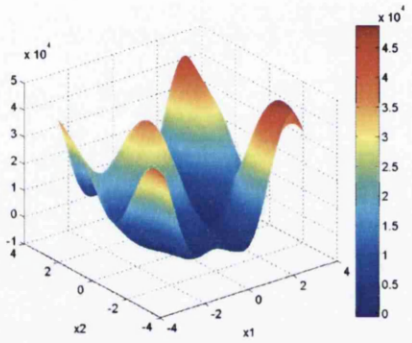



(b) Surface plot of function9(2D) from the CEC05 test suite.

Figure 8.6: Surface and contour plots of the chosen problems from the CEC05 benchmark suite.




(a) Contour plot of function12(2D) from the CEC05 test suite.




(b) Surface plot of function12(2D) from the CEC05 test suite.

Figure 8.7: Surface and contour plots of the chosen problems from the CEC05 benchmark suite.

### 8.1.2 CEC06 Basic benchmark description

Again, as with the CEC05 suite, the problems are specifically designed to test the applied algorithm features, whether by its ability to efficiently handle constraints or perhaps its ability to efficiently solve multimodal problems for example. The first 13 problems are chosen for testing (g01-g13) (sections 11.2.1-11.2.2) while the entire suite is then used for final testing (sections 11.2.3-11.2.4).

Problems in this suite are formulated with equality constraints relaxed to inequalities as follows:

$$|h_j(\bar{x})| - \epsilon \leq 0 \tag{8.11}$$

where $\epsilon$ is the equality tolerance as defined in [51] as $1e-4$.

Inequalities are then defined as;

$$g_j(\mathbf{x}) \leq 0 \tag{8.12}$$

An illustration of this relaxation of equality constraint to inequality constraint is shown in fig. 8.8. This is a standard method defined in [51] for allowing evolutionary algorithms to tackle such problems. The problem formulation can be found in Appendix. A.1.1.

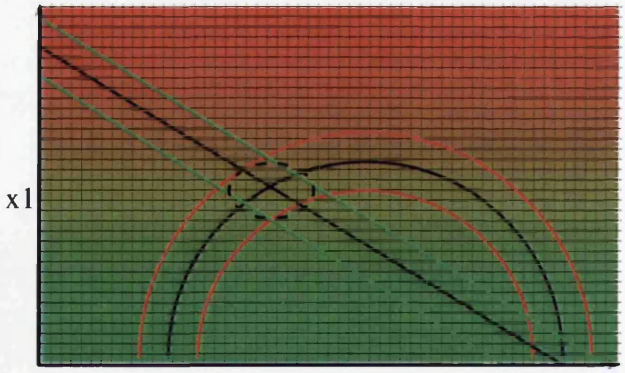
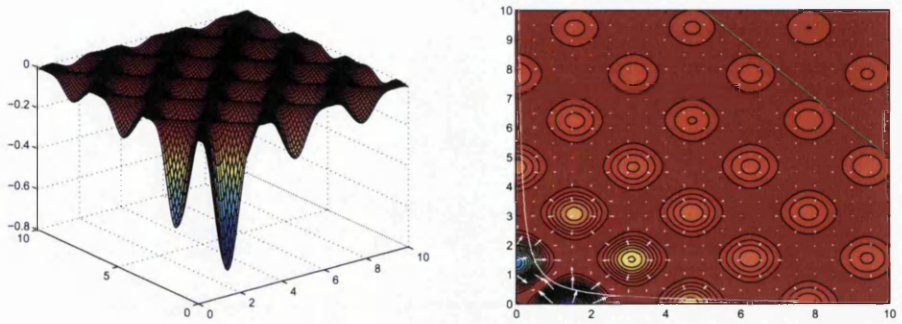



Figure 8.8: Illustration of the formulation of equality constraints as inequalities, through relaxation. Two example equality constraints are indicated by the black curves where the red and green curves correspond to the relaxed formulation of these equalities. The feasible area of the search space is then circled.

The observed behaviour of the swarm with respect to this suite is to be described in sections 9.2 and 10.1 and as such, this description is to be brief. Suffice to say, the CEC06 suite involves a greater complexity to CEC05 with its constraints. The

interested reader is directed to figs. 8.9 to 8.11 for a plot of the various functions, both 2-dimensional and those which were reducible to 2 dimensions, in order to achieve some visual understanding of the complexity of particular problems.



(a) 3D plot (b) Contour plot

Figure 8.9: Visual plot of a reduced dimension g02(2D) from the CEC06 test suite.

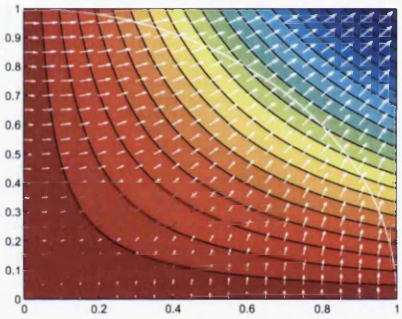


Figure 8.10: Visual plot of of a reduced dimension g03(2D) from the CEC06 test suite where the white curve represents the constraint.

This visual interpretation is limited to those chosen above (g02,g03,g06,g08 and g11), as higher dimensional problems are difficult to visualise.

Problem g02 is quite obviously a highly difficult problem, with not only its multi-modality, but also the way in which the constraints are likely to confine the directional approach of the swarm toward optimality as indicated by fig. 8.9. On the other hand, g03 is likely to be a very easy problem, as the objective function appears to be convex[1] (see fig. 8.10).

[1] "A convex function is a continuous function whose value at the midpoint of every interval in its domain does not exceed the arithmetic mean of its values at the end of the interval" [53]

With those problems originally intended as 2-dimensional functions, the path of the PSO is plotted on the contour plot of the function. From this, it is clear that g06 offers difficulty only due to its constraints, though it offers no problems for the PSO with its advance constraint handling technique (see fig. 8.11a). With g08 being multi-modal, it offers no problems for the PSO, as the constraints confine the swarm to a suitable area of the search space, where the solution is found with ease (see fig. 8.11b). Finally, g11 as shown in fig. 8.11d, is the only problem with more than one global optimum and exhibits rather interesting features with its two equally good solutions (two global optima). This offers to diversify the search until considerable refinement has occurred due to the switching between these two equally good solutions.

Other problems amongst the suite are g17, which is well known amongst the literature as being a highly difficult multi-modal problem [49, 54].

## 8.2 Engineering problems

Though this report is not heavily tilted toward applications, these typical test engineering problems taken from [7, 52], offer a fair comparison with other algorithms in the literature. These problems include: The pressure vessel problem (PVD); welded beam design (WBD); minimisation of the weight of a tension or compression spring design (TCSD); and finally, Himmelblau's non-linear optimisation problem (HBNLP). The formulation of the PVD problem is discrete, although a continuous version of this problem is tackled for application of the SQP local search, called C-PVD.

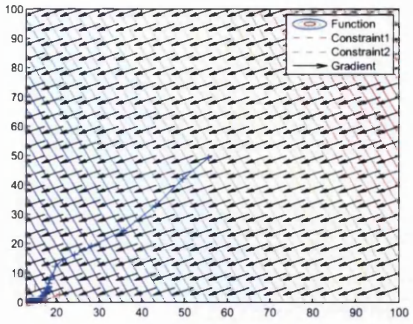
The problem formulations can be found in Appendix A.1.2.

## 8.3 Algorithms in comparison

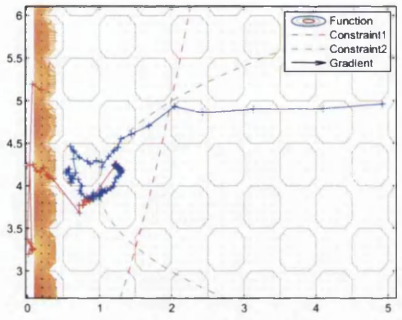
With regards to comparison with other authors in the literature, not all available algorithms are considered, but a small sub-set of cutting edge[2] algorithms which also share some features with the algorithm described here.

Regarding comparison with the CEC05 benchmark, a well known TRIBES algorithm by Clerc [55] is chosen. This algorithm aims to produce a black-box optimisation tool, which adapts to a given problem with the use of tribes (multiple swarms), with both

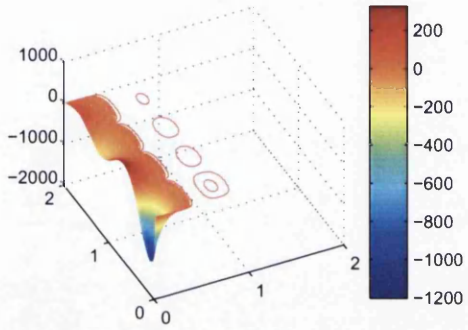
[2]Cutting edge is used to describe those algorithms that perform better than all others for the particular benchmark suite.



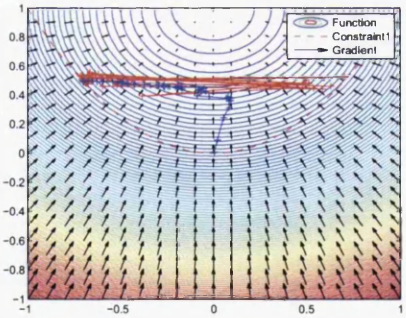

(a) g06




(b) g08




(c) g08




(d) g11

Figure 8.11: Visual plots with typical PSO paths shown.

intra and inter-tribe communication. The number in each tribe and in fact the number of tribes then changes adaptively based on the findings within the solution space. This algorithm requires no defined parameters other than those that define the problem. Further information regarding the works of Clerc and his algorithm TRIBES can be found in [55].

The second algorithm chosen for comparison in the CEC05 benchmark is the DMS-L-PSO by Liang and Suganthan [56], chosen on the grounds of it being an earlier version of the cutting edge implementation for the constrained benchmark suite (CEC06) which is to be described.

For the CEC06 benchmark, the two most successful PSO algorithms are chosen for comparison from the special session on constrained real-parameter optimisation [51]. The first, the Dynamic Multi-swarm Optimiser (DMS-PSO) by Liang and Suganthan [49] and secondly the Particle Evolutionary Swarm Optimisation Plus (PESO+) by Munoz-Zavala et al. [54].

The DMS-PSO by Liang and Suganthan [49], couples a SQP local search to their PSO algorithm (as does the GP-PSO-SQP). Their method describes sub-populations solving their own objectives (individual constraints and/or objective function), and that their number being assigned adaptively and periodically according to difficulty. Local search implementation occurs with the calling of five random pbest particles every 'n' generations (supplying it with initial solutions), and after a number of generations, the gbest particle is used to ensure final solution refinement. The random choosing of these five particles means that no preference is made of one over the other. This is consistent with the fact that particle distances from an unknown arbitrary global optimum gives rise to one 'good' particle being indistinguishable to another. This approach is somewhat similar to the upcoming GP-PSO-SQP, with their chosen hybridisation with SQP, though the continuing implementation of a local search during the global search was disregarded in the upcoming investigation, on the assumption of it both inhibiting the exploration of the global search and also increasing optimised hybridisation complexity (i.e. introducing complexity as to the most suitable way in which to apply the algorithm). Another unique aspect to this algorithm includes the update of the particle positions, where half of the dimensions are kept the same as their pbest values (with the intention of improving global search ability). Secondly, the number of sub-populations is not necessarily equal to the number of objectives or constraints, meaning that computational expense does not necessarily increase as a function of the number of constraints.

The second algorithm used in comparison is the PESO+ by Munoz-Zavala et al. [54], which does not implement a local search, though a perturbation of the solutions is made using operators (perturbation operators), which are said to be similar to those used in differential evolution. For this reason it is arguable whether this algorithm re-

mains within the canonical PSO design, or is in fact a variant of differential evolution. This algorithm implements a constraint handling technique and selection mechanism based on feasibility rules. For its neighbourhood, it uses a ring topology, which are all design choices for the purpose of keeping diversity and ensuring increased exploration. The constraint and selection mechanisms are said to be driven by feasibility and dominance. Initially, the standard PSO algorithm is performed, then the perturbation operator (called 'C-Perturbation') is applied to the entire swarm and recorded as a temporary set. Each of the temporary members are then compared with the corresponding pbest members and are replaced if the former is better. Subsequently, each particle is again perturbed, though with some probability on each component ('D-Perturbation') and assigned to a temporary set. Each member is then compared with its corresponding pbest. By this method, only the best locations (pbest) are altered by the perturbation operators. Another feature added was the implementation of an external file that records a list of the particles best tolerant solutions, so that feasible solutions are not lost through the perturbation. By this method, the perturbation operator compares its feasibility with that of the recorded values. Munoz-Zavala et al. describes the algorithms insensitivity to parameter adjustments and this re-emphasises here that their algorithm no longer describes a particle swarm but bares more resemblance to differential evolution.

Regarding the real engineering problems for final testing, a number of authors are used in comparison to give an overall indication of performance. These problems are taken from a paper by Hu et al. [52].

It should be noted that comparisons are by no means comprehensive, but give indication as to the strength and weaknesses of the GP-PSO-SQP algorithm, since the chosen algorithms exhibit features that are somewhat similar whilst different in other respects.

## 8.4 Measures of the swarm

Measures derived by Innocente [3] for the measurement of swarm dynamics (clustering, diversity and stagnation) are used in this investigation for the purpose of identifying where a local search may be triggered for early switch-over. Such measures include clustering and evolution measures. Furthermore, evolution of clustering measures are newly derived, relating the difference between successive time-steps of the clustering measures. However, the actual formalisation of these measures takes a relative approach

with the use of a swarm in search of the maximum, rather than the minimum. Also a smoothing with previous time-steps occurs with the number recorded as 'Tref' as defined by Innocente [3]. The clustering and evolution measures are now described. It should be noted that unless stated otherwise, all measures use each particles personal best (pbest) solutions to calculate these measures rather than the current time-step solutions within the swarm.

**Clustering measures as derived by Innocente [16]**

$$cb\_me^{(t)} = \frac{\sum_{i=t-tref+1}^{t} \left(\overline{c}^{(i)} - cgbest^{(i)}\right)}{tref.(cgworst^{(t)} - cgbest^{(t)})} \tag{8.13}$$

$$pb\_me^{(t)} = \frac{\sum_{i=t-tref+1}^{t} \sqrt{\sum_{j=1}^{n} \frac{\sum_{k=1}^{m} \left(x_{kj}^{(i)} - gbest_j^{(i)}\right)^2}{m.(x_j max - x_j min)^2}}}{tref.\sqrt{n}} \tag{8.14}$$

$$pb\_cge^{(t)} = \frac{\sum_{i=t-tref+1}^{t} \sqrt{\sum_{j=1}^{n} \left(\frac{cg_j^{(i)} - gbest_j^{(i)}}{x_j max - x_j min}\right)^2}}{tref.\sqrt{n}} \tag{8.15}$$

**Evolution measures as derived by Innocente [16]**

$$cb\_av^{(t)} = \frac{\sum_{i=t-tref+1}^{t} abs\left(\overline{c}^{(i)} - \overline{c}^{(i-1)}\right)}{tref.\left(cgworst^{(t)} - cgbest^{(t)}\right)} \tag{8.16}$$

$$cb\_best^{(t)} = \frac{\left(cgbest^{(t-tref)} - cgbest^{(t)}\right)}{tref.\left(cgworst^{(t)} - cgbest^{(t)}\right)} \tag{8.17}$$

$$pb\_cg^{(t)} = \frac{\sum_{i=t-tref+1}^{t} \sqrt{\sum_{j=1}^{n} \left(\frac{cg_j^{(i)} - cg_j^{(i-1)}}{x_j max - x_j min}\right)^2}}{tref.\sqrt{n}} \tag{8.18}$$

$$pb\_gbest^{(t)} = \frac{\sum_{i=t-tref+1}^{t} \sqrt{\sum_{j=1}^{n} \left(\frac{gbest_j^{(i)} - gbest_j^{(i-1)}}{x_j max - x_j min}\right)^2}}{tref.\sqrt{n}} \tag{8.19}$$

**New evolution of clustering measures**

$$ev\_cb\_me^{(t)} = \frac{\sum_{i=t-tref+1}^{t} abs\left[(\overline{c}^{(i)} - cgbest^{(i)}) - (\overline{c}^{(i-1)} - cgbest^{(i-1)})\right]}{tref.(cgworst^{(t)} - cgbest^{(t)})} \tag{8.20}$$

$$ev\_pb\_me^{(t)} = \frac{\sum_{i=t-tref+1}^{t} abs\left[A - B\right]}{tref \cdot (x_{jmax} - x_{jmin}) \cdot \sqrt{m \cdot n}} \tag{8.21}$$

where $A = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{m}(x_{kj}^{(i)} - gbest_j^{(i)})^2}$, $B = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{m}(x_{kj}^{(i-1)} - gbest_j^{(i-1)})^2}$

$$ev\_pb\_cge^{(t)} = \frac{\sum_{i=t-tref+1}^{t} abs\left[A - B\right]}{tref \cdot (x_jmax - x_jmin) \cdot \sqrt{n}} \tag{8.22}$$

where $A = \sqrt{\sum_{j=1}^{n}(cg_j^{(i)} - gbest_j^{(i)})^2}$, $B = \sqrt{\sum_{j=1}^{n}(cg_j^{(i-1)} - gbest_j^{(i-1)})^2}$

These measures are then itemised for clarity to the reader;

1. $cb\_me^{(t)}$ describes the average difference (in the last *tref* time-steps) between the best and average conflict in the swarm in relation to the difference between the best and worst conflicts found up until the current time-step, where;
   - $\overline{c}^{(i)}$ is the $i$'th time-step average conflict
   - $cgbest^{(i)}$, the best conflict found in the swarm up until time-step $i$
   - $cgworst^{(t)}$ the worst conflict found in the swarm up until time-step $t$
   - $t$ is the current time-step.
2. $pb\_me^{(t)}$ describes the distance between each particle and the best solution. Note that a square root of the average of the squared normalised value is taken (normalised across the bounds of the n-dimensional space), where;
   - $x_{kj}^{(i)}$ is $k$'th particles $j$'th solution coordinate at time-step $i$

- $gbest_j^{(i)}$ is the best solutions $j$'th coordinate found up until time-step $i$
- $x_{jmax}$ and $x_{jmin}$ are the $j$'th dimension bounds of the problem
- $m$ the number of particles and $n$ the number of dimensions

3. $pb\_cge^{(t)}$ describes the distance between the centre of gravity solution and the best solution, where values are again the square root of the squared normalised distances, where;
   - $cg_j^{(i)}$ is the centre of gravity of the $j$'th solution coordinate at time-step $i$
4. $cb\_av^{(t)}$ describes the average (amongst the last tref time-steps) difference between the current average conflict and the preceding conflict, normalised to the distance between the current best and worst conflict found up until time-step $t$.
5. $cb\_best^{(t)}$ is similar to the above, only that it is the difference between the current best and the preceding one.
6. $pb\_cg^{(t)}$ is the average (in the last tref time-steps) of the square root of the squared normalised distance between the current centre of gravity solution coordinates and the preceding one.
7. $pb\_gbest^{(t)}$ is as above only that it is regarding the current and previous current best solution coordinates.

| Type | Measure | Based | Simplified meaning (related meaning) |
|---|---|---|---|
| Clustering | cb_me | Conflict | Conflict(best) - Conflict(average) |
| | pb_me | Position | Average(Solution(best) - Solution(particle(k))) |
| | pb_cge | Position | Solution(best) - Solution(cg) |
| Evolution | cb_av | Conflict | Conflict(average(i)) - Conflict(average(i-1)) |
| | cb_best | Conflict | Conflict(best(i)) - Conflict(best(i-1)) |
| | pb_cge | Position | Solution(cg(i)) - Solution(cg(i-1)) |
| | pb_gbest | Position | Solution(best(i)) - Solution(best(i-1)) |
| Evolution of Clustering | cb_me | Conflict | cb_me(i) - cb_me(i-1) |
| | pb_me | Position | pb_me(i) - pb_me(i-1) |
| | pb_cge | Position | cb_cge(i) - cb_cge(i-1) |

Table 8.1: Simplified measure definition for quick reference for the reader. The abbreviation *cg* corresponds to the Centre of Gravity.

The newly defined evolution of clustering measures can then be described as they are named, as evolution of the clustering measures, while noting that normalisation

occurs according to the current time-step measure and not with regard to the current and preceding one, as would be the case if they were defined as the differential of the clustering measures.

For simplification and ease of reading, table 8.1 summarises and simplifies those measures described.

## 8.5 GP-PSO set-up and description

**Regarding the investigation for the constrained problem set (CEC06 and the real-world engineering problems), the number of repeat runs is 20 with a 10000 time-step limit imposed. To remain consistent with Liang et al. [51], a relaxation of $1e-4$ of its equality constraints is made (by their formulation as inequalities). The number of particles in the maximiser is 10 and the number in the maximiser is 50. For the CEC05 suite, a time-step limit of 20000 is imposed with 25 repeat runs for the derivation of switching thresholds. A function evaluation (FE) limit of $1e4 \times D$ is defined for consistency with Suganthan et al. [50] for the final testing of the algorithm which is to be further discussed ($D$ corresponds to the dimension of the particular problem). Any particularities in the investigation (where different from the above) are mentioned where applicable.**

A simple description of the General-Purpose PSO (GP-PSO) set-up used throughout this investigation is as follows:

- the neighbourhood topology used: Forward topology (similar to the ring topology only that interconnections are not bidirectional). The neighbourhood being those which a particle communicates (exchanges information) with.

- neighbourhood size: either fixed (3-3) or linearly changed with time-step from 0 (no neighbours) to fully global at $t = 0.8t_{max}$ (dynamic neighbourhood size). This results in a delay of convergence, as particles exchange information between only direct neighbours at the beginning. It then takes a number of iterations for the findings of one particle to reach its most distantly indexed neighbour. Toward the end of the search, with it being global, the entire swarm becomes informed of the findings of each particle, resulting in fast convergent properties. This behaviour aids in exploration at the beginning of the search and a refined convergent behaviour toward the end.

- 3 sub-neighbourhoods and 3 sub-swarms are used as shown in fig. 8.12. Neighbourhoods referring to 3 separate indexing of particle neighbourhood topologies (all forward topologies). Sub-swarm referring to the variables of the algorithm for these given group of particles. Each of these sub-swarms has its own parameter set, aiding the search by exhibiting different swarm dynamic properties (one has the tendency to overshoot the attractor while another has the tendency to jump to the attractor for example).
- Particles used in the minimiser: 50. This is the swarm which is to optimise)
- Particles used in the maximiser: 10. The swarm in search of the maximum allows relative convergence and evolution measures to be derived (the reader is referred to section 8.4).
- Initialisation method: Independent best (maximin normalised squared distance) Latin-Hypercube (LH) initialisation.

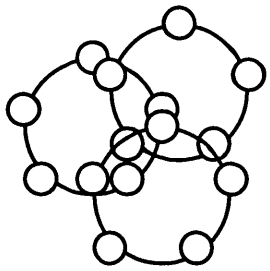


Figure 8.12: Illustration of multiple ring topology neighbourhoods in PSO

The equality and inequality tolerances are relaxed and a feasibility ratio determined by randomly generating 1000 solutions, where these tolerances are then changed (increased or decreased by a factor 10) in order to keep a feasibility ratio (FR) within the target FR (20% as user defined). The inequality tolerances are defined as 10% the equality tolerances. The most significant point to be mentioned, is that the tolerances are forced to their final values at 80% of the search duration [3], regardless of the FR at this point. Regarding the neighbourhood size, this has been kept fixed for the CEC05 suite until the final testing, while a dynamic neighbourhood is used for the CEC06 suite throughout.

Finally, with regards to the constraint handling technique, the GP-PSO has a multitude of possible options. However, a preserving feasibility with priority rules method was chosen here, which handles the conflict and constraints separately. When feasible,

the lowest conflict is considered best, while when a particle is infeasible, its conflict is not evaluated and it is judged solely on its constraint. This has the effect of drawing in the infeasible particle back to the feasible area of the search-space. The priority rules then give feasible particles higher priority over any infeasible ones, and amongst infeasible particles, those closest to being feasible are considered better (this is important in the consideration of the local best amongst the neighbourhood).

For further details, the reader is referred to the PhD thesis of Innocente [3].

# Chapter 9

# Solution refinement with a local search algorithm

This chapter concerns itself with whether the solutions provided by the GP-PSO are able to be refined by the SQP algorithm. Also, the point at which the GP-PSO successfully attains the global optimum is investigated (as defined by the benchmark), similarly the point at which the SQP attains success when applied at each GP-PSO time-step independently. That is, this chapter considers the two algorithms separately. section 9.1 details the investigation with respect to the unconstrained suite (CEC05), while section 9.2 details the investigation to the constrained suite (CEC06).

## 9.1 Refinement of solutions with use of a local search within CEC05

Firstly, consideration is given to establish the extent that a local search will improve the results of the GP-PSO and secondly, whether such results would warrant its use. Only the test problems F1, F6, F7, F9 and F12 are considered at this stage, since the rest of the suite is excluded for testing, once switching criteria has been derived. The results of this investigation are shown in fig. 9.1.

No improvement is observed on any of the 2D problems considered (as shown in fig. 9.2), likely due to the ability of the GP-PSO to easily converge on these functions, however, considerable refinement of mean solutions are shown on problems F7, F9 and F12 in both 10 and 30 dimensions. Another important result of this investigation, reveals that the difficulty met on tackling a problem, does not necessarily relate to the number

| END | problem | acceptable err | SQP success | error best | mean | stdev | GP-PSO success | error best | mean | stdev |
|---|---|---|---|---|---|---|---|---|---|---|
| 2D | F1 | 1E-06 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 |
| | F6 | 1E-02 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 |
| | F7 | 1E-02 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 |
| | F9 | 1E-02 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 |
| | F12 | 1E-02 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 |
| 10D | F1 | 1E-06 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 | 25 | 0.0E+00 | 0.0E+00 | 0.0E+00 |
| | F6 | 1E-02 | 23 | 4.3E-10 | 3.2E-01 | 1.1E+00 | 21 | 4.0E-09 | 3.2E-01 | 1.1E+00 |
| | F7 | 1E-02 | 1 | 7.4E-03 | 9.7E-02 | 4.5E-02 | 1 | 7.4E-03 | 1.4E-01 | 5.7E-02 |
| | F9 | 1E-02 | 13 | 0.0E+00 | 8.0E-01 | 9.9E-01 | 12 | 0.0E+00 | 8.5E-01 | 1.0E+00 |
| | F12 | 1E-02 | 11 | 0.0E+00 | 7.4E+00 | 7.4E+00 | 7 | 5.2E-12 | 6.9E+01 | 2.0E+02 |
| 30D | F1 | 1E-06 | 25 | 5.7E-14 | 5.7E-14 | 2.8E-14 | 25 | 5.7E-14 | 5.7E-14 | 2.8E-14 |
| | F6 | 1E-02 | 18 | 5.9E-10 | 1.1E+00 | 1.8E+00 | 3 | 1.6E-03 | 1.0E+01 | 1.9E+01 |
| | F7 | 1E-02 | 15 | 4.5E-13 | 1.7E-02 | 1.6E-02 | 15 | 4.5E-13 | 1.7E-02 | 1.6E-02 |
| | F9 | 1E-02 | 0 | 2.5E+01 | 4.7E+01 | 1.3E+01 | 0 | 2.7E+01 | 5.7E+01 | 1.6E+01 |
| | F12 | 1E-02 | 1 | 2.8E-13 | 7.9E+03 | 8.5E+03 | 0 | 8.4E+00 | 2.3E+04 | 2.6E+04 |

Figure 9.1: Accuracy of the GP-PSO compared to solutions of the GP-PSO refined by the SQP. Error is the mean normalised difference with respect to the solution found and the known global optimum.

of dimensions, in that a function is not necessarily more difficult to solve compared to its lower dimensional partner.

Problems showing considerable improvement of their final gbest solution, are functions 6(10D & 30D) and 12(30D). The PSO is limited in its ability to refine its solutions due to its heuristic population approach, and in fact a situation can arise where the population reduces to a 'one particle' like swarm (diversity lost). This can occur if the gbest solution is contained within a valley and all other pbest solutions exist in a location where the gbest particle is at the very edge of this population. In this case, if the inertia is lost, convergence to the global optimum solution at the bottom of the valley will not be possible. The combined components of the pbest/lbest and gbest solutions in eq. 7.4 do not allow the particle to further its way past the gbest solution. From this, it is meant that the swarm will make its way closer and closer to the gbest solution but with a smaller and smaller steps (never overshooting it).

From this investigation, it becomes clear that the application of a local search is a highly logical approach to refinement of the final solutions provided by the GP-PSO.

The next consideration is in the early triggering of the local search. The SQP local search attains a successful result (within the tolerance defined by CEC05) before the GP-PSO in every case, as shown in fig. 9.2. This strongly suggests the development of an early triggering mechanism for early take-over. To decide whether an early switching to local search would be suitable, the SQP is applied on every time-step of the GP-PSO search without returning to the GP-PSO its solution (i.e. to run the SQP with the solution provided by the GP-PSO independently). This is done to determine the point at which the SQP becomes successful. Since the application of the SQP to every time-step is so computationally expensive, instead, the GP-PSO search history is recorded then a

| | PROB | GP-PSO-SQP PSO mean Iteration | PSO mean FE | GP-PSO PSO mean Iteration | PSO mean FE |
|---|---|---|---|---|---|
| 2D | F1 | 1 | 0E+00 | 57 | 3E+03 |
| | F6 | 1 | 0E+00 | 128 | 7E+03 |
| | F7 | 58 | 3E+03 | 81 | 4E+03 |
| | F9 | 14 | 9E+02 | 42 | 2E+03 |
| | F12 | 1 | 0E+00 | 39 | 2E+03 |
| 10D | F1 | 1 | 0E+00 | 304 | 2E+04 |
| | F6 | 227 | 1E+04 | 7825 | 4E+05 |
| | F7 | 168 | 8E+03 | 3891 | 2E+05 |
| | F9 | 6677 | 4E+05 | 8695 | 5E+05 |
| | F12 | 293 | 2E+04 | 5227 | 3E+05 |
| 30D | F1 | 1 | 0E+00 | 948 | 4E+04 |
| | F6 | 890 | 4E+04 | 18369 | 9E+05 |
| | F7 | 1129 | 4E+04 | 2758 | 1E+05 |
| | F9 | NA | NA | NA | NA |
| | F12 | 338 | 2E+04 | NA | NA |

Figure 9.2: Point at which the GP-PSO was deemed successful as defined by the suite, compared to GP-PSO refined by the SQP.

scanning method is used to determine the point at which the SQP becomes successful. This scanning method is illustrated in the pseudo-algorithm in appendix A.2(alg. 11). The point at which the SQP becomes successful is shown in fig. 9.2.

As shown from this table, a number of the 2 dimensional cases are successful from the very first iteration. This results from the simplicity of the problems together with the initial sampling of the search space with the initialisation of the swarm. F1 then continues to be successful from the very first time-step in all dimensions, since it is in fact unimodal (as should all unimodal problems). In all cases observed, the refinement of the GP-PSO solutions using local search results in the finding of the global optimum much sooner than the GP-PSO alone.

## 9.2 Refinement of solutions with use of a local search within CEC06

As with the CEC05 benchmark (see section 9.1), the final solutions of the GP-PSO are used to ascertain whether a refinement of solutions is necessary with use of the SQP local search. Firstly, the problems that the GP-PSO finds greatest difficulty are identified. The results of this investigation are shown in fig. 9.3.

Problems g02, g03, g05, g07, g09, g10 and g13 have identified themselves as having less than 100% success by the GP-PSO as shown in fig. 9.3. However, a number of the difficulties can be readily understood. g02 has its difficulty with its high multimodality as previously highlighted in section 8.1.2, where the initial starting point for the SQP depend on the GP-PSO having located the global optimum, or be within such a proximity that it is not attracted towards others. The 2D case of the function g03 has its

| problem | GP-PSO-SQP success | error best | mean | stdev | GP-PSO success | mean FE | error best | mean | stdev | GP-PSO mean FE_err |
|---|---|---|---|---|---|---|---|---|---|---|
| g01 | 20 | 0.0E+00 | 861.5E-15 | 0.0E+00 | 20 | 5.6E+04 | 0.0E+00 | 1.3E-12 | 5.9E-12 | |
| g02 | 6 | 26.1E-15 | 16.2E-03 | 237.1E-15 | 6 | 2.7E+05 | 3.9E-06 | 16.2E-03 | 18.8E-03 | |
| g03 | 20 | 665.0E-15 | 36.9E-15 | 34.6E-15 | 18 | 3.3E+04 | 3.1E-06 | 29.7E-06 | 48.8E-06 | |
| g04 | 20 | 25.5E-12 | 25.5E-12 | 0.0E+00 | 20 | 4.6E+04 | 25.5E-12 | 25.5E-12 | 3.9E-12 | |
| g05 | 20 | 1.8E-12 | 0.0E+00 | 100.3E-15 | 0 | 1.2E+05 | 2.6E-03 | 599.2E-03 | 1.4E+00 | |
| g06 | 20 | 16.4E-12 | 14.6E-12 | 0.0E+00 | 20 | 4.0E+04 | 16.4E-12 | 14.6E-12 | 1.9E-12 | |
| g07 | 20 | 412.1E-15 | 238.0E-15 | 331.2E-15 | 0 | 9.0E+04 | 15.5E-03 | 163.4E-03 | 152.8E-03 | |
| g08 | 20 | 27.8E-18 | 0.0E+00 | 504.5E-12 | 20 | 1.3E+04 | 27.8E-18 | 0.0E+00 | 31.8E-18 | |
| g09 | 20 | 795.8E-15 | 227.4E-15 | 208.8E-15 | 0 | 1.7E+05 | 866.0E-06 | 4.8E-03 | 2.6E-03 | |
| g10 | 17 | 6.4E-12 | 2.7E-12 | 188.8E-06 | 0 | 8.8E+04 | 4.6E+00 | 85.8E+00 | 105.3E+00 | |
| g11 | 20 | 111.0E-18 | 158.5E-15 | 452.2E-15 | 20 | 1.2E+04 | 8.1E-12 | 6.5E-09 | 12.4E-09 | |
| g12 | 20 | 0.0E+00 | 0.0E+00 | 3.4E-15 | 20 | 9.0E+03 | 0.0E+00 | 0.0E+00 | 0.0E+00 | |
| g13 | 20 | 25.5E-15 | 2.5E-15 | 205.5E-15 | 14 | 6.3E+04 | 4.8E-06 | 612.3E-06 | 1.6E-03 | |

Figure 9.3: Accuracy of the GP-PSO compared to the final solution refinement with SQP for the CEC06.

less than 100% success rate due to it previously discussed inability to converge within the desired level of accuracy of the suite ($1e-4$). This is due to a loss of diversity (see appendix A.3.2 fig. 8.10). g05 is thought to present difficulty to the GP-PSO due to either the equality constraints or perhaps the scaling of the variables (this is to be further investigated). However, no problem is met with the performance of the SQP with respect to this function. Functions g07 and g09 show a similar behaviour with solutions close to the global optimum and being feasible but not within tolerance of the global optimum, unlike with application of the SQP (whether due to limitations of the swarm dynamics through a loss of diversity or some other reason). g10 is not well understood since its conflict function is linear and those constraints which are nonlinear (three of six constraints) do not immediately appear to offer difficulty (the reader is referred to [51] for the formulations of these problems). Due to g10 being a multidimensional problem, it is mere speculation on the part of the author of this document, as visualisation is difficult (visualisation is restricted to 2-dimensional problems or those problems which can be reduced mathematically to 2-dimensions).

More analysis on these problems is to be made with the implementation of the stopping/switching criteria, together with convergence analysis. It should be noted however, that refinement of the solution is achieved in nearly all problems, with considerable improvement observed in g03, g07, g09 and g10.

## 9.3 Summary

From the two benchmark suites discussed, it is clear that solution refinement is apparent in both. Secondly, the local search is able to refine a solution provided by the GP-PSO

to the global optimum (within the required accuracy[1]) at a time-step much earlier than the GP-PSO. The consequence of which is to further investigate the application of an early switching mechanism between the GP-PSO and the SQP.

[1]This accuracy is again defined as the distance between the conflict with the known published global optimum for the particular benchmark suite

# Chapter 10

# Convergence properties of the GP-PSO

As with chapter 9, the two algorithms (GP-PSO and SQP) remain independent from one-another here. This chapter primarily concerns itself with the swarm dynamics of the GP-PSO with respect to the unconstrained suite (section 10.1) and to the constrained suite (section 10.2). Such properties are important when combining two algorithms of such differing approaches, allowing the identification of possible strengths and weaknesses of the algorithm. This also allows any issues to be discovered and highlighted with respect to the development of any early triggering mechanisms.

## 10.1 Convergence properties of the CEC05 benchmark

The five chosen problems of the CEC05 benchmark (F1,F6,F7,F9 and F12) are investigated in terms of their convergence in order to determine the difficulty of these problems and appropriate switching mechanisms. Another important issue to be considered, where early switch-over criteria are concerned, is whether the search is likely to have converged within the time-step limit imposed. To this end, swarm dynamic measures are studied as presented in section 8.4, as well as the solution coordinate and conflict histories.

The set-up considers a *popbest* population, where measures extracted from the swarm are those which use pbest particles (each of the members personal bests) rather than current time-step values. A fixed neighbourhood of 3-3 is also used for simplicity, which has the effect of delayed clustering. The difficulty of the problems can then be judged from the resulting swarm dynamic measures.

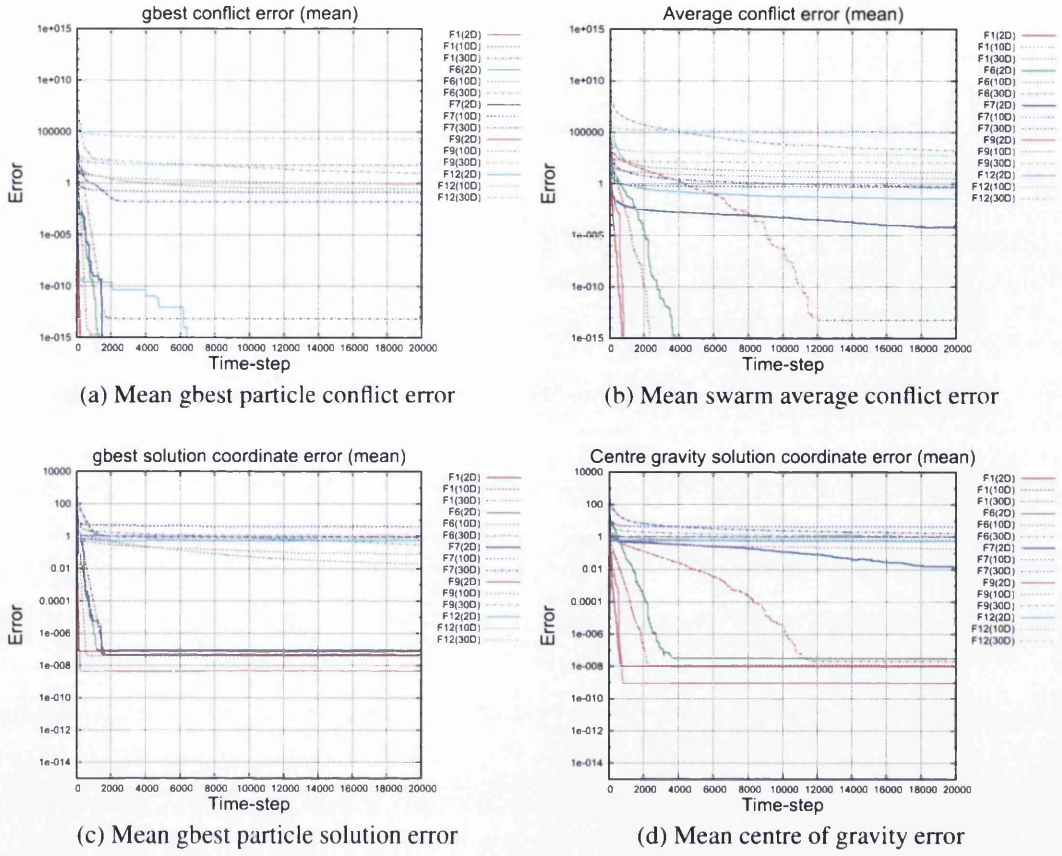
A plot of the mean conflict error and mean solution error (normalised according to the known global optimum conflict and solution coordinates) as a function of time-

steps is shown in fig. 10.1. By error, it is meant in the same context as the previously defined 'accuracy' and is used interchangeably. From fig. 10.1a it appears that only F6(30D) shows continual improvement on average for the gbest particle (i.e. beyond the limits imposed in this investigation). However, fig. 10.1b shows a mean average conflict (among the 25 runs), which has not stagnated and indicates that diversity is not entirely lost, since pbest particles are still converging to the gbest particle in problems F6(30D) and F7. It is also clear from fig. 10.1c, that function F6(10D & 30D) has also yet to converge in solution, as the coordinates of gbest have yet to stagnate. From the centre of gravity solution shown in fig. 10.1d, it is clear that function F7 does not lose diversity toward the end of the search.

These are important considerations while using or implementing switching criteria, where a search may stop before diversity is lost and that measures maybe derived amongst problems that do not reach full stagnation. This appears to confirm that the best method to derive switching measures is to search for the minimum mean magnitude of each value and then due to erratic behaviour (characteristic of heuristic algorithms) to pick the maximum value amongst the chosen function (since this ensures triggering for each case). This follows the same method of threshold derivation as Innocente [7].

Analysing individual samples indicates further information regarding the convergence of the algorithm, especially since termination or switch-over occurs using information gathered on a single sample basis. In particular, the best performing sample is analysed, where similar observations to the above are made, such as F6(10D & 30D) and F12(30D) failing to stagnate. Continual improvement is observed of gbest on these functions, however, the centre of gravity shows next to no improvement, which strongly suggests that the swarm has lost diversity and is perhaps acting as one particle. These are some of the weaknesses of the GP-PSO for which the use of a local search is hoped to alleviate. It should be noted that the cutting-edge GP-PSO considers a dynamic neighbourhood and when activated, it is expected to result in a stagnation-like behaviour at an earlier time-step.

It is now considered alongside the observations made above, the measures to be used for early switching or termination. Plots of these measures for the 2 dimensional cases of these problems are shown in appendix A.3.1 (fig. 8.3-8.7). These figures indicate that the clustering magnitudes differ greatly, specifically in F7 and F12 from the others amongst the 2 dimensional cases considered, where it is obvious that the level of clustering on these functions to be poor. An illustrative case is presented here as shown in fig. 10.2, where F1(2D) quite clearly clusters well, with its clustering measures being




(a) Mean gbest particle conflict error

(b) Mean swarm average conflict error

(c) Mean gbest particle solution error

(d) Mean centre of gravity error

Figure 10.1: Unconstrained mean problem convergence

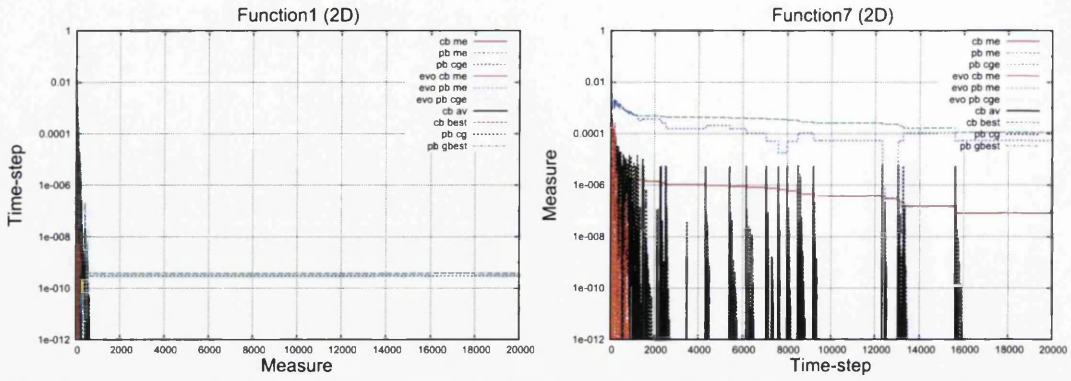
of such small magnitude, while F7(2D) clearly does not.

A full set of plots of these functions together with the final solution coordinates and observations of convergence is shown and described in appendix A.3.1.

## 10.2 Convergence properties of the CEC06 benchmark

As with CEC05, in order to decide on suitable measures for switch-over and gain an insight into the problems tackled by the PSO, solution convergence and stagnation with respect to the gbest particle is analysed across the problems considered.

From fig. 10.3a, it is clear that the majority of problems used for the determination of clustering criteria converge with respect to their objective function. Exceptions to this are, problems g03, g07 and g09. The latter two coincide with the problems that the PSO finds greatest difficulty with, as discussed in sections 8.1.2 and 9.2. However, it




(a) All measures, indicating the level of stagnation in Function1 (2D)

(b) All measures, indicating the level of stagnation in Function7 (2D)

Figure 10.2: Illustrative example of the difference between different swarm dyamic mesures for the CEC05 suite.

is interesting to note, that the mean standard deviation and mean conflict across the 25 samples has stagnated for problem g02, signifying that the algorithm becomes stuck in suboptimal solutions very early on in the search (this results from the numerous highly attractive areas of the search space). These suboptimal regions of the search space can be seen clearly in the reduced 2-dimensional version of this function as discussed in section 8.1.2.

It should be noted that g11 does not stagnate in the conflict found since it has two equally good solutions and as such, diversity is kept for a significant amount of time (a plot of this function can be found in section 8.1.2. Regarding problem g10 for which the PSO finds great difficulty (and indeed so does the local search), there is little to tell from the mean conflict plot since it appears that the solution is far from optimum and that no improvement is observed (fig. 10.3a). This occurs fairly early on in the search and so little is understood as to the difficulty of this problem.

With respect to the constraint violation shown in fig. 10.3b, problems g05, g13 and g10 have identified themselves as violating the constraints right up until 80% of the total search (section 8.5). From this observation, it is clear that the swarm struggles to obtain and maintain a 20% feasibility and so final tolerance is only reached at 80% length of the search (indicated by fig. 10.3d,10.3e). Relaxation of constraints is removed at this point regardless of feasibility.

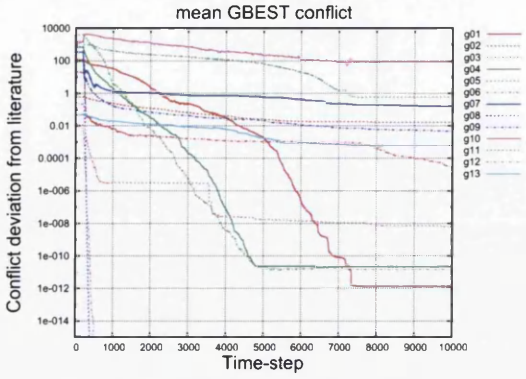
This is particularly interesting for problems g05 and g10, since this identifies the difficulty as within the constraints. g05 has shown that is has a significant difference between scale of the different constraints, but whether this is a problem for the PSO is

not so clear. The problem of both conflict and constraint violation for g13 also seems to signify that considerable difficulty is apparent, which results in a less than 100% success rate.

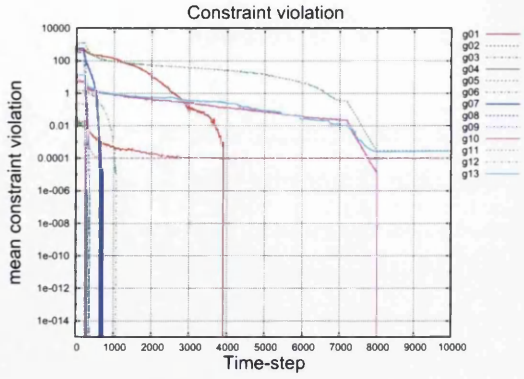
The total mean solution deviation as shown in fig.10.3c signifies the difference between each dimension from the optimum stated in CEC06 (solution coordinate error). Since more than one solution is possible for a number of problems (i.e. some variables may be interchangeable for the particular problem), then the meaning of these plots signifies whether a mean convergence in solution is apparent for each problem. It is clear from these plots that the mean gbest solution to problems g03, g07 and g09 has yet to converge in solution, however, problems g13 and g10 indicate that they have. It is hypothesised that g10 and g13 have their difficulty in satisfying the constraints, since mean convergence in solution by the gbest particle is apparent. Regarding g03, g07 and g09, it is again suggested here, as with the CEC05 (section 10.1), that swarm dynamic limitations of the particle swarm are likely to blame.

With regard to the equality constraints, since the PSO is not designed for hard equality constraints, it is not unexpected that its relaxation is not reduced to its final value ($1e-4$ for each constraint as defined by CEC06) until 80% of the search length (due to the difficulty met in attaining a 20% feasibility within the swarm). This is especially the case for g05 and g13, since these have multiple equality constraints, where g03 and g11 have only one. It is hypothesised, that due to the late removal of this relaxation of equality constraints (inability of the swarm to find feasibility), that the search may be concentrating too heavily on areas of the search space which are later infeasible. This again signifies the reasoning behind the use of a local optimiser, which above all else, quite happily deals with equality constraints. Unfortunately, equalities remain as inequalities, since equalities without relaxation would define a different problem entirely. Consistency and fairness for testing is paramount.

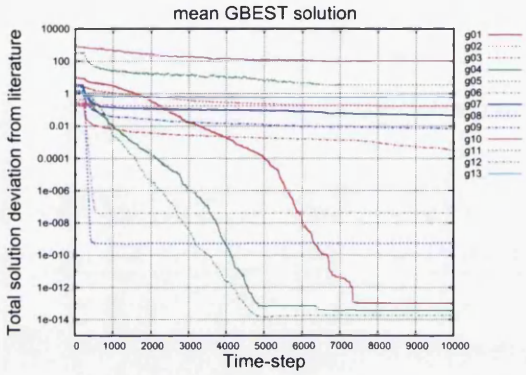
Considering the clustering and evolution measures for these functions, the interested reader is referred to appendix A.3.2. As already identified, problems g03, g07 and g09 fail to stagnate with respect to both their objective function value and solution coordinates, and as such, this is to be taken into account, where the use of measures signifying stagnation or success is made. Firstly, regarding the clustering measures (*cb_me*, *pb_me* and *pb_cge*), it is clear that relating problem success or stagnation of the algorithm to these measures is difficult since correlation across all problems is not apparent. An illustration of this difference is shown in fig. 10.4, with g03 indicating its rather high clustering measures indicating a slow convergence to the global optimum



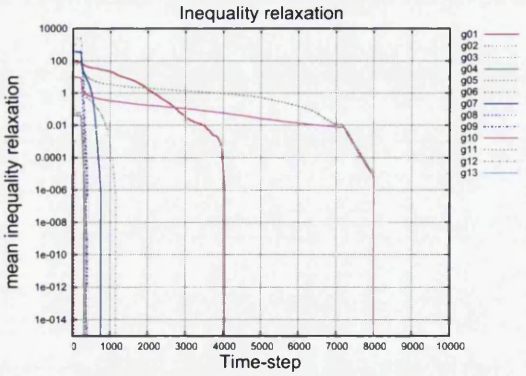

(a) Mean gbest conflict error



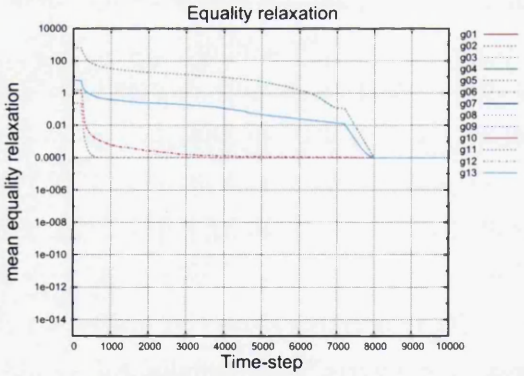

(b) Mean constraint violation




(c) Mean gbest solution error




(d) Mean inequality relaxation error

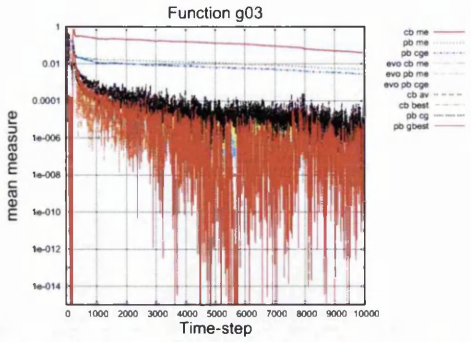



(e) Mean equality relaxation error

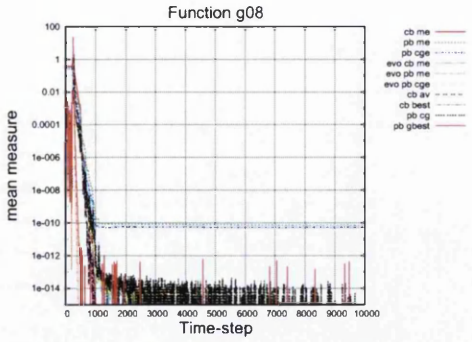
Figure 10.3: Constrained mean problem convergence

due to the large shallow basin apparent in this convex function. Also g08 is shown, which clusters very well within a small number of time steps, for the reason that the feasible area of the search space confines the swarm to a small area of the search space.

The most significant observation made on these measures, is problems g04, g05, g06




(a) Mean measures shown over 20sample runs for termination or switching (g03)




(b) Mean measures shown over 20sample runs for termination or switching (g08)

Figure 10.4: All observed swarm measures, indicating the level of stagnation for two illustrative well understood functions of the CEC06 suite.

and g08, which indicate very small clustering measures, signifying that the difference between the best and centre of gravity of the swarm to be very small in both position and conflict. This suggests a significant level of clustering. It does however indicate that the difficulty of a problem has a major impact on the ability of the swarm to converge to the global optimum solution in both position based and conflict based measures. g08 is a particularly interesting problem, since it is obviously multimodal (the reader is again referred to section 8.1.2 (fig. 8.9)), where g02 (as mentioned previously) has difficulty for this very reason. The obvious differences between these two problems other than the sheer number of dimensions apparent with g02, is the location of the global optimum, with its location restricting the direction in which the PSO swarm can approach it in the case of g02. The ease in which the PSO handles g08, is explained by the fact that exploration by the PSO is good enough to ensure that the valley in which the global optimum is contained is deemed more favourable than those around it. The local minima in g08 are not very deep and those which are, are not contained within the feasible search space, unlike g02, making the likelihood of other local optima pulling the swarm average away from the global optimum unlikely.

It should be noted that g09 and g12 have only a significantly low value in $cb\_me$ with regard to clustering measures, signifying that the swarm is very slowly becoming more densely clustered around gbest. This is likely since $pb\_me$ remains quite high. This is a function specific characteristic, making these measures rather unsuitable for cross-problem early switching. A more likely significant measure for switching, is evolution measures and also evolution of the clustering measures. This is likely to be

true, since these indicate a level of stagnation within the swarm as already discussed for the unconstrained problems. Again, these measures appear very function specific, relating not only to the number of dimensions but also to how the PSO has handled the constraints.

## 10.3 Summary

To summarise the findings of the above investigation of problem convergence, clustering measures have been shown to be clearly problem specific. Secondly, a number of weaknesses have been identified with application of the PSO to the suite: difficulty was encountered in attaining a higher than 20% feasibility ratio for a few of the problems; secondly, swarm dynamic limitations are apparent, limiting the algorithms ability to converge to a refined final solution. This is in fact where the local search is to step-in, where a gradient based optimiser is limited only by the accuracy of computer arithmetic.

# Chapter 11

# Particle swarm hybridisation with local search

This chapter comprises the hybridisation of the local search SQP with the GP-PSO (GP-PSO-SQP). Since unconstrained problems are initially studied (section 11.1), the apparent erratic measures of the GP-PSO (section 11.1.1) is investigated one one example unimodal and one multimodal problem. This is to determine ways in which to overcome such erratic results. Following this, a derivation of thresholds and initial testing is conducted in section 11.1.2 for the early switching of the GP-PSO to the SQP local search. This considers a subset of problems of the benchmark suite. Finally, section 11.1.3 conducts final tests on the derived measures for early termination / switch-over on the entire CEC05 suite.

Section 11.2 considers the constrained problem suite, where additional issues are highlighted and tackled as described in section 11.2.1, such as equality relaxation, conflict and constraint scaling and SQP termination criteria. Following this, like with the unconstrained suite, a stage of measure derivation and initial testing is made (section 11.2.2). This investigation up to now includes a subset of the suite (g01-g13). After which, final tests on the derived measures for early termination / switch-over is made (section 11.2.3) using the entirety of the CEC06 suite.

Since the two suites are designed as a benchmark of various algorithmic properties, some real engineering problems are considered for further testing as made in section 11.3. Following this, section 11.4 concludes on this investigation with respect to its effectiveness and impact compared to available algorithms in the literature.

## 11.1 Unconstrained problems

### 11.1.1 Erratic behaviour of swarm measures

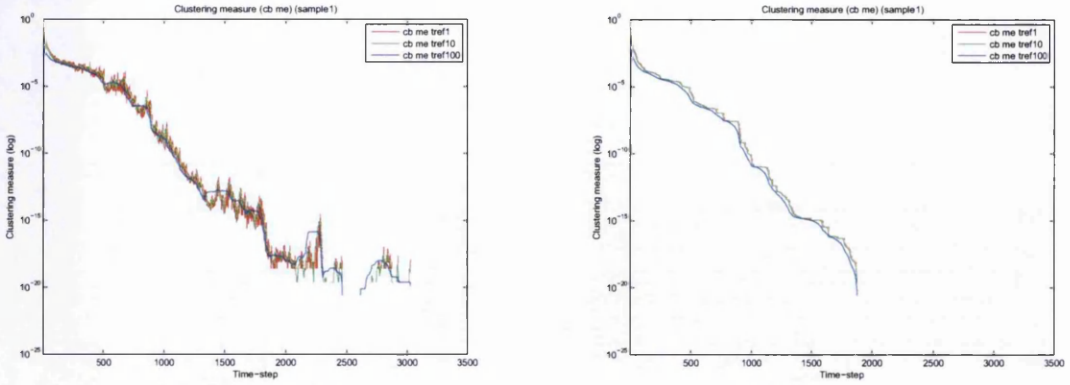
The erratic behaviour highlighted in the previous chapter, signifies that some method is required to extract meaning from the measures. It is within this section that various options are considered for the GP-PSO: One considers the calculation of the measures themselves through the use of either current population particle solutions, or alternatively, using the pbest solutions. The pbest solution may or may not update at every time-step (see section 11.1.1.1). Another method considers utilising a varying number of time-steps over which to average such measures, or alternatively, smoothing by counting the number of times by which a measure remains below a certain threshold (see section 11.1.1.2).

#### 11.1.1.1 Population measure

As described in chapter 10, the evolution and clustering measures are somewhat erratic. In order to address this, measures which use pbest populations are used in their calculation. A comparison is made here between the use of the two set-ups: a set-up consisting of a population of current solutions for measure calculation (popcur); and the previously mentioned population of personal best solutions for measure calculation (popbest). This is also analysed in parallel with a varying number of time-steps over which measures are averaged ('tref', see section 8.4).

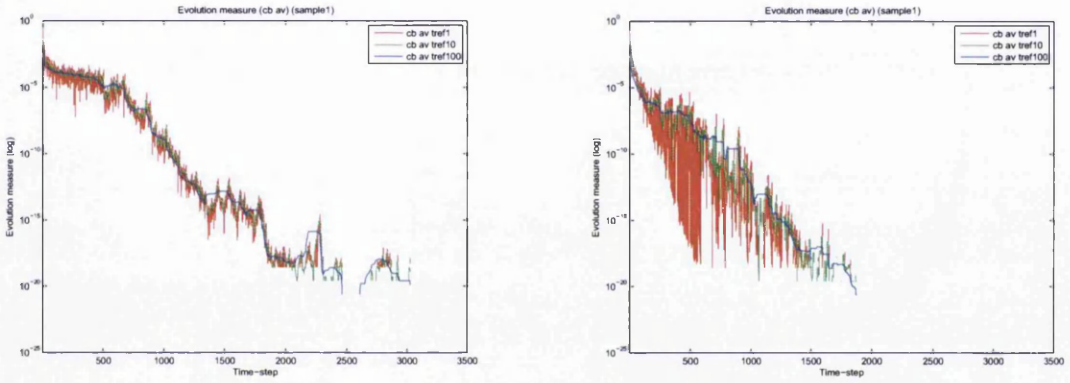
Two problems are investigated for this purpose: one unimodal F1(10D) function and one multimodal F9(10D). Single samples are analysed with these two set-ups. The first observation made, is that measures *cb_best* and *pb_gbest* show no difference between the use of popcur and pbest populations, since both measures utilise only information relating to the best particle within the swarm (gbest), which is common to both set-ups (and so common to all problems). Two illustrative pairs of plots are shown here: one, a representative clustering measure (fig. 11.1) and the other, a representative evolutionary measure (fig. 11.2). All other clustering measures follow a similar trend to fig. 11.1, as do all evolutionary measures to fig. 11.2.

Clustering measures show considerable smoothing with the use of the popbest set-up compared to the popcur set-up, and this is expected, since pbest updates are less frequent (i.e. updates to pbest members become less frequent as time progresses). These less frequent updates to the particles personal bests, result in the less frequent updat-




(a) current population (b) pbest particles

Figure 11.1: tref investigation for measure cb_me Function1 (10D)

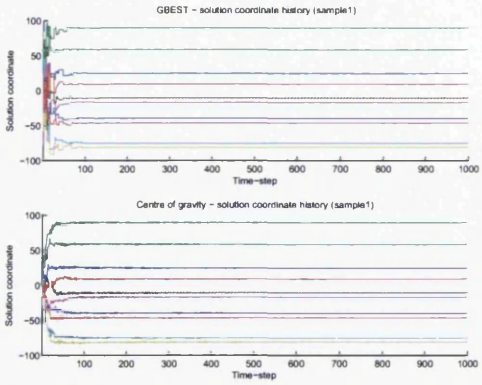



(a) current population (b) pbest particles

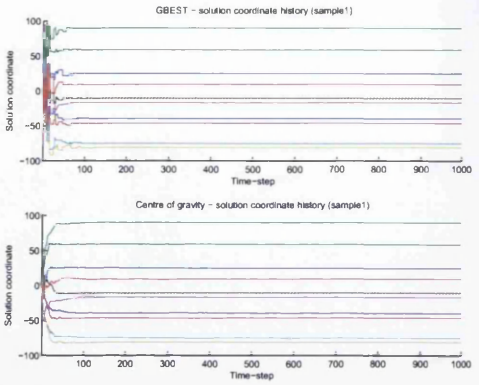
Figure 11.2: tref investigation for measure cb_av Function1 (10D)

ing within the popbest set-up. This less frequent update to pbest solutions have the undesirable effect of causing increasingly erratic curves based on evolution measures (fig. 11.2), since the centre of gravity suddenly changes as pbest members updates become less frequent. Once a pbest member finally finds an improved solution, the particle may have travelled quite some distance and be separated by a considerable number of time-steps. This is further indicated by the plots of both the solution quality (conflict) and solution coordinates (see figs. 11.3 and 11.4), where the decrease in frequency of updated solutions is apparent.

These observations on the pbest evolutionary measures strongly indicate a possibility to identify the statistical likelihood for further improvement, instead of the current state of the swarm for indication of stagnation directly with use of the popcur set-up.

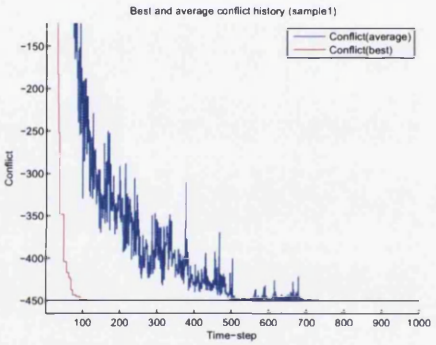



(a) Solution coordinates (current particles), with reduced range (1000 time-steps).

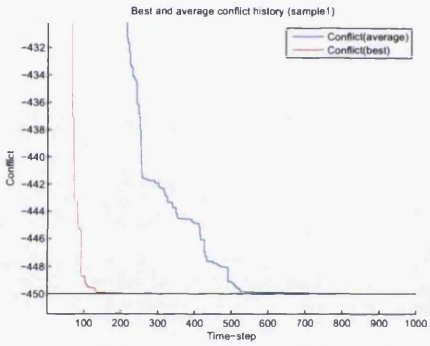



(b) Solution coordinates (pbest particles), with reduced range (1000 time-steps).

Figure 11.3: tref investigation, Solution coordinates for Function1 (10D)




(a) Conflict (current particles), with reduced range (1000 time-steps).




(b) Conflict (pbest particles), with reduced range (1000 time-steps).

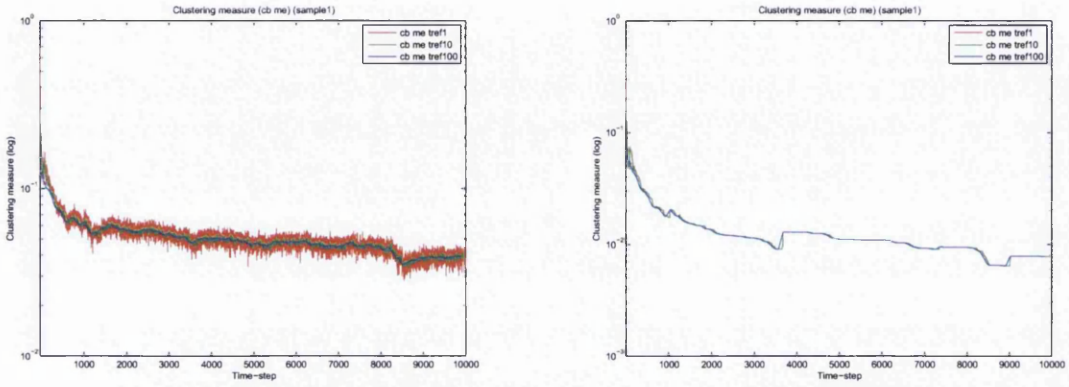
Figure 11.4: tref investigation, Conflict for Function1 (10D)

Regarding the level of smoothing (tref), an increasing tref clearly indicates an ever smoother curve for popcur and pbest set-ups. It is clear that pbest populations are highly suitable, especially in the early stages of the search where frequent updates are apparent.

To further investigate this behaviour, a highly difficult multi-modal problem called Rastrigin's function (F9(10D)) is investigated. Chosen as the attractiveness of many areas is likely to indicate the effect of tref smoothing together with changing frequency of pbest updates. Since some measures are based on the centre of gravity, where updates become very infrequent, and there may be multiple attractions at distant areas of the solution space, the evolution measures are expected to indicate ever more erratic

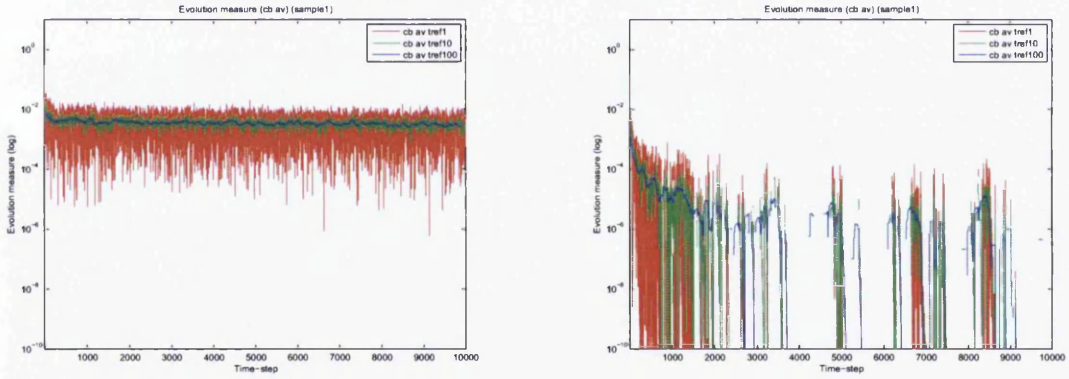
behaviour (more than in F1(10D)).

Again, only two of the total number of measures are plotted here, since they are representative of the general trend of their group (see figs. 11.5 and 11.6).




(a) current population (b) pbest particles

Figure 11.5: tref investigation for measure cb_me Function9 (10D)




(a) current population (b) pbest particles

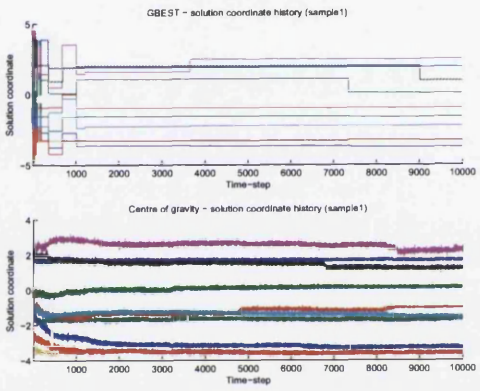
Figure 11.6: tref investigation for measure cb_av Function9 (10D)

Similar trends to problem F1(10D) are observed here with increasing tref, indicating significantly smoother curves. Unlike F1(10D), an ever larger tref is required for evolution measures to become smooth (and usable). With the popbest set-up, again no further smoothing is required for the clustering measures, though the erratic behaviour of the evolution measures increases substantially with the popbest set-up compared to the popcur set-up.

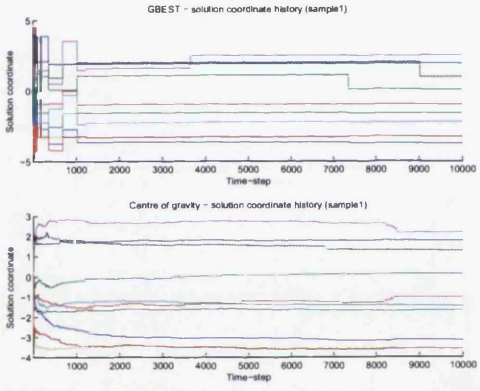
The popcur set-up is highly suitable, since real information about the current state of the swarm is retained. Since popbest does not necessarily relate to the current state, we

loose real-time meaning. However, the popbest method is computationally inexpensive compared to popcur, since updates are only made when members of popbest update (when a new pbest is found). Since popcur requires some level of smoothing as well as popbest, popbest is the chosen set-up with its reduced computational expense.

Frequent updates are apparent for time-steps of less than 1000, with jumps in solution coordinates being apparent due to the characteristics of this multi-modal problem (see figs. 11.5b, 11.6 and 11.7). pbest solutions jump from one local minimum to another. This likelihood reduces as the search progresses as shown in fig. 11.7. Following this, updates become highly sporadic, resulting in significantly erratic evolution measures for the popbest set-up, as shown in fig. 11.6. Conflict and constraint plots are not shown since they show a similar trend to F1(10D).




(a) Solution coordinates (current particles), with reduced range (1000 time-steps).




(b) Solution coordinates (pbest particles), with reduced range (1000 time-steps).

Figure 11.7: tref investigation, Solution coordinates for Function9 (10D)

Through this investigation, three points need to be addressed if a popbest set-up is to be utilised for successful implementation in the derivation of termination or switch-over measures:

- A way in which to encapsulate the statistical likelihood for significant change.
- A method in which to decide what is significant or not.
- Dealing with the issue of erratic evolution measures.

#### 11.1.1.2 Methods to overcome erratic evolution measures

There are two methods which are investigated for solving the problem of erratic behaviour in the measures: One method as discussed with Innocente [3] is called here the 'trefsum' method; the second method investigated here is the so-called 'credibility count' method. The 'trefsum' method works by varying tref for increased smoothing. In fact, it considers the maximum possible level of smoothing, where measures are averaged over all previous time-steps. The second method described here is the 'credibility count' method, which is where the number of time-steps that a particular measure remains below a certain threshold is counted. If this counter reaches a predefined number, together with all other measures considered, then this indicates a level of stagnation suitable for switch-over (illustrated in fig. 11.8).

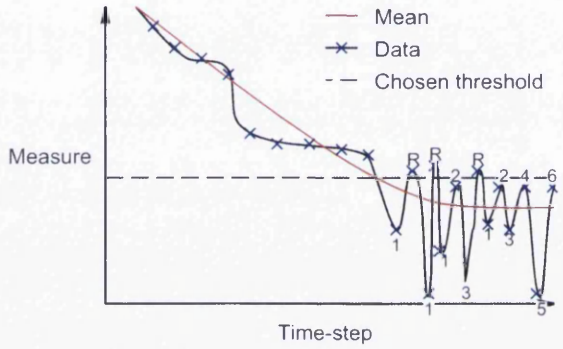
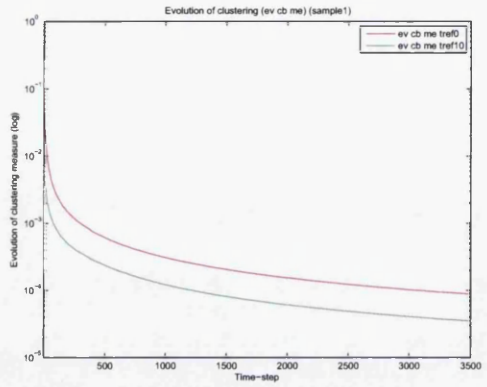



Figure 11.8: Credibility count method (credibility count is the number of time-steps below the threshold required to trigger early switching). $R$ represents the counter reset, $\#$ represents the counter increment.

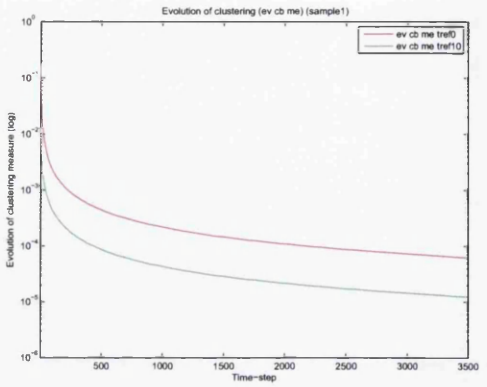
The trefum method is found to exhibit different magnitude criteria for each of the measures for different problems. This occurs as the trefsum method averages over the entire search history, where each problem may or may not have its own unique characteristics, resulting in a varying number of time-steps for measures to reach specific magnitudes. For this reason, this method offers to be an unsuitable method for early termination. However, it is perhaps a suitable method for a stagnation indication (i.e. a safe method to terminate a search if computational expense is not of too much concern).

To better illustrate this point, a set-up with varying number of ignored initial time-steps is tested, with results shown in fig. 11.9. The reasoning behind this investigation, is that the initialisation stage of the search is the most erratic of search stages. Quality of solutions generated in the initialisation of the swarm can substantially change, since it is random and totally unguided. Choosing to remove this period of the search

allows the effect on the averaging trefsum method to be understood further on in the search. The difference in magnitudes shown in fig. 11.9 between the different number of ignored time-steps clearly identifies the problem specific magnitudes. For this method, there is the obvious direct relation between the size of the population and the sample-sample initialisation variation of gbest. However, one significant advantage of the trefsum method is that the popcur set-up can be used, since with maximum smoothing, no erratic behaviour is observed.




(a) Problem1 10D (CEC05), using current particles (Sample1).




(b) Problem1 10D (CEC05), using pbest particles (Sample1).

Figure 11.9: tref investigation, Solution coordinates for Function9 (10D)

The second possible method considered is the credibility count method (CC). Offering much in the way of user control but at the expense of less than certainty of successful switch-over. For this reason, it is the preferred method for overcoming the erratic behaviour observed in the evolution measures.

### 11.1.2 Measure derivation and initial testing

Safe thresholds are required to be derived for use of the credibility count method. In order to extract clustering and evolutions measures from the run at the required time-step, the points which are likely to be suitable are first defined. The chosen points to extract thresholds consist of: the point at which the global optimum is found (within tolerance) by the GP-PSO by the gbest particle (cri_ERRB); where the global optimum is found by the average of the swarm (cri_ERRA); where it is found by the SQP when applied at each time-step of the GP-PSO (cri_SQP); and lastly, the extracted values at the final time-step (cri_END), assumed to be the extracted stagnation thresholds (since

a generous maximum time-step limit is given). Finding the global optimum is where a solution is within the tolerance of the global optimum solution as defined by the benchmark suite.

From this, the various criteria are chosen with a range $\{1, 10, 50, 100, 500\}$ on the *credibility count* (arbitrarily chosen range, where a count of 1 means the disabling of the counter method). It should be noted that individual problem results can be found in appendix A.4 (fig. A.3,A.4) with judgements on these criteria made based on table 11.1. Chosen extracted values are based on the minimum mean for the particular measure. The corresponding maximum value across the repeat runs for this minimum mean value is also presented as it is used for the switching thresholds. The chosen tested set-ups are shown in table 11.2, derived on selected problems of CEC05.

| | SQPBEST PROB | DIMENSION | MIN MEAN | MAX | END PROB | DIMENSION | MIN MEAN | MAX |
|---|---|---|---|---|---|---|---|---|
| cb_me | 6 | 10 | 9E-07 | 8E-06 | 1 | 30 | 0E+00 | 0E+00 |
| pb_me | 7 | 2 | 4E-03 | 8E-03 | 1 | 30 | 6E-11 | 1E-10 |
| pb_cge | 7 | 2 | 2E-03 | 5E-03 | 1 | 10 | 2E-10 | 3E-10 |
| evo_cb_me | 6 | 30 | 2E-08 | 3E-07 | 1 | 2 | 0E+00 | 0E+00 |
| evo_pb_me | 9 | 10 | 4E-06 | 3E-05 | 1 | 2 | 0E+00 | 0E+00 |
| evo_pb_cge | 7 | 30 | 3E-05 | 1E-04 | 1 | 2 | 0E+00 | 0E+00 |
| cb_av | 6 | 30 | 2E-08 | 3E-07 | 1 | 2 | 0E+00 | 0E+00 |
| cb_best | 6 | 30 | 9E-11 | 1E-09 | 1 | 2 | 0E+00 | 0E+00 |
| pb_cge | 9 | 10 | 2E-05 | 1E-04 | 1 | 2 | 0E+00 | 0E+00 |
| pb_gbest | 7 | 2 | 2E-05 | 5E-04 | 1 | 2 | 0E+00 | 0E+00 |
| | ERRB PROB | DIMENSION | MIN MEAN | MAX | ERRA PROB | DIMENSION | MIN MEAN | MAX |
| cb_me | 6 | 10 | 3E-11 | 2E-10 | 6 | 2 | 5E-14 | 9E-14 |
| pb_me | 1 | 10 | 9E-04 | 2E-03 | 1 | 30 | 1E-07 | 5E-07 |
| pb_cge | 1 | 2 | 3E-04 | 5E-04 | 1 | 30 | 1E-07 | 1E-07 |
| evo_cb_me | 6 | 10 | 2E-16 | 2E-15 | 6 | 2 | 7E-16 | 5E-15 |
| evo_pb_me | 6 | 30 | 1E-07 | 2E-07 | 1 | 30 | 9E-09 | 5E-08 |
| evo_pb_cge | 6 | 30 | 1E-07 | 2E-07 | 1 | 30 | 2E-09 | 8E-09 |
| cb_av | 6 | 10 | 2E-16 | 2E-15 | 6 | 2 | 7E-16 | 5E-15 |
| cb_best | 6 | 30 | 3E-18 | 4E-18 | 1 | 2 | 0E+00 | 0E+00 |
| pb_cge | 6 | 10 | 2E-07 | 5E-07 | 1 | 30 | 2E-08 | 6E-08 |
| pb_gbest | 1 | 30 | 1E-07 | 3E-07 | 1 | 2 | 0E+00 | 0E+00 |

Table 11.1: Measures extracted from the results of the CEC05 suite for early switching criteria.

| Measure | Clustering | | | Evo-clustering | | | Evolution | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Criteria label | cb me | pb me | pb cge | evo cb me | evo pb me | evo pb cge | cb av | cb best | pb cge | pb gbest |
| *SQP1 | 1e-6 | 1e-3 | 1e-3 | 1e-7 | 1e-5 | 1e-4 | 1e-7 | 1e-9 | 1e-4 | 1e-4 |
| *SQP2 | - | - | - | 1e-7 | 1e-5 | 1e-4 | 1e-7 | 1e-9 | 1e-4 | 1e-4 |
| ERRB | - | - | - | 1e-15 | 1e-7 | 1e-7 | 1e-15 | 1e-18 | 1e-7 | 1e-7 |
| *ERRA | - | - | - | 1e-15 | 1e-8 | 1e-9 | 1e-15 | 0e0 | 0e-8 | 0e0 |
| END1 | - | - | - | 0e0 | 0e0 | 0e0 | 0e0 | 0e0 | 0e+0 | 0e0 |
| END2 | - | - | - | - | - | - | - | 0e+0 | - | 0e+0 |

Table 11.2: The six chosen criteria for switch-over. *chosen thresholds for further testing.

Of those set-ups shown in table 11.2, three are further tested as indicated by the asterisk. These three are chosen from some initial testing on problems 1,6,7,9 and 12 (2D,10D and 30D) based on three extremes: one is the safest criteria (SQP1) (takes into account the clustering measures meaning that only a limited number of problems

will trigger such a measure); secondly, the cri_SQP2 is chosen as a measure in which to apply across a wider range of problems; finally cri_ERRA is chosen as it fulfils the same criteria as cri_SQP2 only that it is a slightly safer option, indicating that further testing is required.

cri_SQP1 criteria offers to be a 100% successful and safe criteria and utilises clustering measures. This makes it problem specific, but at the advantage of its insensitivity to credibility count due to the smoothness of the clustering measures based on the popbest set-up. The lack of generality of this criteria is observed by its low triggering count of 42%, as it simply does not trigger in problems that do not cluster to a high enough degree.

cri_SQP2 and cri_ERRA criteria however, offer to give user control over computational expense and accuracy, they do so to a varying and unpredictable degree due to the problem specific characteristics that are apparent.

cri_SQP2 required between only $4 - 39\%$ the number of FEs to trigger (over the corresponding CC range), compared to the point at which the PSO would be deemed successful[1] and between $2 - 16\%$ compared to the total number imposed by the final time-step (10000). With respect to accuracy, between $72 - 87\%$ is observed compared to final time-step switch-over, over the credibility count of $1 - 500$.

To clarify, this corresponds to the mean accuracy across problems F1, F6, F7, F9 & F12 (2D,10D and 30D). That is, the mean percentage accuracy compared to the known global optimum solutions for each particular problem. For the cri_ERRA criteria, similar accuracy was observed with those of the previous analysis, only a range of $30 - 98\%$ FEs was required on average for triggering to occur compared to the point at which the PSO found the global optimum over the CC range.

Since an attempt for a successful switch-over to the local search is to be made regardless of whether it occurs at the moment in which the local search would be successful, it is good enough to trigger a switch-over at the point at where success is likely to have been attained by the PSO, on average. That is, the results indicate a performance that relates to a stagnation measure more than an early switching measure. Problems of the CEC05 suite are highly difficult and clearly do not stagnate to a high enough degree for problems of 10D or more. This then results in the lack of triggering of the criteria derived.

[1] success measured by the definition of the benchmark

### 11.1.3 Testing of the chosen measures

Further testing of these three criteria on all problems 1-14 in 2,10 and 30 dimensions is made now with a dynamic neighbourhood (The results of which can be found in appendix A.5 (table A.6). These results clearly reaffirm the lack of accurate triggering with anything less than 500 time steps on those problems which offer difficulty to the PSO. Regardless of these observations, a fair comparison is made with those available in the literature. For this, only the first 14 problems of the suite are considered and of these, only the 10 dimensional cases.

It should be noted that since a local search is to be triggered at the final time-step, if no criteria has been met, any search length may be used. However, depending on the chosen credibility count, it may be unlikely that a criteria will be met before the search terminates if the search is too short. Similarly, in the case of the first criteria cri_SQP1, the high clustering may not occur with very tight restrictions on the search length.

For suitable comparison, both TRIBES and DMS-L-PSO are chosen as already described (section 8.3). The results are summarised as follows in table 11.3:

| (10D) | Acceptable error | FE TRIBES | success | FE DMS-L-PSO | success | FE GP-PSO | success | mean error | FINAL TIME-STEP FE GP-PSO-SQP | success | mean error | COUNT 100 FE GP-PSO-SQP | success | mean error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | 1e-6 | 1.36E+003 | 100% | 1.19E+004 | 100% | 4.33E+04 | 100% | 0,00E+00 | 1.0E+05 | 100% | 0.00E+00 | 8.86E+003 | 100% | 0.00E+00 |
| F2 | 1e-6 | 6.62E+003 | 100% | 1.23E+004 | 100% | 9.9E+04 | 20% | 1.82E-04 | 1.0E+05 | 100% | 0.00E+00 | 9.88E+003 | 100% | 0.00E+00 |
| F3 | 1e-6 | 1.04E+004 | 100% | 1.25E+004 | 100% | 1.0E+05 | 0% | 2.13E+05 | 1.0E+05 | 100% | 0.00E+00 | 1.07E+004 | 100% | 0.00E+00 |
| F4 | 1e-6 | 1.74E+004 | 100% | 1.00E+005 | 0% | 1.0E+05 | 0% | 5.99E-02 | 1.0E+05 | 0% | 6.17E-02 | 9.65E+004 | 0% | 7.23E-02 |
| F5 | 1e-6 | 5.54E+004 | 88% | 1.00E+005 | 80% | 9.8E+04 | 24% | 9.23E-04 | 1.0E+05 | 28% | 5.88E-04 | 1.53E+004 | 0% | 1.35E+00 |
| F6 | 1e-6 | 9.83E+004 | 4% | 5.47E+004 | 100% | 1.0E+05 | 0% | 3.53E+00 | 1.0E+05 | 88% | 4.78E-01 | 1.16E+004 | 76% | 9.57E-01 |
| F7 | 1e-2 | 9.66E+004 | 4% | 1.00E+005 | 16% | 1.0E+05 | 0% | 3.07E-01 | 1.0E+05 | 0% | 1.39E-01 | 9.88E+003 | 0% | 5.94E+00 |
| F8 | 1e-2 | 1.00E+005 | 0% | 1.00E+005 | 0% | 1.0E+05 | 0% | 2.02E+01 | 1.0E+05 | 0% | 2.00E+01 | 4.72E+004 | 0% | 2.00E+01 |
| F9 | 1e-2 | 1.00E+005 | 0% | 3.46E+004 | 100% | 1.0E+05 | 0% | 7.65E+00 | 1.0E+05 | 0% | 7.21E+00 | 1.48E+004 | 0% | 5.01E+01 |
| F10 | 1e-2 | 1.00E+005 | 0% | 1.00E+005 | 0% | 1.0E+05 | 0% | 1.61E+01 | 1.0E+05 | 0% | 1.09E+01 | 1.42E+004 | 0% | 7.22E+01 |
| F11 | 1e-2 | 1.00E+005 | 0% | 1.00E+005 | 0% | 1.0E+05 | 0% | 4.23E+00 | 1.0E+05 | 0% | 4.17E+00 | 6.69E+004 | 0% | 6.58E+00 |
| F12 | 1e-2 | | | 1.25E+004 | 76% | 1.0E+05 | 0% | 5.77E+01 | 1.0E+05 | 48% | 6.70E+00 | 1.32E+004 | 12% | 4.28E+03 |
| F13 | 1e-2 | | | 1.00E+005 | 0% | 1.0E+05 | 0% | 1.22E+00 | 1.0E+05 | 0% | 7.28E-01 | 1.23E+004 | 0% | 1.38E+01 |
| F14 | 1e-2 | | | 1.00E+005 | 0% | 1.0E+05 | 0% | 3.01E+00 | 1.0E+05 | 0% | 2.75E+00 | 2.09E+004 | 0% | 4.07E+00 |
| | | 6.24E+4 | | 6.70E+4 | | 9.58E+4 | | 1.01E+5 | | 2.52E+4 | | | | |
| | | 62.00% | | 66.63% | | 95.22% | | 100.00% | | 25.01% | | | | |

Table 11.3: Comparison with the literature between the GP-PSO, GP-PSO-SQP and two leading pso algorithms.

With the added restriction of CEC05, with a maximum number of FE of $1e5$, using a credibility count of 500, no triggers were made on the problems considered and for this reason they are not presented here. Measures based on a credibility count of 1 are very unreliable (non general) and have also not been included here (apart from cri_SQP1 which did not trigger at all on these problems within this range).

Results shown in table 11.3 appear to be positive with regards to early switch-over of the GP-PSO to the local search. However, there is no generality across problems using these criteria. On a positive note, considerable savings are made with use of the 100 count cri_SQP1 method together with the fact that better success rates are evident across this range of problems. However, there is still no success on F5 and reduced success on F6 compared to the high success rates of the PSO, if allowed to continue its search with final solution refinement. Since the tolerance of what is described as a successful result or not is rather small, the mean conflict is shown in order to illustrate that termination is possible with the criteria shown in table 11.3. This is dependent on the necessary accuracy defined by the user or perhaps with multiple restarting points for the local search around this solution. This deduction is made, since the mean conflict is near the global optimum, and in fact, analysing the individual samples indicates that the majority of these samples are to within $1e-4$ of the global optimum for F1-F6 and to within $1e+2$ of the global optimum for F7-F14. Those problems that fail at the final time-step, are just outside the range of being deemed successful by the suite.

With use of a credibility count of 100, there is not a significant reduction in accuracy observed over the problems investigated, as indicated by the mean error, and a significant reduction in FE is observed (on average, significantly less than DMS-L-PSO). This indicates that the current set-up of GP-PSO is in fact perhaps a 'greedier' algorithm. The GP-PSO meets success on F4 if given enough time, together with F9 and F12, which then brings it in line with DMS-L-PSO. An interesting difference is that the DMS-L-PSO attains a successful solution (to within acceptable tolerance of the global optimum as defined by the suite) on F7, which the GP-PSO-SQP does not, though the mean error is within $1e-1$ ($1e-2$ is acceptable), even if not restricted by the CEC05 limits.

Comparison with TRIBES indicates that the search again appears to be more efficient than with the GP-PSO-SQP, although with it achieving rather low success rates on F6, for which the GP-PSO-SQP does not share. From the paper by Cooren et al. [57], it is apparent that this is due to the trapping within a local optimum. It is then speculated that F7 then offers the GP-PSO and indeed GP-PSO-SQP much difficulty for this very

reason.

Comparing the three algorithms, it appears that the GP-PSO is the weaker of the three search algorithms, however this comes as no surprise with its most recent developments targeting constraint handling. It should be noted however, that this serves as an indication only, since more recent versions of the algorithm are far superior than the test bed algorithm used here. It should also be re-emphasised that results for comparisons with the CEC05, are made with an early termination in mind, corresponding to the FE limit imposed by the CEC05 suite.

It is concluded from the results presented, that the introduction of the local search enables the GP-PSO to refine its solution if the GP-PSO has successfully searched the solution space but perhaps not to the required accuracies as defined by the suite. Early termination is perhaps not suitable for such problems, but instead a termination based on stagnation only, where the search will terminate with no likelihood for further improvement by the PSO if it were allowed to continue.

Since these measures offer to be a less than a sufficient method for early switching in terms of the defined success of the CEC05 suite, constrained problems are now considered.

## 11.2 Constrained problems

The constrained version of the algorithm utilises a relaxation of tolerances which progressively restricts the feasible area of the search space. This relaxation of tolerances is then based on the feasibility of swarm members. For this reason, the constrained application of the PSO is likely to demonstrate much more favourable swarm dynamic measures which can be related between problems, since the algorithm itself imposes an ever more restrictive search as it progresses.

### 11.2.1 Additional remarks on implementation

Since constrained problems offer significant differences to the unconstrained problems encountered previously, these differences must be addressed and their significance in the hybridisation of the two techniques be considered. One such consideration is of the equality relaxation defined by the suite (see section 11.2.1.1), which defines a different problem entirely to a formulation without any relaxation. Secondly, a basic considera-

tion is made toward the scaling of the problems with the investigation of liner scaling of both conflict and constraint function as achieved in section 11.2.1.2. Finally some consideration is made with regards to the tolerances defined for the SQP algorithm (section 11.2.1.3).

#### 11.2.1.1 Equality relaxation for the SQP:

It should be noted, that for consistency with the requirements of the suite, problem equalities are reformulated as inequalities for application of the SQP local search as with the GP-PSO. In CEC06 [51], the definition of a feasible region on equality constraints ($\eta = 1e-4$) is subject to the constraint having been already relaxed. However, in this investigation, the equality constraints are relaxed to $\epsilon$ and then treated as strict inequality constraints with a tolerance of only $1e-12$ for the SQP (estimated rounding-off errors). No tolerance is given to the GP-PSO since it is not limited by the calculation of the gradient. An illustration of this relaxation is shown in fig. 8.8. In this figure, it is shown how two equality constraints are relaxed, resulting in an area of feasibility, rather than a single feasible point in the search space in the case of strict equality formulation. It has been observed with numerous functions that the SQP path follows along the outer edge of relaxed feasibility for the very reason of this relaxation of its equalities.

Re-formulation of equality constraints as inequalities, has already been suspected of causing problems in the local search implementation since it seems reasonable to assume that regions normally separated by infeasibility may become connected. It is also possible that regions of sub-optimality may be created that would not exist in the case of strict equality formulation. Equality relaxation is the necessary method for the implementation of EAs to equality constrained problems and so this was the intended application of this suite. The local search method cannot utilise the statistical advantages of a diverse swarm and so this matter of equality relaxation is an important consideration.

An investigation into the strict equality formulation of problems g01-g13 was also made, with the SQP converging with much greater speed (reduced number of steps required to reach the global optimum). However, this solution no longer corresponds to the same global optimum defined by the suite, since the formulation without equality relaxation results in the definition of a different problem being tackled.

#### 11.2.1.2 Scaling for the SQP

The scaling of a problem may be considered for the case of each functions variables, where one variable maybe more sensitive than another. Another type of scaling is between the magnitudes of the functions themselves. The later is considered here, where the constraints are normalised to bring their feasible values to within the range 0-100. The conflict (objective) function can also be similarly scaled, however, since its lower bound for a general problem is not known, then its range of values (over all space) is normalised between 1-100.

Scaling of the function is considered, to identify as whether the suite is sensitive or not to it. It should be noted that scaling is made using the search history of the GP-PSO over the infeasible search-space, since constraints are not considered when they become feasible.

Scaling the conflict and constraint functions is achieved by eq. (11.1);

$$||A|| = \frac{(A - D2)}{(D1 - D2)} \times (S1 - S2) + S2 \tag{11.1}$$

where $A$ is the non-normalised value, $||A||$ is the normalised value, $D1$ and $D2$ are the maxima and minima of the function respectively and finally, $S1$ and $S2$ are the chosen scaling (upper and lower scaling respectively). It is important to note that this method of scaling is linear and as such is not necessarily the best method to scale non-linear functions. The reasoning behind this statement is that problems may change sensitivity with respect to one another as a function of their location in the search-space. In such a circumstance, one might expect linear scaling to be sometimes beneficial but at other times harmful, depending on the starting location within the search-space. The extremities of the functions are determined by the GP-PSO search. g07 is successful from the very first time-step when linear scaling is applied but without it, takes a considerable number of time-steps to achieve successful convergence (if successful convergence even occurs). The linear scaling of functions has shown that a number of functions are sensitive to it, but are just as likely to be harmful as they are beneficial to it, as explained above. g07 does however appear to indicate a significant benefit to linear scaling. In conclusion, linearly scaling a function results in unpredictable performance and so it is not pursued in the application of the GP-PSO-SQP to this particular suite.

#### 11.2.1.3 SQP termination criteria (tolerances)

The tolerances set for the SQP algorithm are based on some initial runs, which concluded that a tolerance of $1e-14$ on both the conflict and constraint and $1e-12$ for the difference in solution coordinate to be suitable. These resulted in the successful termination measures of the algorithm. These tolerances not only define the point at which the SQP terminates, but also the point at which an area of the solution space becomes feasible.

Looking at the points at which the SQP fails, reveals that when the tolerances are reduced for the constraint and conflict to $1e-12$, considerable improvement is observed in functions g05, g07 and g10. The observed 'singular matrix error' given by the SQP algorithm, has its cause in the computation of the inverted matrix (required in the calculation of the Hessian) beyond that of double precision (i.e. a divide by zero will occur with the function appearing 'flat' within computer accuracy). This property is once again inherently linked with scaling, since it is observed that g07 shows considerable improvement when scaled but also when reduced tolerances to the SQP termination criteria are used. With no relaxation or scaling on this function, an overshooting of the optimum is observed as shown in fig. 11.10. This figure clearly shows the convergence in solution to the globally known optimum, but then a continuation is observed past this point, as termination measures of the algorithm have yet to be triggered.

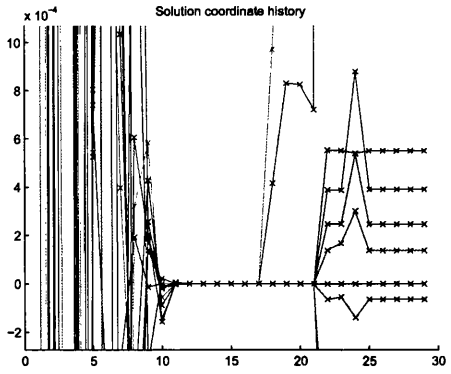



Figure 11.10: Overshooting due to chosen tolerances in g07 with normalised solution coordinates shown.

Function g01 shows improvement with the tighter tolerances of $1e-14$. Function g10 results in an unsuccessful convergence and on occasion, an overshooting when tolerances are reduced and so this effect it rather unpredictable. The problem in fact derives from the magnitude specific tolerances defined within the SQP, resulting in an

unpredictable overshooting behaviour as the search continues past computational accuracy. The other extreme, is where the search terminates before convergence to the global optimum is made (according to the predefined tolerance requirements of CEC06). This results from the fact that problem-specific (magnitude-specific) round-off is apparent.

Overall, a relaxed conflict and constraint tolerance is shown to achieve better results, but as a result of this investigation, a solution recovery method is used to ensure that overshooting does not occur. A pseudo-code is shown in appendix A (alg. 16). Since the SQP algorithm is used as a single-use mechanism, the extra computation and memory requirements in memory recovery is considered negligible.

### 11.2.2 Measure derivation and initial testing

Another problem arises in the derivation of the thresholds. The pseudo-adaptive relaxation of tolerances based on the feasibility of the swarm means that measures do not relate to the final problem being tackled until either 80% of the search has been reached or the swarm has successfully remained more than or equal to 20% feasible (so that final constraint tolerance is set). Due to this relaxation of tolerances based on the feasibility of the swarm, the following pseudo-code is used to illustrate the application of the SQP local search to the GP-PSO with respect to chosen criteria (see appendix A alg. 13). In summary, in addition to the requirement for each criteria to be met, final relaxation values of the constraints must be achieved before the GP-PSO is allowed to trigger solution refinement with SQP.

Thresholds are defined according to the points at which the PSO attains success as deemed by the suite (cri_ERRB & cri_ERRA) and also the final time-step extraction values (cri_END), rather than the point at which the SQP attains success (with the additional requirement that final tolerance be reached). These are then derived based on a sample number of problems of the CEC06 suite (problems g01-g13) as with the CEC05 suite. The extracted thresholds for these criteria are shown in table 11.4. The corresponding chosen thresholds for further testing are then shown in table 11.5.

Initial investigations into constrained problems, reveal there to be a considerable difference with the unconstrained suite investigated in section 11.1 (p.98). This results from the added requirement for final problem tolerance being reached. Indications are, that this is the main drive for successful switching, rather than the chosen thresholds or perhaps their credibility count. This is indicated by a lack of sensitivity in the credibility count on constrained problems and the insensitivity to derived thresholds.

| | END | | | ERRB | | | ERRA | | |
|---|---|---|---|---|---|---|---|---|---|
| | min mean | problem | max | min mean | problem | max | min mean | problem | max |
| cb_me | 0.0E+00 | 12 | 0.0E+00 | 1.4E-06 | 13 | 5.4E-06 | 1.3E-08 | 6 | 1.7E-08 |
| pb_me | 7.5E-11 | 8 | 1.3E-10 | 7.1E-06 | 13 | 2.7E-05 | 2.6E-09 | 6 | 4.4E-09 |
| pb_cge | 5.3E-11 | 8 | 1.1E-10 | 4.4E-06 | 13 | 2.7E-05 | 5.5E-10 | 6 | 7.2E-10 |
| cb_av | 0.0E+00 | 8 | 0.0E+00 | 2.5E-08 | 13 | 1.0E-07 | 9.8E-12 | 6 | 7.4E-11 |
| cb_best | 0.0E+00 | 1 | 0.0E+00 | 0.0E+00 | 2 | 0.0E+00 | 0.0E+00 | 6 | 0.0E+00 |
| pb_cg | 0.0E+00 | 8 | 0.0E+00 | 5.9E-08 | 13 | 1.6E-07 | 4.0E-13 | 6 | 3.1E-12 |
| pb_gbest | 0.0E+00 | 1 | 0.0E+00 | 0.0E+00 | 2 | 0.0E+00 | 0.0E+00 | 6 | 0.0E+00 |
| ev_cb_me | 0.0E+00 | 8 | 0.0E+00 | 2.4E-08 | 13 | 1.0E-07 | 9.8E-12 | 6 | 7.4E-11 |
| ev_pb_me | 0.0E+00 | 8 | 0.0E+00 | 6.0E-08 | 13 | 6.0E-07 | 3.1E-13 | 6 | 3.4E-12 |
| ev_pb_cge | 0.0E+00 | 8 | 0.0E+00 | 1.0E-07 | 6 | 8.1E-07 | 4.0E-13 | 6 | 3.1E-12 |
| excluded | none | | | 5,7,9,10 | | | Remainder | | |
| included | all | | | Remainder | | | 6,8,11,12,13 | | |

Table 11.4: The derived measures, where the minimum mean signifies the minimum mean magnitude for the derived category across the investigated problems and 'max', the maximum magnitude on the problem with minimum mean.

| Measure | Clustering | | | Evolution-clustering | | | Evolution | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Criteria label | cb me | pb me | pb cge | evo cb me | evo pb me | evo pb cge | cb av | cb best | pb cge | pb gbest |
| END1 | 0e0 | 1e-10 | 1e-10 | 0e0 | 0e0 | 0e0 | 0e0 | 0e0 | 0e+0 | 0e0 |
| END2 | 0e0 | 1e-10 | 1e-10 | - | - | - | - | - | - | - |
| *END3 | - | - | - | 0e0 | 0e0 | 0e0 | 0e0 | 0e0 | 0e+0 | 0e0 |
| *ERRB | - | - | - | 1e-7 | 0e0 | 1e-7 | 0e0 | 1e-7 | 1e-7 | 1e-7 |
| ERRA | - | - | - | 1e-11 | 0e0 | 1e-12 | 0e0 | 1e-11 | 1e-12 | 1e-12 |

Table 11.5: Chosen criteria for switch-over. *chosen thresholds for further testing.

The full set of results for this initial testing stage can be found in appendix A.4 (table A.5). Unlike with the unconstrained problems, the main sensitivity of accuracy appears to be between a credibility count of 10 and 50, according to the error based comparison (err_def2). err_def2 signies the average fractional difference in error relative to the result at the nal time-step, expressed as a percentage (further details as to the definition of this error can be found in appendix A.4). That is, a credibility count of 50 appears to be adequate for almost all problems. One problem that does not follow this trend, and effects the statistics of these results, is problem g02. This problem is multi-modal and as described in an earlier section, offers significant difficulty to the SQP, since a very good starting solution in proximity to the global optimum is required and the only way to ensure maximum chance of this occurring, is to provide the SQP with a solution as late as possible in the PSO search. However, with a credibility count of 500, the solution improves to that of the end time-step value (this is shown by a 100% accuracy relative to both the end time-step value but also the point at which the PSO attains success).

Unlike the unconstrained problems, it appears there might be the potential to control accuracy and computational expense with use of the credibility count, as the feature which most determines a successful switch, is to whether final tolerances are met. At this point the search is likely to have reached a point at which particles have already

heavily clustered around their final solution. As already discussed and confirmed by results shown in appendix A.4, clustering measures do not relate between any tested problems (criteria utilising clustering measures fail to trigger), as none cluster to a high enough degree. A similar observation was made on the unconstrained suite. The dynamics of the swarm are effected to an even greater degree than in the unconstrained suite with the added effect of constraints, creating dramatic differences between measures of different problems. This was observed in section 10.2 (p.91).

The mean accuracy across all problems on 20 sample runs, offers to be an indication only of the accuracy of the algorithm. That is, 'err_def2' is defined in such a way that the level of accuracy attained by different problems may result in a significant difference of error scale with respect to one another, making the mean accuracy little or no use in meaningful terms. Since each problems scale and accuracy of solutions found, appear significantly different, another means of comparison is used, which identifies the error with respect to the conflict, rather than the absolute error (err_def1). From this measure, it is clear that results are rather good on all credibility counts except for problem g02, as already identified. It is conjectural whether a credibility count may be necessary in cases where extreme multi-modality are present, and perhaps no count is required on those problems where it is not. In a multi-modal problem, sudden increases at different points in the search history, result from gbest falling into better suboptimal locations (as these are numerous in problem g02). Even though g08 is a multi-modal problem, the constraints appear beneficial in addition to the low dimensionality of the problem and reduces the likelihood for such problems to occur.

In addition to g02, g04 and g01 also both show a mean increase in error when triggered, without a suitably high credibility count (a CC of less than 10 in the case of g04 and a CC of less than 50 for g01). Upon further investigation, it is observed that this result is in fact very close to the global optimum with non-zero gradient and furthermore, it is within feasible space (zero constraint violation). It should be noted that only 1 sample of the 20 on g04 resulted in a solution with higher than acceptable fixed level accuracy (4.1e-4). The solution found by the SQP on this particular run remains unchanged from that found by the PSO, signifying a converged locally optimal solution. Regarding problem g03 and other functions demonstrating fluctuating error according to credibility count, from analysing the raw results, it is indicated that this error is due to round-off and can be discounted from being an issue (i.e. the error is very small and as such, the percentage between the mean error becomes large, thus err_def2 has been removed from the table of results).

If the SQP algorithm is supplied with an already optimised starting solution, then it is observed that the number of iterations required by the SQP to converge is small, resulting in perhaps minimal change to the initial solution. To further clarify, the further on in the search a trigger occurs, the more likely that no change in solution will result and so the solution remains fixed and no difference observed in round-off.

It is clear by these detailed comparisons of the individual results, that only g02 offers difficulty to the application of early termination, however, the causes have been identified. It is suggested, that in the case of highly difficult unpredictable multi-modal problems, the local search be applied at the very final time-step, or a suitably high credibility count be set. From the observations made above, a credibility count of 1 (disabled) is adequate for the purpose of early switch-over, if accuracies required are limited to those required by CEC06 ($1e-4$). However, to increase the mean accuracy and obtain results that represent the final time-step values, a credibility count of 50 appears to be required in the general case, though in the vast majority of cases, similar quality of solutions are observed at a credibility count of 1.

Criteria cri_END1 and cri_END2 have been removed as they are based on the clustering measures. Since all three measures (cri_END3, cri_ERRB and cri_ERRA) result in similar accuracies across the chosen problems, computational expense is used as the main means in which to choose and further test with the entire suite of CEC06. Again the computational saving of each of the chosen criteria are remarkably similar, resulting from the insensitivity of the CC and the dominance of the swarm feasibility on the termination of the PSO search. Since this is the case, the marginally less computationally expensive cri_ERRB criteria is chosen and further tested. With regards to cri_ERRB, only 63% of the required number of FEs were required compared to the point at which the GP-PSO attains error and only 24% compared to the limits of the run, with a credibility count of 1.

### 11.2.3 Testing of the chosen measures

For testing on the entire cec06 suite, both cri_END3 and cri_ERRB criteria are chosen to check the insensitivity of the suite to different thresholds. Again, the range of 1-500 for the credibility count is tested, to investigate the (in)sensitivity of the suite to the credibility count with respect to uni/multi-modal problems. Firstly, a few issues are highlighted when dealing with the entire CEC06 test suite. Concerning g14, some starting solutions provided by the GP-PSO allowed the SQP algorithm to venture outside

the feasible search-space, where a less than or equal to 0 solution coordinate results in a logarithm of zero which is no longer real. A safety measure is written into the conflict and constraint functions for the SQP, which is used on this function alone, to force a solution coordinate that becomes less than or equal to zero to be set to $1e-12$. This procedure is made, as prior knowledge of the problem in terms of its feasible bounds determines that no solution coordinate should be allowed to cross this boundary. However, problems g21 and g22 also contain logarithms, defined such that some of the constraints should be *hard*[2]. Both PSO and SQP drift into infeasibility, due to attractive areas of the search space. This is beyond the investigation of switching criteria, as the ability to tackle hard constraints within either algorithm is yet to be made. A summary table of results is shown in appendix A.5 (table A.7).

As with the previous section, the results have again shown to be somewhat insensitive to credibility count and indeed the threshold, with the latter indicated by the similarity of the two criteria. Measures below a certain threshold represent an average over a certain number of time-steps upon which no change is observed by pbest particles of the swarm. To further clarify, a value for evolution measure close but not equal to zero results from the averaging of 10 time-steps (which under this investigation is a constant). The most suitable choice as already discussed in section 11.2.2, is the case where no change is apparent over a certain number of time-steps.

The difference in the solution conflict found by the SQP when triggered is marginal when compared to the solution conflict when taken at the very final time-step for each of the problems in the test suite except g02 (as already described in section 11.2.2). Another problem indicating a slight deviation from the end solution when a credibility count is used, is problem g20. This problem has no known solution and upon further inspection of the results, it becomes clear that the issue lies somewhere in satisfying the constraints, since the PSO does not find a feasible region of the search space. The reasons behind this are beyond the investigation made here, however, it should be noted that no feasible solution has been found in the literature to this problem (to the best knowledge of the author of this document).

Lastly, a comparison between the GP-PSO-SQP and the GP-PSO is to be made, together with the two other leading algorithms who have tackled this suite (the DMS-PSO and the PESO+ algorithms). Since the GP-PSO algorithm uses a relaxation of tolerances for the equality and inequality constraints, the algorithm works by defining a time-step (iteration) bound rather than a FE bound, making comparison with the CEC06

[2]A hard constraint is one which must be satisfied under all conditions

difficult (this results from the constraint handling method utilised).

In order to make a fair comparison, the feasibility and success rate of the algorithms is first considered for comparison (see table 11.6).

| | PESO+ | | DMS-PSO | | GP-PSO | | GP-PSO-SQP | | |
|---|---|---|---|---|---|---|---|---|---|
| | Feasible Rate | Success Rate | Feasible Rate | Success Rate | Feasible Rate | Success Rate | Feasible Rate | Success Rate | % FE Limit |
| g01 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 21.8% |
| g02 | 100.0% | 56.0% | 100.0% | 84.0% | 100.0% | 30.0% | 100.0% | 30.0% | 61.3% |
| g03 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 90.0% | 100.0% | 100.0% | 10.9% |
| g04 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 39.6% |
| g05 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 0.0% | 100.0% | 100.0% | 23.9% |
| g06 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 36.8% |
| g07 | 100.0% | 96.0% | 100.0% | 100.0% | 100.0% | 0.0% | 100.0% | 100.0% | 18.0% |
| g08 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 118.9% |
| g09 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 0.0% | 100.0% | 100.0% | 34.3% |
| g10 | 100.0% | 16.0% | 100.0% | 100.0% | 100.0% | 0.0% | 85.0% | 85.0% | 17.5% |
| g11 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 30.8% |
| g12 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 108.8% |
| g13 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 70.0% | 100.0% | 100.0% | 18.4% |
| g14 | 100.0% | 0.0% | 100.0% | 100.0% | 100.0% | 0.0% | 100.0% | 100.0% | 14.7% |
| g15 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 25.5% |
| g16 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 55.2% |
| g17 | 100.0% | 0.0% | 100.0% | 0.0% | 100.0% | 95.0% | 100.0% | 95.0% | 19.8% |
| g18 | 100.0% | 92.0% | 100.0% | 100.0% | 100.0% | 5.0% | 100.0% | 100.0% | 16.0% |
| g19 | 100.0% | 0.0% | 100.0% | 100.0% | 100.0% | 0.0% | 100.0% | 100.0% | 13.7% |
| g20 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 12.6% |
| g21 | 100.0% | 0.0% | 100.0% | 100.0% | NA | NA | NA | NA | NA |
| g22 | 0.0% | 0.0% | 100.0% | 0.0% | NA | NA | NA | NA | NA |
| g23 | 96.0% | 0.0% | 100.0% | 100.0% | 0.0% | 0.0% | 100.0% | 100.0% | 9.9% |
| g24 | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 99.8% |

Table 11.6: Comparison of success and feasibility rates of the GP-PSO + GP-PSO-SQP with literature. $\%FElimit$ signifies the percentage average number of FEs for the GP-PSO algorithmic set-up (fixed time-step limit of 10000).

Since the benchmark sets a $5e5$ FE limit, the percentage mean number of FEs used for each problem for the GP-PSO is quoted. From this table, it is apparent that the same difficulty is found between all algorithms concerning g02 (due to its multi-modality). Concerning the GP-PSO, similar difficulties are apparent with the algorithm that also does not implement a local search (PESO+), with less than 100% success observed for the both, on problems g07, g10, g14, g18, g21, g22 and g23. Additionally, the GP-PSO finds difficulty in a number of other problems, including g03, g05, g09 and g13, however, it is postulated here as being due to the lack of refinement characteristics of the PSO, since the GP-PSO-SQP achieves 100% success on these problems[3].

Regarding feasibility, the power of the relaxation of tolerances based on feasibility is demonstrated by its ability to achieve 100% feasibility in all but two problems. One is g20 (which has no feasible solution in the literature) and g23. g23 is puzzling, since the GP-PSO cannot retain a single feasible particle after relaxation of the constraint tolerances has been removed. However, it is apparent that the constraints also pose some difficulty on these problems with the PESO+, with its less than 100% fea-

[3]It should be noted that the GP-PSO version implemented is simplified, and does not represent the true performance of the cutting edge algorithm. Additional research has been made and is in fact underway concerning the tackling of this very benchmark.

sibility. A significant improvement is observed by using the GP-PSO on problem g17 compared to both DMS-PSO and PESO+ algorithms, which is interesting, since Liang and Suganthan [49] describes their having trouble with this problem due to its highly difficult multi-modality characteristics. Since the GP-PSO achieves such an improvement over the two algorithms on this problem, it is suggested here that perhaps the GP-PSO conducts its search with higher diversity, but then suffers considerably from a lack of refinement characteristics. It should be noted that this hypothesis is based upon the success rates and feasibility rates of the algorithm over different limits and that on average, the GP-PSO is given a much tighter limit than that given to those algorithms applied to the CEC06 benchmark. As a consequence of this, the GP-PSO is expected to achieve much better performance if given a similar FE limit to those algorithms used for comparison.

In order to make a fair comparison between the number of FEs required for the algorithm to find the global optimum, only those problems which result in 100% success rates are included in the statistics. The reasoning behind this is as follows;

The number of FEs required to achieve the fixed accuracy level will be equal to the limit applied to the run if early termination does not occur (fixed accuracy level is not met). Since the runs are bounded by time-steps and not by FEs, a meaningful comparison can only be made on those problems which achieve a 100% success rate and do so to within the limit imposed by the suite. For this reason, three comparison tables are provided: one table compares the GP-PSO-SQP with the GP-PSO for all problems considered (table 11.7), utilising the time-step limit used in this investigation; the second, is a comparison between the GP-PSO-SQP and the literature on problems successful to both (table 11.8); and thirdly, a comparison between GP-PSO, GP-PSO-SQP and the literature, considering a reduced number of problems that are successful on each algorithm (table 11.9).

| Algorithm | % FEs |
|---|---|
| GP-PSO-SQP (count=1) | 40.6% |
| GP-PSO | 53.2% |
| GP-PSO-SQP (count=50) | 73.9% |
| GP-PSO-SQP (count=500) | 100.0% |

Table 11.7: Comparing GP-PSO-SQP GP-PSO. Percentage FEs compared to the maximum mean algorithm. Taking into account all problems, apart from $g21$ and $g22$.

With a comparison of the GP-PSO with the hybridised GP-PSO-SQP (see table 11.7), it is shown that termination can take place on average, earlier than the point at which the

| Algorithm | % FEs |
|---|---|
| DMS-PSO | 12.6% |
| GP-PSO-SQP (count=1) | 25.1% |
| GP-PSO-SQP (count=50) | 45.4% |
| GP-PSO-SQP (count=500) | 63.5% |
| PESO+ | 100.0% |

Table 11.8: Comparing GP-PSO-SQP with literature. Percentage FEs compared to the maximum mean algorithm. Taking into account only 100% successful problems (g01,g03,g04,g05,g06,g08,g09,g11,g12,g13,g15,g16 and g24).

| Algorithm | % FEs |
|---|---|
| DMS-PSO | 14.3% |
| GP-PSO | 18.7% |
| GP-PSO-SQP (count=1) | 30.7% |
| GP-PSO-SQP (count=50) | 60.0% |
| PESO+ | 89.4% |
| GP-PSO-SQP (count=500) | 100.0% |

Table 11.9: Comparing GP-PSO and GP-PSO-SQP with literature. Percentage FEs compared to the maximum mean algorithm. Taking into account only 100% successful problems (g01, g04, g06, g08, g11, g12, g15 and g16).

GP-PSO attains success and achieve the desired accuracy of the GP-PSO-SQP (which is a considerable improvement over the GP-PSO alone). Comparing problems commonly successful to all three algorithms (GP-PSO-SQP, PESO+ and DMS-PSO), indicates a much smaller mean number of FEs required for the GP-PSO-SQP to successfully trigger compared to the PESO+. However, the GP-PSO-SQP does appear to lag behind the performance of the DMS-PSO. Finally, in comparison between those problems commonly successful to all four algorithms (as shown in table 11.9), on average, the switch to local search occurs later than the point at which the GP-PSO finds the global optimum (on average).

The GP-PSO and GP-PSO-SQP are shown to perform somewhere in-between the two algorithms used for comparison. This strongly indicates the power of the hybridised method for solution refinement. However, for problems where the GP-PSO successfully converges early on in the search, the switching to local search occurs much later than necessary. This is again due to the influence of feasibility ratios of the swarm, where the main drive is the point at which the relaxation of tolerances has been dropped and a feasibility of over 20% is retained by the GP-PSO.

### 11.2.4 Summary

It is apparent that the application of the local search, bridges the gap between the two optimisers (PESO+ and DMS-PSO). The PESO+ contains no local search implementation, but the DMS-PSO does, with its implementation of local search during the PSO search phase (together with its final solution refinement). With implementation of the local search to the GP-PSO, it is possible to achieve similar success to the DMS-PSO and at a rather similar computational cost, though still falling short somewhat. Comparing with PESO+, GP-PSO is shown to attain a much faster convergence rate, however, with reduced success rates (though given much reduced computational limits). With the refinement capabilities of the GP-PSO-SQP, it outclasses the PESO+ algorithm as it does not implement a local search. Another reason is perhaps that the dynamics of the search do not allow it to thoroughly search the solution space under certain conditions. This is hypothesised due to the ability of the GP-PSO to successfully find the global optimum in g17 on most occasions. Both PESO+ and DMS-PSO fail to achieve high success on this problem. To counter this success, the DMS-PSO achieves a 100% success on problem g10 and g21, where the GP-PSO-SQP does not.

To conclude, a credibility count of 1 appears to be adequate, with the prominent driving factor for successful switching being determined by the point at which final tolerance is reached (PSO constraint handling technique). The possibility of increasing the credibility count is always to be considered with problems such as g02, who are known for their high level multi-modality, where in this case, a credibility count of 500 is necessary to account for the uncertainty of improvement.

Only one criteria is chosen, 'cri_ERRB', due to the insensitivity to the derived thresholds. One implementation is to disable the credibility count (default, where this is suitable for the vast majority of cases, or at least as observed by the suite investigated). The second option is to choose a count in the range of 1-500, giving the user control over the likelihood for further improvement (keeping in mind that a high credibility count may result in a lack of triggering (g02)). Thirdly, as with the unconstrained problems, the use of clustering measures maybe used if thresholds are derived specifically for the chosen problem, whether by some trial-run or if there is a priori knowledge of the function. Similar to the unconstrained problems, the clustering measures are shown to be problem specific, though to an even greater extent in constrained case, since the constraints confine the swarm in problem specific ways, to which the user may not be able to predict a priori.

The implementation of the local search to the in-house GP-PSO, results in consider-

able improvement of solution quality with guaranteed convergence to locally optimum solutions (which the PSO cannot alone claim). The results also indicate that it is possible to utilise early switching methods, which at the very minimum, function as suitable termination criteria, but in most cases result in serious computational savings.

## 11.3 Test Engineering problems

To conclude on the derived switching method described, a selection of typical test engineering problems are tackled (taken from [52]). These problems include: The pressure vessel problem (PVD); welded beam design (WBD); minimisation of the weight of a tension or compression spring design (TCSD); and finally, Himmelblau's non-linear optimisation problem (HBNLP). The formulation of the PVD problem is discrete, although a continuous version of this problem is tackled for application of the SQP local search, called C-PVD.

With the application of the SQP to the GP-PSO, the SQP achieves 100% accuracy when applied from the very first time-step on C-PVD, WBD and TCSD. However, this is not the case for HBNLP. The solutions found are 100% feasible and the objective function is within $1e-02$ of the literature optimum. With respect to the switching criteria, only a credibility count of 10 is necessary for these problems, with no degradation in the converged solution (with a CC of 10 being only required for the HBNLP problem). Furthermore, with a CC of 10, all sample runs were triggered on all problems.

With a count of 1, a comparison is made with a selection in the literature, in order to emphasise the effectiveness of the local search, together with the minimum required knowledge of the user in its application.

From this, it becomes apparent that the GP-PSO achieves competitive results according to not only the solutions to which it finds, but also the number of FEs required to achieve such levels of accuracies. With the SQP applied with a credibility count of 1, a similar number of FEs is required to the point at which the GP-PSO alone finds the global optimum (on average). This confirms the observations of this switching method as a suitable early termination measure. The HBNLP problem required a credibility count of 10 to achieve the same level of accuracy as the case when a local search is applied at the very last time-step (as previously discussed). With this increased credibility count, a mean FE of $9.5e4$ is required, which still compares very well with the results achieved by other authors.

| Prob. | Optimum | SQP Conflict (count1) | | | | GP-PSO | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Fes | Runs | Best | Mean | Fes | Runs |
| C-PVD | 5885.332774 | 5885.332774 | 5885.332774 | 2.8E+04 | 20 | 5885.431448 | 5894.288539 | 1.6E+05 | 20 |
| WBD | 1.724852 | 1.724852 | 1.724852 | 7.7E+04 | 20 | 1.724927 | 1.724942 | 2.2E+04 | 20 |
| TCSD | 0.012665 | 0.012665 | 0.012665 | 1.9E+04 | 20 | 0.012673 | 0.012735 | 1.3E+04 | 20 |
| HBNLP | -31025.561420 | -31025.561420 | -31025.551307 | $*^3$ 6.2E+04 | 20 | -31025.561368 | -31025.561328 | 5.9E+04 | 20 |
| | | Cagnina et al. [58] | | | | Vaz et al. [47] | | | |
| | | Best | Mean | Fes | Runs | Best | Mean | Fes | Runs |
| C-PVD | 5885.332774 | - | - | - | - | 5885.33 | - | 8.8E+05 | - |
| WBD | 1.724852 | 1.724852 | 2.057400 | 2.40E+04 | 30 | 1.81429 | - | 9.6E+05 | - |
| TCSD | 0.012665 | 0.012665 | 0.013100 | 2.40E+04 | 30 | 0.0131926 | - | 7.6E+05 | - |
| HBNLP | -31025.561420 | - | - | - | - | -31012.1 | - | 7.8E+05 | - |
| | | Hu et al. [52] | | | | Worasucheep [59] | | | |
| | | Best | Mean | Fes | Runs | Best | Mean | Fes | Runs |
| C-PVD | 5885.332774 | - | - | - | - | - | - | - | - |
| WBD | 1.724852 | $*^1$ 1.72485084 | $*^2$ 1.72485084 | 2.00E+05 | 11 | 1.724852 | 1.724852 | 2.0E+05 | 30 |
| TCSD | 0.012665 | 0.012666 | 0.012719 | 2.00E+05 | 11 | - | - | - | 30 |
| HBNLP | -31025.561420 | -31025.56142 | -31025.56142 | 2.00E+05 | 11 | *-31026.647264 | -31002.170814 | 2.0E+05 | 30 |

Table 11.10: Comparison of results for the four standard engineering problems between the GP-PSO, GP-PSO-SQP and the literature. *Corrected conflict according to given design variables (in brackets), $*^1$1.72485084 (1.724855), $*^2$1.72485084 (1.724855), $*^3$ $9.5E+04$ with credibility count 10.

## 11.4 Conclusion

The investigation into the implementation of a local search with the in-house particle swarm optimiser (GP-PSO) has been made. Improvement with the hybridisation of the GP-PSO is apparent on most problems and success rates substantially increased by this method. This has been demonstrated through the refinement of solutions provided by the GP-PSO. In addition to this, methods in which to successfully switch between the global and local search have been investigated and tested, resulting in a mean computational expense (indicated by FEs) comparable to the point at which the GP-PSO finds the global optimum (on average). Unfortunately the general success of this switching method appears to be limited to the constrained problems, which derives from the novel constraint handling technique implemented by Innocente [3], utilising important information about the feasibility of the swarm, rather than the continuing swarm dynamic measures themselves.

While the GP-PSO is shown to be competitive amongst other algorithms, at least on the tested problems investigated here, the hybrid between the GP-PSO and the SQP appears to have bridged the gap between those algorithm which themselves utilise a local search with those who do not.

# Part III

# Ant Colony Optimisation

# Chapter 12

# Background

## 12.1 Preferace

Significant resources for the review of recent trends of the ACO paradigm until the point at which it was published (2005), together with a brief introduction to its origins can be found in a paper by Blum [9]. This source, together with "Swarm Intelligence from Natural to Artificial Systems" by Bonabeau et al. [60] and "Ant Colony optimisation" by Dorigo and Stützle [5], provide a good grounding for the paradigm known as ACO. However, further research of the current trends is made amongst the literature regarding various topics, including successful areas of development.

## 12.2 Origins

ACO has its origins in the early 90's by Dorigo et al. [61, 62] as an approximate method for dealing with combinatorial optimisation (CO) problems, where stigmergy is the method in which the agents (ants) indirectly communicate by interacting with the environment. The paradigm is inspired by the foraging behaviour of real ant colonies and the earlier investigations by Deneubourg J. and colleagues. Modelling this natural form of optimisation in an artificial manner for discrete and continuous problems has become an ever popular method. However, research on ACO only began to flourish after Dorigo et al. published their paper on Ant System (AS) [62].

ACO belong to the group of approximate methods such as artificial intelligence and is known as one of the most successful swarm intelligence methods (swarm intelligence being a subset of artificial intelligence). Although ACO is placed amongst the category

of swarm intelligence, it technically does not belong here (depending on set-up and the problem being tackled). As described by Clerc [63], the original AS (Dorigo et al. [62]) works without the need for 'swarm intelligence', since the ants only share information via the pheromones on arcs and ants modify these independently. However, this topic is discussed by Dorigo and Stützle [5] directly, where having a 'swarm' is described as necessary in the cases of a differential path effect.

Swarm intelligence has its goal in the design of intelligent multi-agent systems. It takes inspiration from the collective behaviours of social insects such as ants, termites, bees, wasps, and other animal societies such as flocks of birds or fish schools. [9]

Since ACO can be considered as a single agent (ant), constructing a number of paths before choosing to influence its future behaviour (next ant cycle), it is concluded here that is does not fulfil this collective behaviour required by the definition as swarm intelligence.

ACO belongs to the category of *Ant Algorithms*, that is, algorithms inspired by ants, however this can be sub-categorised into a number of ways (see section 5.3). ACO belongs to a sub category of models inspired by foraging and path-making as described by Dorigo and Stützle [5], where other algorithms are 'Edge ant walk' and the 'Vertex ant walk' as described in section 5.3.3. Though similar to ACO, they were discussed as having difficulty in their implementation. Other sub categories are those inspired by 'division of labour', 'brood sorting' and 'cooperative transport' which have shown great promise with respect to robotics. This is discussed in greater detail in section 5.3.

## 12.3 Real ants and the Simple Ant Colony Optimiser (SACO)

Real ants initially randomly set out exploring the area while leaving chemical (pheromone[1]) trails behind them which other ants may follow, increasing their probability to follow that particular path. After an ant finds its source of food, it may leave a quantity of pheromone on its return journey dependent on the quantity and quality of its food source. Through this method, other ants will lead to the finding of the food source with likely the shortest possible route, through only the indirect communication by modification of the environment by previous ants with use of their pheromone trails (stigmergy). However, there is a number of differences between a real ant colony and

[1]Pheromone is a chemical substance that ants deposit and can smell.

the developed algorithms for solving real optimisation problems, since a real ant colony is highly prone to become stuck in sub-optimal solutions and real-world optimisation problems are considered much more complex [5]. The individual behaviour of an insect (ant) does not necessarily meet with success, however, with the collective performance of the colony, likelihood of their success increases and so cooperation is key. This is partly genetic with different types of ants being physically different in the case of division of labour, however collective activities can derive from Self-Organisation (SO) as described in [60], where the collective behaviour of the ants is an emergent property through the interaction amongst individuals. This method is considered highly flexible and robust. Flexible, since tuning is possible through the changing of the environment and robust, as the colony will continue to function even if some individuals fail to carry out their task(s).

The foraging behaviour of many species of ants has been observed by a number of researchers, the most notable of which are of Daneubourg J. and colleagues as described by Bonabeau et al. [60]. Daneubourg created the double bridge set-up to observe the behaviour of ants, with the hope of gaining insight into the power of pheromone trail laying. The most significant results are explained here.

The first set-up consisted of two branches of equal length as shown in fig. 12.1, where the choice of branch was random to begin with. After some time, all the ants would tend to choose one of the two paths, with the probability of ant $k$ choosing branch $i$ being given by eq. (12.1). The general formulation for the probabilistic choosing of branches between a nest and food supply for any number of branches is then given by eq. (12.2).

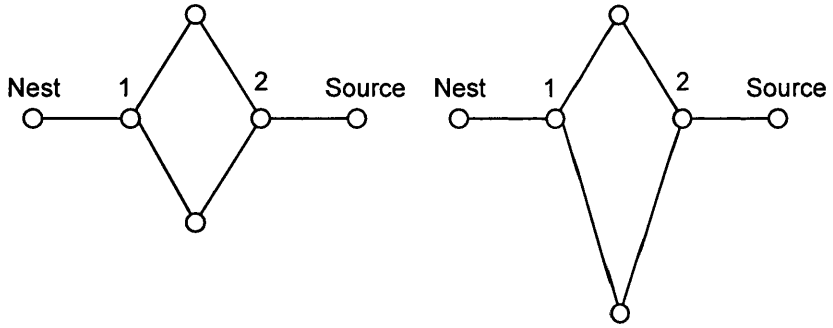



Figure 12.1: The double bridge experiment with the left figure showing equal length arcs and the right hand figure of unequal lengths.

$$\mathbf{p}_i = \frac{\tau_i}{\tau_1 + \tau_2}, i = 1, 2 \tag{12.1}$$

$$\mathbf{p}_i = \frac{\tau_i}{\sum_1^n \tau_i}, i = 1, 2, ..., n \tag{12.2}$$

Since the random fluctuations in the initial choosing of the branches causes one of the branches to have slightly more pheromone, this pheromone difference between branches then becomes more and more until eventually all (nearly) ants converge to a single branch path. This is described by Bonabeau et al. [60] as a *positive feedback* or *auto-catalytic process*, which is an example of the self-organising behaviour of ants.

The second set-up considered two branches of unequal lengths as also shown in figs. 12.1 and 12.2, where ants again begin with an equal chance of choosing either path. Ants following the shorter path would reach the food source before those following the longer path and as such, will have reinforced their trail with pheromone on their return journey before those ants following the longer path have chance to make their return journey. From this, is it clear that the ants are highly likely to converge to the shorter path over time. In fig. 12.1, the Left figure shows the initial state where ants begin at the nest and have a 50/50 chance of choosing either branch. The middle figure shows the point at which ants on the shorter branch reach the food source before those following the longer branch. The right hand figure shows the return journey of the ants where those that followed the longer branch have increased likelihood of following the shorter branch as they are biased by the pheromone trails already deposited by the ants already returning on it.

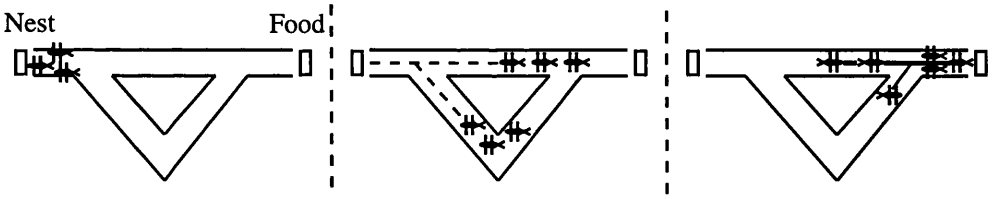



Figure 12.2: Double bridge experiment. Illustrating the possible path of ants. The thickness of the red line is an indication of pheromone strength.

An additional set-up was also made by which only a single long path is possible and then an additional short path is introduced between the nest and the food. It was observed by Daneubourg that the ants are likely to be stuck on the long path, since it is too heavily reinforced. From this observation, the general statement maybe made that

real ants are subject to generating loops and may get stuck in suboptimal solutions. The forward updating of pheromone trail result in the suboptimal convergence of ants as loops can become highly attractive. Since real ants cannot converge to the shortest path if either the forward of backward pheromone update were to be removed [5], deviations from real ant simulations are apparent in the approach of real problem optimisation problems.

Simple ACO (SACO) was the algorithmic implementation of the double bridge experiment with additional changes or tunings made from the original modelling of the natural ant colony for the above reason of suboptimal convergence. Firstly, a limited memory is implemented, storing partial paths together with the cost of the links so that artificial ants may deposit only on their return journey. The overall differences between them can be summarised as follows:

- Real ants move asynchronously, where artificial ants move synchronously.
- Probabilistic solution construction, biased by the pheromone trails without forward updating is made by artificial ants.
- A deterministic backward path with loop elimination and pheromone update is made (with use of the limited memory) by artificial ants.
- Artificial ants evaluate the quality of a solution to determine the quantity of pheromone to deposit as opposed to real ants who forage based on implicit evaluation of a solution (path length).
- Artificial ants use a negative feedback with the consideration of pheromone evaporation (which is not required in the rather simple organising systems apparent in a real ant colonies). This evaporation of the pheromone also functions as a maximum value for the pheromone trails. This pheromone evaporation has the effect of allowing the colony to escape suboptimal trails and avoid stagnation.

SACO can be considered to work in two modes; forward mode where a solution is built by the probabilistic choosing of neighbouring nodes biased by previously deposited pheromones (no pheromones deposited); and backward mode, where the limited memory described previously allows the ant to re-trace the path it followed together with the elimination of loops, where the ant now deposits pheromone dependent on the quality of the solution path. Loop removal is achieved by scanning iteratively for the first encounter of a node from source to destination, where loops are eliminated in the

order they are created. An iteration in SACO is then defined as a complete cycle after all ants movement, pheromone evaporation and deposition have occurred. The pseudo-code for this algorithm is shown in alg. 6.

**Algorithm 6** Pseudo-code - SACO

```
1: Initialise Begin at source node, pheromone of all arcs set to 1.
2: while termination conditions not met do
3:     Update Probabilistic choosing of ants next node location
4:     Calculate Loop removal
5:     Calculate Pheromone update
6:     Test termination condition(s) met?
7:     if conditions met then
8:         Exit
9:     else
10:        continue
11:    end if
12: end while
```

The probability which an ant chooses node 'j' in SACO is given by:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{l \in N_i^k}} & \text{if } j \in N_i^k; \\ 0, & \text{if } j \notin N_i^k; \end{cases} \tag{12.3}$$

Where $N_i^k$ is the neighbourhood of ant $k$ at node $i$.

The return travel (pheromone update) is then defined by:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^k \tag{12.4}$$

where $\Delta\tau^k$ is the change made to the pheromone, in most cases set to the differential path length (i.e. dependent on solution quality). The pheromone trail evaporation is then given by:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \forall (i,j) \in A, \tag{12.5}$$

where $\rho \in (0, 1]$.

In the original definition of ants, the coefficient $\alpha$ is found to be equal to 2 (Danembourg), which is a measure of the amplification of the random fluctuations. However, it is found to be more efficient to set this to 1 in the case where path length is taken into

account (where the amount of pheromone is inversely proportional to the length of the path they have found). In terms of pheromone evaporation, it is found that setting it too high (fast evaporation) results in suboptimal convergence, so setting $\rho = 0.001$ seems to be a reasonable value for best results [5]. The ants are said to evaluate a solution by means of both implicit or explicit means, where implicit evaluation takes place by exploiting the differential path length effect (with following ants biased). Explicit evaluation comes from the pheromone deposited according to some quality function of the constructed solutions.

## 12.4 Most popular algorithms in use

The general 'skeleton' for ACO algorithms when applied to static combinatorial problems as described by Dorigo and Stützle [5] is shown below in the form of a high-level pseudo-code (alg. 7).

**Algorithm 7** Pseudo-code - high-level ACO

```
1: Initialise parameters, pheromone trails
2: while termination conditions not met do
3:     Construct ant solutions
4:     Daemon actions(optional)
5:     Update pheromones
6: end while
```

It should be noted that the *daemon actions* are the procedure used to implement centralised actions which cannot be performed by single ants i.e. to make use of information outside that available to the single ant. The most obvious example of which would be the use of a local search.

Since there are a number of algorithms available in the literature, only the two most popular and successful are described in detail when applied to the TSP, however, other popular algorithms are described. These include the MAX-MIN Ant System (MMAS), Ant Colony System (ACS), Rank-Based AS (AS_rank), Elitist strategy (EAS) and the Approximate Non-deterministic Tree Search (ANTS). The two most theoretically studied of which are the MAX-MIN Ant System (MMAS) and Ant Colony System (ACS) algorithms to which will be discussed in further detail.

### 12.4.1 Ant System (AS)

Ant System (AS) is the original algorithm published by Dorigo et al. [61, 62], and resulted in the growing research in ACO amongst the optimisation community. The AS algorithm improved over the S-ACO by its implementation of heuristic information and by adding memory capability (tabu list). AS as published in [62], the most successful of three set-ups investigated in this paper was called the ant-cycle set-up and developed as a test-bed for the TSP problem. The three set-ups of the 'Ant system' are described; Ant-density; Ant-quantity and Ant-cycle where the first two are local and the third a global set-ups (pheromone update based at the end of a tour). The difference between these set-ups are within the pheromone update.

The AS algorithm is described by the following:

- Random choosing of a node to start at.
- Ant builds tour in the TSP graph by moving to a neighbouring node that it has not yet visited (making use of a memory of all visited nodes).
- Each step - traversed edge added to the solution under construction.
- When no unvisited nodes are left, the ant closes a tour by moving from current node to the node at which it started construction (each ant $k$ has a memory $T^k$).

The equation to which an ant 'k' chooses the next node is given by what's called the random proportional rule (action choice rule):

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j \in \mathcal{N}_i^k \tag{12.6}$$

Where $\eta$ is the heuristic value or heuristic information which represents a priori information about the problem instance or run-time information not provided by the ants. In many cases this is the cost of connecting a component/connection to the solution which is under construction, however in the case of the TSP for the AS, it is defined as the inverse between city distances ($\eta = 1/d_ij$). The heuristic informations allows an explicit bias towards the most attractive solutions and by this very definition is a problem-dependent function. Parameters $\alpha$ and $\beta$ now determine the relative influence of the pheromone trail and the heuristic information. As already discussed, $\mathcal{N}_i^k$ is the feasible neighbourhood of ant 'k' where the feasible neighbourhood in the case of the

TSP are those cities not yet visited. It is described as absolutely necessary in stopping a randomly initialised system from reinforcing bad tours, however it is also described that heuristic information is required less and less as the search progresses [5]. With the use of a local search method, the heuristic information is also described as no longer being absolutely necessary.

Once all ants in the colony have completed construction of their solutions, pheromone evaporation (on all arcs) is performed according to:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} \tag{12.7}$$

where $\rho$ is the pheromone evaporation (parameter).

The ants then perform return trips and in doing so, perform a pheromone deposit based on the quality of the solution:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{12.8}$$

Where

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k}, & \text{if arc (i,j) belongs to } T^{k}; \\ 0, & \text{otherwise;} \end{cases} \tag{12.9}$$

As described in [5] the performance of AS when compared with other metaheuristics tends to decrease as the size of the instance increases. To improve upon this algorithm, researchers began looking into search control, where a stronger exploitation of the search history to direct ants is made to improve performance. The most popular and successful of which are described in some detail. The need for a greedier algorithm than AS however causes problems of premature stagnation and so methods in which to combine an improved exploitation of the best solutions found during the search together with a mechanism in which to avoid early search stagnation is observed in the extensions to AS. Another characteristic feature of all extensions detailed here is that they are all considered elitist methods.

### 12.4.2 Ant Colony System (ACS)

This algorithm by Dorigo and Gambardella [64] has its main differences with the AS algorithm in that it exploits search experience more than in AS. This algorithm also deposits pheromone and evaporates only on arcs belonging to the best-so-far tour. Also, each time an ant moves from one city to another, it removes some pheromone from the arc to increase the exploration of alternative paths.

The transition rule is now replaced by a new transition rule called the pseudo-random-proportional action rule where an ant 'k' at node 'j' moves according to:

$$j = \begin{cases} \text{argmax}_{l \in \mathcal{N}_i^k} \{\tau_{il} [\eta_{il}]^\beta\} & \text{if } q \leq q_0; \\ J, & \text{otherwise;} \end{cases} \tag{12.10}$$

where $q$ is a random variable, uniformly distributed in [0,1], $q_0 (0 \leq q_0 \leq 1)$ is a parameter and $J$ is a random variable selected according to the probability distribution given by eq. (12.6) with $\alpha = 1$ (i.e. when $q \geq q_0$ the same transition rule as AS is used).

So with probability $q_0$ an ant makes the best possible move while with probability $(1 - q_0)$ it performs a biased exploration of the arcs. Changing the parameter $q_0$ allows the degree of exploration to be varied and to whether the search is concentrated around the best-so-far solution or to explore other tours. That is, a bias is made towards nodes connected by the shortest links together with a large amount of pheromone with parameter $q_0$ allowing to balance between exploration and exploitation. As $q_0$ becomes smaller, the less that best links are exploited while as $q_0$ is larger, the more exploration is made.

An important distinction with AS is that the pheromone update is performed by one ant (best-so-far ant) and this occurs after each iteration and so the pheromone update equation is defined by;

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho \Delta \tau_{ij}^{bs}, \quad \forall (i,j) \in T^{bs} \tag{12.11}$$

where $\Delta \tau_{ij}^{bs} = 1/C^{bs}$. Trail update and evaporation only occur on arcs that belong to $T^{bs}$ and not to all arcs like in AS and as such, the computational complexity is greatly reduced. This pheromone update is then the weighted average between the old pheromone value and the amount being deposited. Note that applying to iteration -best or global-best is possible and has been implemented by Dorigo and Gambardella. Unlike AS, the pheromone values in the above equation have the effect of influencing the influence of

the best route found, that is, with small values of $\rho$, pheromones concentrations on the links evaporate slowly and influence of the best route dampened and vice versa. The effect of a large $\rho$ is neglect previous experiences and favour recent experiences (those of higher pheromone concentrations).

A local pheromone trail update is also performed after crossing an arc (i,j) during the tour construction and is given by;

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0 \tag{12.12}$$

where $0 < \xi < 1$ and $\tau_0$ are parameters. $\tau_0$ is set to be the same value as the initial value for pheromone trails. $\xi$ found to be good for setting as 0.1 and $\tau_0 = 1/nC^{nn}$, where $C^{nn}$ is the length of the nearest neighbour tour.

The effect of this local update is that the pheromone trail is reduced, increasing exploration of arcs not yet visited and due to this, no stagnation behaviour is observed with this algorithm (ants don't converge to a common path). It should be noted that sequential/parallel construction matters in ACS due to this local trail update. Normally this method is implemented in parallel.

Like in MMAS, a pheromone trail limit is given though implicitly by:

$$\forall(i,j) : \tau_0 \leq \tau_{ij} \leq 1/C^{bs} \tag{12.13}$$

A candidate list is also implemented with this algorithm which defines for each city $i$ a list of cities $j$ that are close to it. It sorts the neighbours of a city $i$ according to non-decreasing distances and then inserts a fixed number of the closest cities into $i$'s candidate list. Note that the neighbourhood is the cities that it has not visited yet however the candidate list is the list of closest cities to the current city. The candidate list method can however be applied to a number of the algorithms.

A quote taken from [64] describes well, a simple yet intuitive explanation of the ACS algorithm:

> *ACS can be seen as a sort of guided parallel stochastic search in the neighbourhood of the best tour.*

The number of ants as described by Dorigo and Gambardella [64] is experimentally determined to be 10 by studying the relation between three family groups of tours with

the calculation of average pheromone-closeness: one family group is the arcs that belong to the best-so-far tour; another family are those arcs that belong to the previous two iterations best-so-far tour, and finally the last tour are those that do not belong to these two families. No theoretical relation could be made as a function of instance size however the number 10 is described as experimentally suitable. Heuristic information is described as being key in the success of the ACS algorithm allowing it to find good solutions in reasonable time and the pheromones are essential since this has the effect of cutting off cooperation (reinforcement provided by the global updating rule). The use of a local optimiser in its implementation was made in [64] which was a 3-opt method, giving the algorithm the abbreviation ACS-3-opt.

### 12.4.3 MAX-MIN Ant System (MMAS)

This algorithm was explicitly developed by Stützle and Hoos [65, 66, 67] to face the stagnation issue in AS (ants following the same path, occurring prematurely due to ants too rapidly exploiting the highest pheromone concentrations and as such, suffer from a lack of exploration). Only either the iteration-best ant or the best-so-far ant is allowed to deposit pheromone in MMAS. On its own, this may lead to stagnation where all ants follow the same tour due to the excessive growth of pheromone trails on good but sub-optimal tours. This is counteracted by introducing a limit to the range of pheromone trail levels to an interval $[\tau_{min},\tau_{max}]$ and the pheromone trails initialised to this upper limit which is the main underlying difference with AS. These pheromone trails are reinitialised to the maximum value each time stagnation occurs or when no improved tour is generated after a number of consecutive iterations (where stagnation in MMAS is often measured by means of the $\lambda$-branching factor to which will be discussed in further detail.

After all ants have constructed a tour, evaporation occurs as normal in AS, but then deposition according to;

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau ij^{best} \tag{12.14}$$

where $\Delta\tau_{ij}^{best} = 1/C^{best}$.

The ants allowed to deposit pheromone uses either; $\Delta\tau_{ij}^{best} = 1/C^{bs}$ or ; $\Delta\tau_{ij}^{best} = 1/C^{ib}$. Where $C^{ib}$ is the length of the iteration-best tour. These are normally used in an alternate way (frequency determines the greediness of the alg.) as described in [66].

Using the global-best path only may cause the search to concentrate too quickly around the global best solution (limiting exploration) while utilising the iteration-best path also reduces this problem.

The upper and lower limits impose pheromone trail limits (limit the probability $p_{ij}$ of selecting a city $j$ when an ant is at city $i$ to the interval $[p_{min},p_{max}]$, with $0 < p_{min} \leq p_{ij} \leq p_{max} \leq 1$. When ant $k$ has only one possible move (one neighbour) then $p_{min} = p_{max} = 1$. After a time the upper pheromone limit on any arc becomes bounded by $1/\rho C^*$, where $C^*$ is the optimal tour. An estimate of this value is made $1/\rho C^{bs}$ in order to define $\tau_{max}$ and so each time a new best-so-far tour is found, $\tau_{max}$ is updated. $\tau_{min} = \tau max/a$ where $a$ is a parameter. $\tau_{min}$ is more important since it avoids premature stagnation.

MMAS is described as being one of the most studied and successful ACO algorithms and since MMAS like AS, ants construct solutions at each step at the same time (in sequence) that parallelising is easy to implement.

The re-initialisation of pheromone trails as defined in [66] is considered where stagnation occurs. The pheromone trail evaporation is said to be a highly expensive on each iteration (which is not the case for ACS) and so a candidate list is often used as is the case in [66].

The most important observations made by Stützle and Hoos [66] appear to be:

- Initialising the pheromone to $\tau_{max}$ improves performance due to it then favouring larger exploration at the beginning.

- $\rho$ determines the convergence speed, and having a low value for a low number of iterations gives better results (pheromone trails on arcs which are not reinforced decrease faster so the search concentrates earlier around the best tours so far).

- $\rho$, if too high then too few iterations performed so that still the difference between arcs belonging to the best tours and the rest is too small.

- $\rho$, higher value for a larger number of iterations is recommended.

- updating trails with iteration-best ants results in better performance on average compared to just using the global-best ant (worst solution for standard set-up found to be better than average solution of $s^{gb}$.

- Lower pheromone trail limit helps improve the performance when using $s^{gb}$.
- for larger instances it is recommended to use of both $s^{ib}$ and $s^{gb}$ with a frequency that increases over time to increase convergence speed and solution quality.

Multiple set-ups for the MMAS algorithm were made, one with what is called 'pheromone trail smoothing' (PTS) MMAS$_{+pts}$ and one without MMAS where the one with was found to perform better. PTS works by increasing the probability of selecting solution components with low pheromone trail, and is said to be likely to be advantageous in all elitist versions of AS. The smoothing strategy was used since stagnation was still observed by the MMAS algorithm, and works by reducing the difference between high and low pheromone concentrations. This strategy works at the point which stagnation occurs, the pheromone concentrations are increased proportional to the difference with the maximum bound as described in [6]. That is:

$$\Delta_{ij}(t) \propto (\tau_{max}(t) - \tau_{ij}(t)) \tag{12.15}$$

Other set-ups include MMAS$_{+nri}$ which does not use any re-initialisation, MMAS$_{+ri}$ which uses re-initialisation of its pheromones and MMAS$_{+rs}$ which uses re-initialisation of pheromones together with the use of the restart-best solution on occasion rather than the best-so-far solution.

The way in which these set-ups differ are described in the following:

**Algorithm 8** MMAS$_{+ri}$

```
1: if Convergence condition met (branching factor) then
2:     if No improvement for ≥ 50 iterations then
3:         Re-initialise pheromone trails to τ_max
4:         Restart f^gb          ▷ frequency of global best ant in pheromone update
5:     end if
6: end if
```

**Algorithm 9** MMAS$_{+rs}$

```
1: if > 250 iterations without re-initialisation then
2:     if ≥ 25 iterations without improvement then
3:         Use iteration best solution s^ib instead of s^gb
4:     end if
5: end if
```

Overall the MMAS$_{+rs}$ algorithm set-up was concluded to be best. Testing of the algorithm with a local optimiser is done and results making use of the Lin-Kernighan

heuristic are shown to be of much higher solution quality but at the cost of much greater computational time. Early stagnation is avoided using this algorithm by the implementation of the pheromone limits but also the sometimes used trail smoothing (not implemented in the freely available code online). Finally the heuristic information is described as having less importance as time goes on and that with local optimiser implementation, the importance of the heuristic information dramatically decreases.

Another variation (again not standard) is in the incorporation of the pseudo-random action choice rule of ACS to make MMAS a greedier algorithm ([65, 68]), where this algorithm was given the name MMACS. For this case, the larger $q_0$ the tighter the pheromone trail limits chosen are required (lower $\tau_{max}/\tau_{min}$ ratio), in order to prevent ants from preferring links of high intensity as described in [6]. This algorithm was shown to be rather promising though few in number are the number of authors who implement this algorithm.

### 12.4.4 Population based approaches

Regarding population-based approaches, these are distinguished here, since it seems that a population of solutions leads logically toward niching [69, 70], dynamic problems [71] and multi-objective problems [72, 73] as was made with the PACO algorithm. The two mentioned here are the population-based ACO (PACO) by Guntsch and Middendorf [74] and *Omicron* ACO (OA) by Barán and Gómez [75].

#### 12.4.4.1 Population-based ant colony optimisation (PACO)

PACO is particularly effective for multi-objective problems, dynamic and continuous problems, where motivation for its implementation was in the application to dynamic problems as discussed by Guntsch [76] in his PhD thesis.

The PACO algorithm works in a similar way to the standard AS, only that no pheromone evaporation occurs (it is instead removed) and that a population of solutions is maintained (whereas solutions are lost in the case of other ant algorithms). A significant difference is that the pheromone deposited is not based on the quality of the solution generated, but instead, the deposit amount is fixed according to the limits imposed. The gbest solution is put into the initially empty solution archive and after $k$ generations there are $k$ solutions in this archive. From generation $k+1$ onwards, one solution within this population is removed and an amount of pheromone subtracted to elements belonging to this solution. Solutions entering this population result in the

deposition of pheromone on arcs it has visited. There are a number of methods possible for determining which solutions enter and leaves the archive. Those considered by Guntsch [76] are listed here;

- Age-based - Oldest solution in the population leaves at each generation with the iteration-best entering it.
- Quality-based - Worst solution in the population leaves at each generation with the best entering it.
- Probability-based - Probability of any solution within the population leaving, though weighted toward worse solutions.
- Age-Probability-based - ibest solution is included in the new population with a probability-based removal of solution that does not include this ibest solution.

The age-based strategy means that each solution has $k$ iterations of influence after which it is removed, giving this strategy a highly explorative approach; quality-based leads to a highly convergent behaviour (after some time, the best-so-far solution is likely to be found $k$ times in the population); probability-based is designed to counter the effect somewhat of the quality-based approach; and finally the age-probability approach ensures the feature of the age-based strategy with its speed with the probability based strategy which can hold good solutions. Inclusive of these set-ups, there is an elitist strategy which can be implemented which excludes the best solution from this update within the population.

The original name for PACO, was the *FIFO-Queue* algorithm, an early age-based strategy. This algorithm is rather promising, since even though little interest has been shown by researchers, in those that do, such paths as niching have been successively approached and also dynamic and multi-objective problems too (which are at the topics of interest at the moment in ACO). A recent paper by Oliveira et al. [77], which conducted a parameter study of the algorithm, concluded that it outperformed MMAS for short runs. When Oliveira et al. implemented a similar restart of pheromone procedure to MMAS, it was found to perform competitively to MMAS over longer runs as well. The most significant advantage that PACO has over other algorithms, is its population archive. This offers a significant advantage to the paradigm, enabling many more features to be borrowed from GAs for example (in respect to its approach to niching and multi-objective problems ). However, it is not clear as to the best set of rules which

define those which enter and leave the archive. Another significant advantage of PACO over other algorithms such as AS, MMAS and ACS, is that since the pheromone update is based on only $n$ additions and $n$ subtractions, that a significant reduction in computation time is apparent ($O(n)$). In MMAS, the entire matrix undergoes an evaporation giving it a computation complexity $O(n^2)$ for this task. In ACS, even though it has $O(n)$ complexity, it is multiplied by a constant $(1-\rho)$, which makes each step higher in complexity, together with the fact that a local pheromone update is used (see section 12.4.2). Oliveira et al. [77] used a profiling tool to compare between ACS, MMAS and PACO and concluded that the pheromone update of PACO to be much faster than both MMAS and ACS algorithms.

To discuss similarities with the other algorithms, $\tau_{min}$ and $\tau_{max}$ are imposed (taking inspiration from MMAS). The update of the pheromone for solution $\pi$ is a positive update for solutions entering population and negative for those leaving the population, given by the following (eqs. (12.16) and (12.17)):

$$\tau_{i\pi(i)} = \tau_{i\pi(i)} + \Delta, \quad \forall i \in [1,n] \quad \text{solution entering the archive} \tag{12.16}$$

$$\tau_{i\pi(i)} = \tau_{i\pi(i)} - \Delta, \quad \forall i \in [1,n] \quad \text{solution leaving the archive} \tag{12.17}$$

where $\Delta$ is determined by the size of the population $k$ and the bounds of the pheromone. $\tau_0$ is said to be set arbitrarily to $\tau_0 = 1/(n-1)$ for the TSP, so that all columns and rows add up to 1, since $\tau_{max}$ maybe scaled accordingly. In the case where an elitist method is incorporated into the algorithm, with $\Delta$ corresponding to the update made by normal members and $\Delta_e$ by elitist members, the following formulation is used:

$$\Delta = (1-w_e)\frac{(\tau_{max}-\tau_0)}{k} \tag{12.18}$$

$$\Delta_e = w_e\frac{(\tau_{max}-\tau_0)}{k_e} \tag{12.19}$$

where $k_e$ corresponds to the number of elitist members (default= 1). It should be noted that those sources which describe the algorithm are not clear in the formulation of this update, and so a formulation which intuitively makes sense is used. An illustration of the magnitude of pheromone deposit is shown in fig. 12.3, where the deposit of every ant, including the elitist results in the maximum pheromone deposit ($\tau_{max}$). This is

the interpreted meaning of the update used here which should be noted to differ to the interpretation given by Oliveira et al. [77].

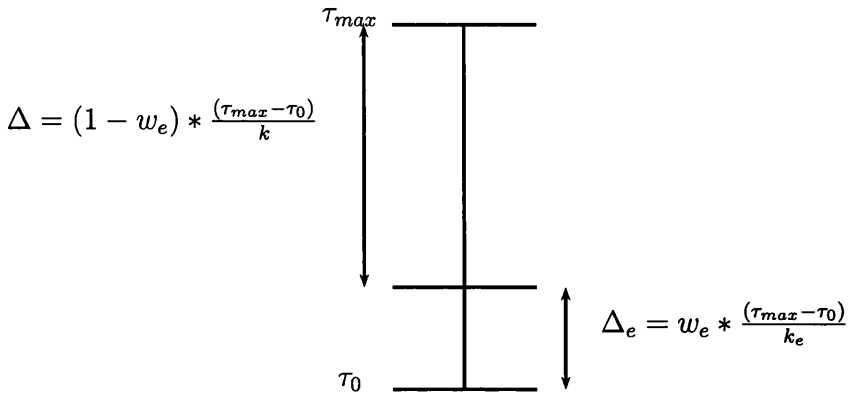



Figure 12.3: Illustration of the idea behind pheromone deposit and removal in PACO.

A pseudo-code is now provided to aid in the exact application of the algorithm (see alg. 10), with influence taken from a thesis by [78].

In the static application of the algorithm as described in [74], the age-based strategy was found to be the most effective, however, in the dynamic problem cases [71] it was found to be problem specific. In the case of the dynamic TSP, the Age-Probability based approach with a population of 3 was said to be overall most effective. In the case of the QAP however, where Guntsch and Middendorf used no heuristic information, not much difference between the strategies was observed for small populations except in the probabilistic case which showed worse performance. Overall, the conclusions of the paper were that a population size of 3 was good as a general set-up. Regarding the specifics of the algorithm when tackling dynamic problems, Guntsch and Middendorf used what he calls the *KeepElite* method. It is mentioned that most ACO algorithms applied to dynamic problems apply a repairing of their pheromone matrices, which can be computationally expensive. Using the KeepElite method however, the old solution components are removed and the successor and predecessor are described as now neighbours in this tour, and now the new cities are inserted into the tour individually and greedily, such that this results in the least decrease in solution quality.

An outline of the history of this algorithm is given here, reasoning that its development to a range of problem types makes it a very interesting candidate for further development. Initially, as already described, the algorithm was developed by Guntsch and Middendorf [74] as a population-based approach with dynamic problems in mind and

**Algorithm 10** Pseudo-code - PACO algorithm (age-based + elitist member strategy)

```
Initialise pheromone map to τ_0
Construct m solutions
Update ibest and gbest solutions
Insert elite solution into the population and update with eq. (12.16) (Δ_e)
while Termination criteria not met do
    Construct m solutions
    Update ibest and gbest solutions
    if gbest better than elitist member then
        Remove old elitist solution from the archive and update with eq. (12.17) (Δ_e)
        Insert new elitist member to the archive and update with eq. (12.16) (Δ_e)
    else
        if population (archive) size < k then
            Insert ibest into the population and update with eq. (12.16) (Δ)
        else
            Remove oldest solution from the archive and update with eq. (12.17) (Δ)
            Insert ibest solution from the archive and update with eq. (12.16) (Δ)
        end if
    end if
end while
```

prooved to perform well in them [71]. When applied to multiobjective problems, Angus [73] describes Guntsch and Middendorf as tackling the single machine total tardiness problem with changeover costs in his PhD thesis (2004). Following this, Angus [70] was the first and is still the first to tackle the issue of niching by ant colony, incorporating successful techniques used in EAs such as crowding (SC-PACO) and fitness sharing (FS-PACO) [69, 70, 72, 73, 78, 79]. With the easier analysis of function optimisation for defining the performance of a niching algorithm (since certain functions are obviously highly multi-modal), the PACO is extended to the continuous domain for testing not only on a multi-modal TSP but also common multi-modal benchmark functions from the literature. With multiobjective problems in mind, the natural extension to multiobjective problems was made with the crowding population-based algorithm (CPACO) as described by Angus [73]. Following this, Angus applied his algorithm to multi-objective function optimisation. More recently, a paper by Oliveira et al. [77] showed that even in the single objective cases, that PACO is competitive and may even be considered to perform better if short searches are concerned.

To conclude, this algorithm shows great promise for further development and understanding, from its range of possible applications to the efficiency of its implementation (less computation time required per iteration). The reasoning behind the lack of uptake

of this algorithm in the literature is assumed to be because it incorporates a population of solutions and as such, is not in tailoring with the original ant colony design. When comparing implementations of ACO on multi-objective problems, that is in-fact the exact reason that García-martínez et al. [80] gave for not including it in their comparison.

#### 12.4.4.2 Omicron ant (PACO)

OA by Barán and Gómez [75], is a very similar approach to the PACO algorithm. OA maintains a population of $k$ members which consist of $k$ unique best solutions. The algorithm begins by initialising the pheromone to 1. Solutions are then constructed in the standard way, with ibest solution entering the archive if it is both better than the worst member within it and is different to all other members within it. After $l$ iterations, all arcs are reset to their initial values, then $\frac{O}{k}$ is added to an edge each time it is present in any of the $k$ solutions within the archive. This process is said to repeat until some termination measure is reached. The limits of the pheromone are then implicit $(1 \leq \tau_{ij} \leq (1 + O))$, where the pheromone present on an edge is its minimum (1) if it shares no arcs contained within any of the archive members and maximum (O+1) if it shares every edge with those in the archive. The main difference between this algorithm and the PACO algorithm are said to be that identical solutions are not allowed to enter the solution archive and that updates occur only every $l$ iterations in the case of OA.

This population of best solutions, overcomes the shortcoming of the MMAS algorithm searching around the best only solution, however the OA has the unfortunate drawback in that, with its faster convergence to the $k$ best solutions found, pheromone evaporation does not occur, and so the memory of the population is not retained. Even though this paper results in what seems to be better convergence properties in comparison with MMAS, very limited results are apparent for a proper comparison of their algorithm (only two TSP problems from the TSPLIB are used). Furthermore, no further articles referring to this technique have been found in the literature and so one can only assume its effectiveness in particular cases shown in the article. Additionally it is noticed here, that determining how many edges are common to different solutions is a very computational task. For these reasons, this algorithm is discounted from any further consideration.

### 12.4.5 Other algorithms to consider

Amongst other available algorithms common in the literature, though to a lesser extent than the ACS and MMAS algorithms, is the Rank-Based AS ($AS_{rank}$ or $AS_r$), Elitist ant strategy (EAS or $AS_e$) and Approximate Nondeterministic Tree Search (ANTS) for example. These are briefly described with respect to their differences with AS as discussed by Dorigo and Stützle [5].

#### 12.4.5.1 Elitist AS ($AS_e$)

The first improvement made on AS was the elitist AS (EAS or $AS_e$), introduced by Dorigo and Stützle [5]. It differs to the AS as it has additional reinforcement of arcs belonging to the best tour found since the beginning of the algorithm ($T^{bs}$) which provides additional feedback. This can be considered to have an additional ant called the best-so-far ant and is a daemon action. With the best-so-far-tour denoted by $T^{bs}$ a new parameter is defined $e$, being a weight given to the best tour $T^{bs}$ and its length $C^{bs}$. The pheromones update rule is then modified to:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs} \tag{12.20}$$

where

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/C^{bs}, & \text{if arc (i,j) belongs to } T^{bs}; \\ 0, & \text{otherwise;} \end{cases} \tag{12.21}$$

As described in [5], this strategy with appropriately selected weightings results in better performance of the algorithm with respect to the original AS.

#### 12.4.5.2 Rank-based AS ($AS_{rank}$)

$AS_{rank}$ was proposed by Bullnheimer as discussed in [5], where each ant deposits an amount of pheromone that decreases with rank. The best-so-far ant always deposits the largest amount of pheromone in each direction. Before the pheromone update, the ants are sorted by increasing tour length and the quantity of pheromone an ant deposits is weighted according to the rank $r$ of each ant. In each iteration, only the $(w-1)$ best-so-far ranked ants and the very best-so-far ant are allowed to deposit pheromone. The

pheromone update equation is now modified as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs} \tag{12.22}$$

The $AS_{rank}$ is said to perform slightly better than the elitist strategy and again, significantly better than the original AS as discussed in [5]. To conclude, many authors have chosen to use this method in the tackling of applications though the tendency toward the MMAS and ACS is obvious.

#### 12.4.5.3 Approximate Nondeterministic Tree Search (ANTS)

With the ANTS algorithm described in [5], the lower bounds are computed on the completion of partial solutions to define heuristic information used by each ant during the solution construction and thus the attractiveness of following/adding a particular arc. The method can be interpreted as an approximate way of branch & bound procedure. A novel action choice rule is used which makes this algorithm differ from the AS and also a modified pheromone trail update rule is implemented. Further details regarding this algorithm can be found in [5, 6]. The heuristic information is calculated by adding an arc to the current partial solution and by estimation of the cost of a complete tour with this added arc with use of the lower bound. This method of calculating the heuristic information is said to have its advantage, in that feasible solutions which would otherwise be accepted that would lead to a higher estimated cost with respect to the best-so-far solution are disregarded. However, a significant computational overhead is described in the calculation at each ant step. With regards to solution construction, the probability by which an ant 'k' at city 'i' chooses city 'j' is given by:

$$p_{ij}^{k} = \frac{\xi\tau_{ij} + (1-\xi)\eta_{ij}}{\sum_{l\in\mathcal{N}_i^k}\xi\tau_{il} + (1-\xi)\eta_{il}}, \text{ if } j \in \mathcal{N}_i^k \tag{12.23}$$

where $\xi$ is a parameter between 0 and 1 and $\mathcal{N}_i^k$ the feasible neighbourhood. Only one parameter is used in the above equation unlike the standard equation used in AS. The pheromone update incorporates no evaporation and is defined as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{12.24}$$

where $\Delta\tau_{ij}^{k}$ is given by:

$$\Delta\tau_{ij}^{k} = \begin{cases} \vartheta\left(1 - \frac{C^k - LB}{L_{avg} - LB}\right), & \text{if arc (i,j) belongs to } T^k; \\ 0, & \text{otherwise;} \end{cases} \tag{12.25}$$

where $\vartheta$ is a parameter (often denoted $\tau_0$) and $LB$ is the lower bound value on the optimal solution (computed at the beginning of the algorithm), where $LB \leq C^*$ ($C^*$ is the length of the optimal tour, $L_{avg}$ is described as the moving average of the last $l$ tours generated by the ants (average length of the tours generated by the ants over the $l$ most recent tours). In this way, if an ant's solution is worse than $L_{avg}$ then the pheromone trail on the arcs visited by this ant are decreased and increased if better than. This update based on the moving average has its use to avoid the use of $\rho$ from AS as it is seen as a highly sensitive parameter, easily leading to stagnation if not fine-tuned to the particular problem. The advantages outlined in [5] of using the above pheromone update equation, is that it dynamically scales the objective function. This algorithm was written specifically to tackle the quadratic assignment problem, where heuristic information is said to play a smaller role than in TSP.

#### 12.4.5.4 Best-Worst AS (BWAS)

Another algorithm is called the best-worst ACO (BWAS) by Fernandez I. et al. (2000), which is essentially the original AS, only that the global updating rule is changed for the worst ant. If an edge belongs to the worst ant and does not belong to the best ants tour, then edge updates its pheromone according to:

$$\tau(i,j) = (1-\rho)\tau(i,j) - \eta\frac{L_{worst}}{L_{best}}$$

where $\eta$ is a parameter, $L_{worst}$ and $L_{best}$ are the lengths of the worst and best ants tours respectively. The idea is to further enhance the search around the best tour rather than searching around the worst solution (on the idea that a superior solution may be found in the neighbourhood of good solutions). A recent paper by Li et al. [81], reports an improved performance of the algorithm in terms of 'searching speed and convergence efficiency'[2], through the incorporation of ideas from EA. That is, with use of a heuristic crossover operator extension called improved BWAS (IBWAS) in order to overcome the inherent weakness of initial pheromone updates of the worst ant, since pheromone

[2]convergence efficiency is the time it takes for the algorithm to converge to the global optimal solution

initialisation is based on the distance between cities. In IBWAS, the best and second best ant are said to carry out a heuristic crossover and secondly, the worst ant is not allowed to execute its global updating rule in the initial stages. Results are encouraging but not conclusive since only four problem instances are tested and a lack of information on the details of the results is apparent (i.e. mean or best solution found? stdev of solutions found?). Also, this algorithm is tested on only rather small TSP instances. To conclude on this algorithm, not many researchers use it as a baseline to build from.

#### 12.4.5.5 Hyper-Cube Framework for ACO

This method is not so much an extension of the AS but a method which can be applied to any ACO algorithm and came about by Blum and Dorigo [82]. This method works by re-scaling the pheromone values so that they lie in the interval [0,1]. Quoting from [5] (p.81), it is said "This choice was inspired by the mathematical programming formulation of many combinatorial optimisation problems, in which solutions can be represented by binary vectors." Decision variables take the values $\{0,1\}$ and typically correspond to the solution components as they are used by the ants for solution construction. A solution to a problem corresponds to a corner of the n-dimensional hyper-cube, where 'n' is the number of decision variables.

To generate the lower bounds, the problem is relaxed, allowing each decision variable to take values in the interval [0,1] and as such, the set of feasible solutions $\mathcal{S}_{rx}$ consists of all vectors;

$$\bar{v} \in \mathbb{R}^n \tag{12.26}$$

which are convex combinations of binary vectors;

$$\bar{x} \in \mathbb{B}^n \tag{12.27}$$

An illustration of the meaning of of this convex combination of binary vectors is shown in fig. 12.4. A point which is a convex combination of the binary vectors, is the linear combination of binary vectors that is non-negative and sum up to 1. Here the vector 'v1' is a convex combination of the three points, however, 'v2' is not ('v2' is an affine combination of the three points).

Pheromone values are normalised in the interval [0,1], where the pheromone vector $\bar{\tau} = (\tau_1, ..., \tau_n)$ corresponds to a point in $\bar{\mathcal{S}}$. When applied to the TSP problem, decision variable $x_{ij} = 1$ when the arc is used and $= 0$ when not. Within the hyper-cube

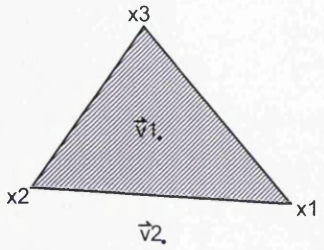



Figure 12.4: Illustration of the convex combination of the three points.

framework, the pheromone trails are forced to stay in the interval [0,1].

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{12.28}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1/C^{k}}{\sum_{h=1}^{m}(1/C^{h})} & \text{if arc } (i,j) \text{ is used by ant } k; \\ 0, & \text{otherwise;} \end{cases} \tag{12.29}$$

where this pheromone update guarantees that the pheromone trails remain smaller than 1.

A recent paper by Birattari et al. [83] however, proves that the three most used ACO algorithms ACS, MMAS and AS, produce solutions that are independent of the scale of the problem and that only the initialisation of both the heuristic information and pheromones are sensitive to scaling. This paper also demonstrates a modification of the heuristic and pheromone initialisation to produce a scale-insensitive formulation and proves the new formalised versions to be functionally equivalent to their original counterparts. Furthermore, the paper describes the Hyper-Cube Framework to be another extension of the AS algorithm rather than a method of implementing an insensitive version of the chosen extension of AS.

#### 12.4.5.6 AntNet

A particularly effective ACO algorithm as described by Dorigo and Stützle [30] in tackling dynamic problems (in particular the dynamic routing network problem), is called the AntNet algorithm by Di Caro and Dorigo M. Network routing is where activities are necessary to guide information from source nodes to the destination nodes. This problem difficulty is said to lye in it stochastic and time-varying properties. Since dynamic

optimisation is beyond the scope of this review, the reader is referred to [5].

## 12.5 Current areas of research

A number of factors are considered with respect to the performance of ACO algorithms and also in their design for a multitude of applications. The main areas of research consist of:

- Parameter tuning or scheduling
- Use of ACO to a wider range of problem types and in increasing performance on these problems.
- increasing algorithm efficiency (removing computational redundancies).
- Theoretical understanding of the algorithm and the dynamics of the search.
- Parallelism of the algorithm

In particular, a discussion of parameter tuning and or scheduling is made here and following this, a discussion is made with regards the TSP with respect to ACS, MMAS and PACO algorithms, since the TSP is used as a problem to further develop algorithmic ideas. Concerning the other topics listed in the above, the reader is referred to appendix B.1.

### 12.5.1 Parameter setting and convergence rates

One of the main focuses in the development of the ACO algorithm is with respect to the setting of their parameters. A number of methods have been approached in the study of setting parameters within the algorithm: one concerns a theoretical study of a selection of problems with the intent to fine-tune the parameters according to problem type; another considers an adaptive method in which to set the parameters during the search, whether through hybridisation with other methods or through means of information gathered during the search, pointing the direction toward a preferred parameter adjustment. A paper by Stützle et al. [84] reviews the advances and recent studies made in the area of *online* parameter adaption. Offline tuning as described in [84] finds the appropriate settings of an algorithm's parameters before the algorithm is used. With regards

to online parameter tuning, Pellegrini et al. [85] describes that parameters are adapted while solving an instance, which then avoids the cost associated with off-line methods. Online tuning is then where the parameters are adjusted during the run of the search. This paper concludes that there is a lack of understanding of the effect of individual parameters on the behaviour of the ACO algorithm. This paper also makes an empirical study of the solution quality as a function of computational time for both ACS and MMAS algorithms. With respect to fixed parameter settings, a conclusion is made that the parameter settings of MMAS depend strongly on the computational run-time and that pre-scheduled parameter variations can improve the any-time performance of this algorithm. A key issue with respect to research amongst the literature is also touched upon within this paper, in that research on ACO is often made without proper comparison with leading ideas and algorithms. For this reason, no clear deduction can be made of its performance with respect to others. This paper also offers a method in which to choose parameter settings or apply scheduled parameter adjustments.

Papers are often released and appear no closer to defining a set standard in which future algorithms should be built by. The most recent papers have further addressed these issues, such as Maur et al. [68], who tackles the problem of the poor (slow) convergence property of the MMAS algorithm (using a merger of ACS with MMAS with its use of ACS's pseudo-random action choice rule) through use of a pre-scheduled or adaptive parameter variation method. The conclusions of which can be summarised as follows:

- An adaptive method and scheduled method show similar performance on the test problems, however it is conjectured that adaptive methods are likely to be the preferred method, where good parameter settings do not vary smoothly during the run (i.e. where pre-scheduled variations cannot be found).

- With the investigation mainly focused on $m$ and $q_0$, $q_0$ was found to have grater influence and no better performing combination could be found with the use of both $m$ and $q_0$ since they are strongly paired parameters.

- Improvement of performance through the use of schedules on $\beta$ and $\alpha$ is indicated but remains as of yet an open question.

- Improvement of adaptive/scheduled methods for the TSP is strong, even with an applied local search, however, there is little to no improvement in the case of the quadratic assignment problem (QAP).

Research in this area appears rather premature and non-conclusive, however, some conclusions can be made on the effect/behaviour of the two leading algorithms based on parameters setting as discussed by Stützle et al. [84]: The ACS was found to be rather insensitive to parameter adjustment during the search with or without an applied local search but was highly sensitive to the initial parameters if no local search was used. The findings of Stützle et al. with regards to the specific parameters are described here:

- $\beta$ - Suitable range of 2-5
- $m$ - 10 (smaller or much larger values produces worse results with the later having likely cause to too much diversification due to the local pheromone update rule).
- $\rho$ - Differences of $\rho$ are almost not perceptible. Without local search, large $\rho$ produces faster convergence. $\rho = 0.1$ produces progressively better results.
- $q_0$ - close to 1 appears best, though in some cases Stützle et al. describes that a value of 1 can lead to search stagnation while a smaller than 0.75 results in very slow convergence (similar results with and without local search).

With regards to the MMAS algorith, Stützle et al. found much greater sensitivity to parameter adgustement during run-time when no local search was applied. The individual parameter findings are detailled below:

- $\beta$ - large values give an advantage especially, during the initial stages of the search compared to the default $\beta = 2$, where a low $\beta$ eventually reaches the same results as the higher $\beta$.
- $m$ - The number of ants shows a clear trade-off between early and late convergence (low number of ants produces best results during early stages of the algorithm but higher number produces much better results during the end of the run). Altering this will inevitably alter the point at which stagnation occurs.
- $\rho$ - A trade-off between small and large values is apparent with the larger value giving faster convergence than the default, however, low values are able to reach the performance of the default ones if given enough time (most notable without local search). Starting with a high evaporation factor and reducing it over time to its default value seems the best approach.

- $q_0$ - with the pseudo-random proportional rule of ACS, there is clear trade-off between high values that perform best over short run-times and low values which revert to the standard MMAS generally resulting in better final performance.

One of the papers highlighted here is by Li et al. [86], which improves the MMAS algorithm by decreasing the convergence time while still stopping premature convergence through improving the search characteristics. It was again noticed that after some time, the likelihood of certain paths being chosen becomes low and so the pheromone level needed to be re-initialised. However, here the triggers for re-initialisation and also to what they are initialised to, changes from the original algorithm as defined below:

$$\tau_{tour} < \frac{(\tau_{min} + \tau_{max})}{2} \rightarrow \tau_{min} \tag{12.30}$$

$$\tau_{tour} > \frac{(\tau_{min} + \tau_{max})}{2} \rightarrow \frac{(\tau_{min} + \tau_{max})}{2} \tag{12.31}$$

This method shows considerable improvement of convergence speed as well as solution quality and offers to be a simple but effective alteration to the original algorithm.

A leading author in this area of research (parameter tuning methods) appears to be Pellegrini et al.. A recent paper by Pellegrini et al. [85] made a rather thorough comparison between on-line and off-line parameter tuning with the MMAS algorithm. A thorough background on this area of the field (common in papers by Pellegrini et al.) is also made. Off-line tuning methods, as already discussed, exploit knowledge in a priori tuning phase and that the parameter values being optimised are based on a training set of instances and is often considered a black-box approach. Examples of off-line methods include F-Race, Iterated F-Race, Calibra and ParamILS. The most popular of which identified here by the number of citation, is by far the F-Trace method. The F-Trace method as designed by Birattari et al. [87], essentially works by an initially very large number of candidate parameter settings being provided and an elimination of certain configurations being made when enough statistical evidence is given against them (inferior ones). This method consists of having a probability for each instance, for which will be solved.

A brute-force approach to solving this problem, brings to light a number of disadvantages, including that the number of poor configurations are as much tested as those that are particularly better and the size of the training set must be defined a priori and no way is used for deciding the number of runs required for each configuration. The

F-Trace method then overcomes the three shortcomings of the brute-force method described.

It should be noted that more recently (2007 [88]), an iterative F-Trace method has been developed which is targeted in particular to overcoming the inherent weakness of the F-Trace method for large cases.

The main conclusion of this paper is that the off-line tuning performs particularly well in homogeneous cases (for example where different problems have similar characteristics and as such, underlying similar behaviour of the algorithm). This is intuitive, since in heterogeneous cases, an on-line tuning method adapts to the current problem state. The higher the number of parameters being tuned, the worse the performance observed with use of the on-line tuning method. Unlike popular belief, the heterogeneous instances were found to perform equally well when using either off-line or on-line tuning methods.

It was concluded by Pellegrini et al., that an implementation of a hybrid off-line on-line tuning method is suggestible, with perhaps the off-line tuned parameter set given to the on-line tuning during the search.

## 12.6 Measuring exploration and stagnation

Since research in the area of each aspect in the ACO field is too wide and too numerous to be studied in detail, the way in which stagnation or exploration is measures is described here, in-part, as a summary from the most recent book on ACO by Dorigo and Stützle [5] together with other sources [89, 90].

**Standard deviation:** Measuring stagnation/exploration can be done through calculation of the standard deviation of tour lengths constructed by the ants after each iteration, where obviously a standard deviation of 0 would indicate that all ants construct the same tours. The standard deviation relies on the absolute values of the tour length and so normally a variation coefficient is used, which is the standard deviation divided by the average solution quality (average tour length in the case of the TSP). This is often favourable since it is independent of scale.

**Distance between tours:** The distance between tours is a better indication, where in the TSP problem, this is to measure the number of arcs contained in one tour but not in the other. A decrease in the average distance between the ants tours indicates that

preferred paths are appearing and if the average distance becomes zero then stagnation has occurred (this measure is however computationally expensive).

**$\lambda$-branching factor:** Another method is called the $\lambda$-branching factor ($\lambda BF$) $\bar{\lambda}$, which measures the distribution of pheromone trails directly. This method was indented specifically to deduce a restart point with use of the MMAS algorithm, however has been commonly used amongst many algorithms in the literature. The $\lambda$-branching factor is given by the number of arcs incident to $i$ that have a pheromone trail value:

$$\tau_{ij} \geq \tau^i_{min} + \lambda(\tau^i_{max} - \tau^i_{min}) \tag{12.32}$$

where $\tau^i_{max}$ and $\tau^i_{\min}$ are the maximum and minimum values on the pheromone trail values on arcs incident to city $i$. $\lambda$ ranges over the interval [0,1] and $\lambda$-branching factor ranges over the interval [2,n-1], where $n$ is the number of construction nodes in the graph (number of cities in the TSP). This gives indication of the size of the search space being explored (if $\bar{\lambda} = 3$ then only 3 arcs on average have high probability of being explored). In TSP, the minimal $\bar{\lambda}$ is 2, since there must be at least two arcs used by the ants to reach and leave each city. This measures disadvantage is in having to choose the parameter $\lambda$, however it is a popular method in which to investigate the convergence of the algorithm. This convergence measure is used to determine the point at which to re-initialise the pheromones in $\mathcal{MMAS}$[66, 67, 89]. Another method considered by Favaretto et al. [89] is to measure exploration by the number of clusters of solutions visited.

**Direct measure of exploration:** Measurement of exploration is defined and studied in detail by Pellegrini et al. [91] and was intented to being independent on the specific procedure considered. The underlying idea is to group together similar solutions (characterised by the number of similar edges with one another) and that the exploration is then defined as the number of these clusters built. The method as described in [91] is, as combining at each step the two closest solutions to form a cluster and that the distance between this cluster and others is called the *maximum distance*. This procedure is then said to stop when the distance between the two closest clusters is greater than a predefined threshold $\epsilon$. A notable result of this investigation is that conclusions can be drawn irrespective of the chosen value of $\epsilon$ as long as exploration does not equal extreme values (1 or the number equal to the number of objective function evaluations).

**Similarity ratio:** This measure is used by Ke et al. [90] for the purpose of determining the ants ability to explore space and determine the effect of $P_{best}$ on the lower bound pheromone definition as defined by Stützle and Hoos [66]. It measures the amount of diversification directly by measuring the difference between ants solutions, similar to the distance between tours. This measure is said to be taken from EA and is defined as:

$$\frac{\sum_{j=1}^{n}\left(\sum_{i=1}^{n_a} s_j^i \cdot \left(\sum_{i=1}^{n_a} s_j^i - 1\right)\right)}{(n_a - 1) \cdot \sum_{j=1}^{n}\sum_{i=1}^{n_a} s_j^i} \tag{12.33}$$

where $s_j^i$ is the $j^{th}$ element of solution $s^i$ which is constructed by the $i^{th}$ ant and $n_a$ is the number of ants. Where all solutions are the same, the ratio will equal one, if all solutions differ then the ratio will be zero.

**Re-sampling ration:** Another measure taken from [90], is for the purpose of 'measuring how effective the algorithm is in sampling the search space'. This is done by measuring the number of unique solutions which are generated, which as one can imagine, may be memory intensive. This is then mathematically described as follows:

$$\frac{(TotalNum - DiffNum)}{TotalNum} \tag{12.34}$$

where $TotalNum$ is the number of solutions generated, $DiffNum$ is then the number of unique solutions generated. This ratio will equal zero if no duplicated solutions are generated, however, as the number of solutions generated by the ants become ever more similar, this ratio will tend to one, where a value of 1 will indicate total stagnation, where all ants construct the same tour (i.e. no new solutions not yet visited are generated).

**Fitness-distance correlation (FDC):** Another aspect to consider is the properties of the problem. One important aspect is whether better solutions tend to be found next to good solutions (that is an investigation of solution quality with distance from a very good or optimum solution). This is called fitness-distance correlation (FDC) analysis and gives indication as to the applicability (how well the ACO is likely to perform) for the problem in question. This correlation is described by Stützle and Hoos [66] as defined by:

$$\rho(F, D) = \frac{\mathbf{Cov(F, D)}}{\sqrt{\mathbf{Var(F)}} \cdot \sqrt{\mathbf{Var(D)}}} \tag{12.35}$$

where $\mathrm{Cov}(F, D)$ is the covariance between the random variables $F$ and $D$, which describe probabilistically the fitness and the distance of the local optima to the global optimum and *Var* is the variance. This method is commonly used to study the effectiveness of adaptive algorithms and in their design. Through this type of analysis, it is known that an enhanced search around a superior solution often results in a better final solution found amongst its neighbourhood. This is true of many problems, with TSP, QAP and set covering problem being amongst them!

**Various other methods:** Another method is to consider the average entropy:

$$\bar{\varepsilon_i} = \sum_{i=1}^{n} \frac{\varepsilon_i}{n} \tag{12.36}$$

The selection probabilities at each node are given by:

$$\varepsilon_i = -\sum_{j=1}^{l} p_{ij} log\, p_{ij} \tag{12.37}$$

where $l$ is the number of possible choices.

Yet another choice for measuring stagnation is given by the formula (which tends to zero as the algorithm moves toward stagnation):

$$\frac{\sum_{\tau ij \in T} min(\tau_{max} - \tau_{ij}, \tau_{ij} - \tau_{min})}{n^2} \tag{12.38}$$

## 12.7 Comparison of the three algorithms under interest

With respect to the TSP, there have been a number of algorithms which were developed in tackling this problem (extensions and improvements over the original AS) as described by Blum [12]. These include the 'Elitist AS (EAS)', 'Rank-based AS (RAS)', 'MAX-MIN Ant System (MMAS)', 'Ant Colony System (ACS)' and 'Hyper-Cube Framework (HCF)'. The most successful and most theoretically developed of which are the MMAS and ACS algorithms which are both at the forefront of continual research. Features are to be described in detail so as to conclude on which algorithm is better suited for in the application to other problem types. Additionally the population based ACO (PACO) is to be discussed, as an underdeveloped yet highly suggestible algorithm available in the literature.

When run-time is not of concern it is said that the best algorithms are the tour-merging approaches (TSP-specific heuristic) [5] (p.151) and the iterated version of Helsgaun's Lin-Kerninghan variant. Exact methods are said to show impressive results. Though the ACO family does not appear to reach state-of-the-art performance on the TSP problem, it does however for particular cases of the TSP, and for the quadratic assignment problem (QAP) etc.

Consideration of the most popular algorithms are made: concerning the ACS algorithm, it is known for being a greedy algorithm that applies a local pheromone update as described in [5, 29, 64]. Both ACS and MMAS use trail limits, though they are defined implicitly in the former and explicitly in the later. ACS was also the first algorithm to make use of candidate lists, however, these have been implemented within MMAS also [67] and indeed almost variants, being a standard method for speedup.

A summary comparison of the two algorithms is now made based on considerations such as their performance, parameter sensitivity and such difficulty for application to different problems and finally parallel implementation.

ACS is likely to outperform MMAS for shorter search times since it is a greedier algorithm as previously discussed and this is still likely to be the case as described by Pellegrini and Ellero [92]. It should be noted that a direct comparison between the two considered leading algorithms ACS and MMAS is subject to scrutiny, since the MMAS algorithm was designed for relatively long computation times ($10000 \cdot n$) which as discussed by Maur et al. [68] has likely influence from the first international contest on evolutionary optimisation [93]. As already discussed, this means that MMAS has a relatively long exploration phase with then a transition to a strong exploitation phase with the set-up given in [66]. MMAS is said to perform better on symmetric (TSP) instances [66], while giving similar results to ACS in the case of ATSP cases. With regard to other leading algorithms, the 'pheromone trail smoothing' (PTS) method described by Stützle and Hoos [66] together with lower influence of heuristic information result in both $AS_{rank}$ and $AS_e$ catching up with the performance of MMAS, except $AS_{rank}$ on ATSP instances and reach the solution quality of ACS on symmetric TSPs but still worse on ATSP instances. The PTS method helps to improve the performance of the MMAS slightly, but considerably improves $AS_{rank}$ and $AS_e$. It is conjectured that ACS concentrates its search too heavily around the global best solution, making it the most aggressive of the AS extensions. For this reason however, ACS is known for providing the better quality solutions when shorter computational times are given [5].

Amongst the literature there is a common theme, in that the two most leading algo-

rithms are MMAS and ACS, however many different authors disagree as to the better of the two. The second consideration to make is that of sensitivity to changes during run-time of the variables, where MMAS is found to be highly sensitive and ACS to be rather insensitive [68, 84, 89, 92]. This makes MMAS a much more difficult algorithm in which to implement with various problem types but also gives a greater control of the search. Another way in which to look at this, is to consider that MMAS is much more open to alteration to problem specific needs but similarly ACS can be considered as robust due to its results with very insensitive parameters. Since ACS is found to perform best with the number of ants set to 10 and that this value to be rather independent of the instance size [64], this algorithm appears to be good for its scalability and is often the reasoning for it being chosen by some authors in the literature, for example Lalbakhsh et al. [94], who chose ACS on these very grounds.

With regards to parallelism of the algorithm, many attempts have been made in which to make MMAS work on parallel, with a recent attempt found in the literature by Bai et al. [95], however ACS is not excluded from this area of research and so comparison between the two appears to be premature. What is clear is that MMAS updates at the end of each iteration and that due to ACS's local pheromone updates, parallelism is not straight forward with respect to offloading CPU load to multiple CPU's (the behaviour of the algorithm will differ since updates occur during solution construction).

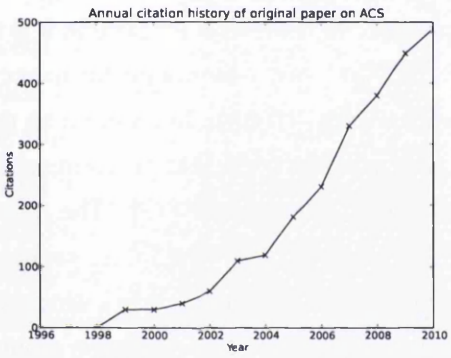
The main area of sensitivity for performance between the two ACO algorithms appears to be the chosen local search implementation as described by Dorigo et al. [96] but also to the parameter settings of the algorithm.

To summarise, ACS is described as a greedier algorithm and as such performs better than MMAS over shorter computational times using the default parameter values, but given enough time, MMAS is expected to yield better results at the cost of a less robust algorithm (in terms of the sensitivity to parameter adjustment). Incorporation of the ACS action-choice rule however appears a logical decision in the attempt to control the greediness of the MMAS algorithm (in keeping with a number of authors in the literature as already discussed).

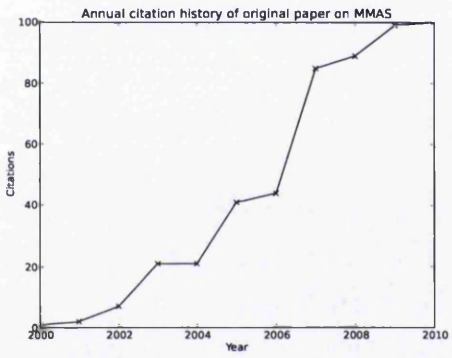
A particularly insightful algorithmic idea is that of the population-based ant colony algorithm by Guntsch [76], to which recently a technical report detailing the performance of this algorithm on the TSP and QAP was made by Oliveira et al. [77]. In this paper the PACO is shown to be competitive with MMAS and indicate better performance than MMAS in short searches. The PACO algorithm has success in multi-

objective optimisation also, with one of the leading algorithms called the crowding population-based ant colony optimisation algorithm (CPACO) [72, 73, 79], though other authors do not consider it amongst the main methods in tackling multi-objective problems. This is perhaps because it drifts somewhat away from the standard ACO framework. Other interesting areas tackled with respect to this algorithm have been in niching [69].

To further emphasise the research based on the two most popular algorithms, the following plots are given to demonstrate the general research trend (fig. 12.5). Information regarding PACO however is not included, since there is not enough citations to warrant a fair comparison. It is needless to say that PACO is a novel approach which distinguishes itself from the other two methods and for this reason is subject to interest here.




(a) Citation history of the original paper on ACS [96], generated using data from google scholar.




(b) Citation history of the most cited paper on MMAS[66], generated using data from google scholar.

Figure 12.5: Citation history of the two most cited papers of the two most popular ACO algorithms.

# Chapter 13

# Numerical testing and verification

## 13.1 The choosing of the PACO algorithm

The primary reason for the choosing of the PACO algorithm for research, is that it is the only algorithm to have tackled niching problems and has shown strong performance in dynamic and multi-obective problems as well (see section 12.4.4). In addition to this, to restate that mentioned in this previous section, indications are that performance of the algorithm is at least on par with other leading algorithms in use [77]. The overall intention of this algorithm and its choosing, is in its application to scheduling problems (or more specifically job-shop-scheduling problems). This algorithm allows successful methods from EAs to be migrated over to the ACO domain with it population archive. The reader is again referred to section 12.4.4 for further details on this matter.

Scheduling problems are where resources are allocated to tasks over time and are said to be central to production and manufacturing. Put in another way, scheduling is where we have $n$ jobs which are required to be carried out on $m$ machines, where each job consists of a set of operations to be performed on one or more of these machines. The input to these problems are then processing times and often additional set-up times, release dates and due dates of jobs, job importance and precedence constraints amongst jobs, as described in [97]. To clarify, an operation is the term used when jobs require to be processed on more than one machine. That is, a job completed on one machine is a single operation completed, while the other operations on the other machines (the remainder of this single job) remain.

A gantt chart is often very useful to understand such problems as shown in Fig. 13.1. Here, $q$ denotes the completion time, while $p$ denotes the processing time of jobs. $M_j$ then correcsponds to the machine, where $j$ is the machine index which corresponds

to the number of jobs. $O_{i,j}$ then refers to operation of job $i$ on machine $j$. In this figure,

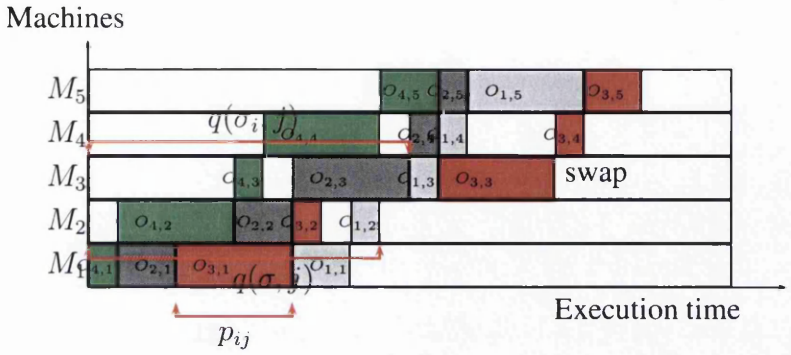



Figure 13.1: Representation of the different times to determine objectives recursively.

an example solution, or scheduling of operations is observed.

The job-shop-scheduling problem has had extensive research, indicating its multi-modality. For this reason, an ACO specifically designed to take advantage of a population archive to force diversity is ideal. To highlight the issue with this problem, see the following illustration by Ikeda and Kobayashi [98] (Fig. 13.2). This is what is called the UV-structure hypothesis as defined in [98], with the investigation to genetic algorithms. Class1, was defined as where the likelihood of early solutions to be of high quality (which contains the optimum) is less than the much wider more suitably shaped local-valley. This means that individuals are less likely to find a good solution in the valley containing the global optimum. Class2, is where the number of solutions in any one particular valley may depend on the structure of this valley (perhaps it is narrow, making individuals less inclined to search in its direction). Finally, class3 was defined as, where the shape of the valley itself may result in the evolution of the solutions at different rates amongst differing valleys.

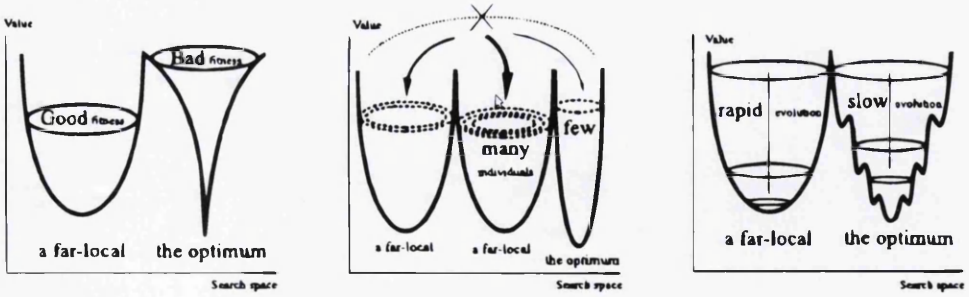



Figure 13.2: The 'intuitive' representation of the UV-structure hypothesis solution space taken from [98]. The left (class1), the centre (class2) and the right (class3).

For a detailed review of shop scheduling problems and of the issues and algorithmic approaches to this problem, the reader is referred to Appendix B.2.6.2.

## 13.2 Experimental

In order to implement the PACO algorithm, two of the most successful algorithms applied to the TSP are replicated. This allows the verification of those mechanisms which are common to the PACO algorithm, as not enough information is available in the literature regarding the results and performance of this algorithm for verification. In order to verify the two most successful implementations of ACO, a comparison is made with the publicly available code by Stützle [99], referred from now on as $ACOTSP_{v1.0}$. The code written here for which investigations are to take place will be called the In-house ant colony optimiser ($IHACO_{alg}$) where $alg = \{ACS, MMAS, PACO\}$. Common terms to be used amongst this section are iteration and cycle which are to be used interchangeably. Also the phrase, *total information* is used here which is commonly coined in the literature ([5]) as the combined heuristic and pheromone information to the power of $\alpha$ and $\beta$ respectively.

### 13.2.1 Compiler and set-up

$IHACO_{alg}$ is written in FORTRAN, with either Ifort or Gfortran compilers. For consistency, the Intel compiler will be used from here-on, however the code is written and compile tested to ensure successful compilation with Gfortran, since this compiler is opensource under the GNU Compiler Collection (GCC). The version of the Intel compiler used here is as follows:

*Version 12.0.2.137 Build 20110112*

(All runs are compiled on a 1GB ram, 2.1GHz Intel system).

With regard to $ACOTSP_{v1.0}$, it is written in ANSI-C, again compiled under the GCC. The set-up for algorithmic verification, considers the parameter shown in table 13.1 for both $MMAS$ and $ACS$ algorithms, with those which are particular to $ACS$ indicated in brackets.

It should be noted that these represent not the optimised values, but those which are suitable and simplify the comparison. For guidance to optimised parameters, the reader is referred to [5]. Regarding the maximum number of solutions generated (termination

| | | |
|---|---|---|
| Runs | 100 | trial runs |
| $max_{tours}$ | $2500 * n$ [66] | number of tour constructions |
| $f^{gb}$ | 25 $(N/A)$ | frequency of global best update in MMAS |
| $\alpha$ | 1.0 | |
| $\beta$ | 2.0 | |
| $\lambda$ | 0.05 | average branching factor |
| $min_{branch}$ | 2.00001 | Threshold for $\lambda$ (1.00001 for $ACOTSP_{v1.0}$ due to normalisation) |
| $\rho$ | 0.02 (0.1) | |
| $Flag_{reinit}$ | Disabled | re-initialisation flag disabled |
| $q0$ | 0.0 ($q0 = 0.9$) | |
| $m$ | 25 ($m = 10$) | |
| $nn$ | 20 | length of candidate list |

Table 13.1: Defined parameters for the algorithmic comparison with description of those variable not previously defined. See section 12.4.4 for the necessary formulae for PACO.

measure), an alternative maximum CPU-time is instead used in cases were varying parameters will obviously cause dramatic differences of computation time per cycle (these include sections 14.7 to 14.9). These include parameters $m$, $nn$, $k$ and coupled parameter change $k\tau max$.

### 13.2.2 Problems used in this study

The three standard symmetric TSP problems are chosen here as used by a number of authors [64, 66], influenced by the First International Contest on Evolutionary Optimisation (1st ICEO) [93]. These problems include eil51, kroA100 and d198 from TSPLIB95 [100]. Since these are but standard problems, the reader is referred to appendix C for further information.

For the parameter study of PACO however, a more insightful custom made problem group is made using the TSP instance generator 'portgen' of the 8th DIMACS Implementation Challenge [101]. Two variations are considered: variation of instance size and instance node distribution. The later allows the investigation of instances with nodes that are clustered into groups. Uniformly distributed problems are indicated by 'portgen', while more clustered problems are named with 'portcgen'. The problem size is then indicated by the subsequent number following the distribution type.

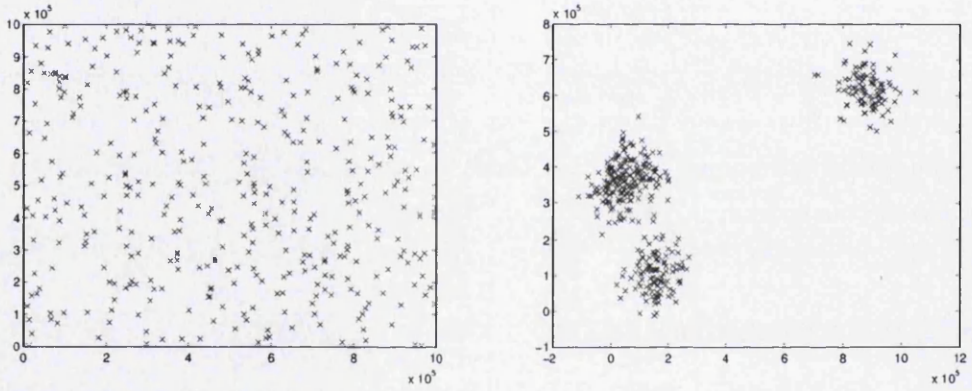
Since these problems are generated, no known optimum will be available. Instead,

the freely available Lin-Kernighan algorithm by Helsgaun [102] is applied to each of the problems and the resulting final best solution presented here as an assumed optimum. Details of these problems are as follows:

| NAME | portgen | portgen | portgen | portcgen | portcgen | portcgen |
|---|---|---|---|---|---|---|
| SEED | 284019776 | 262603184 | 298462752 | 284019776 | 262603184 | 298462 |
| TYPE | TSP | TSP | TSP | TSP | TSP | TSP |
| DIMENSION | 100 | 200 | 400 | 100 | 200 | 400 |
| EDGE_WEIGHT_TYPE | EUC_2D | EUC_2D | EUC_2D | EUC_2D | EUC_2D | EUC_2D |
| LHK_OPTIMUM | 7543551 | 10860490 | 14988108 | 3636921 | 4794600 | 734030 |

Table 13.2: Designed test problems for the ACO parameter study using portgen.

One problem of uniformly distributed nodes and one of clustered distribution is shown in fig. 13.3.



(a) portgen_N400 (b) portcgen_N400

Figure 13.3: Two problems generated with portgen, fig. 13.3a for a uniformly distributed problem and fig. 13.3b for a clustered problem.

The full set of problem plots can be found in appendix C.

### 13.2.3 Notable comments on $ACOTSP_{v1.0}$

Since verification with $ACOTSP_{v1.0}$ is only possible if it is well understood, the differences observed in this code from that reported in the literature are described here. The use of this algorithm, allows a detailed comparison of the behaviour of the algorithm with respect to $IHACO_{alg}$, which would otherwise be impossible with the data which is available in the literature.

Differences are listed here between the documentation of $ACOTSP_{v1.0}$ and the coding itself:

**1.** The use of logical statements within $ACOTSP_{v1.0}$, for example a $\leq$ argument, will cause a loop to continue until the last occurrence where the statement holds true (i.e. to which the statement is equal). On the other hand, a $<$ logical argument holds true up until the first occurrence, for example in the nearest-neighbour tour construction. This is important since the way in which arrays are ordered or indexed is not random but generally based on their closeness (nearest neighbours first). An erratic choosing of iteration best (*ibest*) is an apparent consequence of this, since the global best (gbest) solution is not updated when an improved solution is found, but rather, it is set on every iteration as the best solution of the current cycle if $\leq$ than the current gbest. Since more than one solution may have the same tourlength, this may result in an erratic solution change as the pheromone update bases its deposit on different equally good solutions with no clear amplification of any one of them (i.e. a change of gbest solution and then deposition even if the solution quality has not improved). An important point to clarify is that functions MAXVAL, MAXPOS, MINVAL and MINPOS (Fortran specific), utilise first occurrences and that on the problems investigated here, no significant difference in behaviour is observed with use of first or last occurrences.

**2.** The formulation of $\tau_{min}$ for $MMAS$ is different in $ACOTSP_{v1.0}$ in respect to [66]. To quote [66], limits are derived based on two assumptions: One assumption is that better solutions are found a short time before stagnation[1] occurs such that the probability of constructing the global-best tour is much greater than zero in a single iteration; secondly that the influence of the solution construction is much greater by the relative difference between $\tau_{min}$ and $\tau_{max}$ than the relative differences between the heuristic information.

The limits of the pheromone are said to be derived by Stützle and Hoos [66] in the following way:

With $P_{best}$ being the probability of constructing the best solution after $MMAS$ has converged [2] and $P_{dec}$ being the probability of choosing the solution components at each choice point [3] that belong to the best tour. The two are said to be related by the following

[1] stagnation as defined for $MMAS$ in [66], is the point where all ants follow the same path.

[2] convergence as described in [66], is where, for each choice point, one of the solutions corresponds to $\tau_{max}$ (its associated pheromone trail) while all other solutions correspond to $\tau_{min}$.

[3] choice point is a term taken from [66], which describes the point at which a decision is made (next

relation: $P_{dec}^{n} = P_{best}$, since an ant must make 'n' correct choices to construct the best solution (since there are 'n' solution construction steps). $P_{dec}$ is then calculated using the probabilistic choice rule by which an ant chooses its next solution component as shown in eq. (13.1).

$$P_{dec} = \frac{\tau_{max}}{\tau_{max} + (AVG - 1) * \tau_{min}} \tag{13.1}$$

where AVG is the average number of choices an ant has at each choice point. This is set as $n/2$ and $P_{best} = 0.05$. Re-arranging for $\tau_{min}$ gives eq. (13.2).

$$\tau_{min} = \frac{\tau_{max} * (1 - P_{dec})}{(AVG - 1) * P_{dec}} = \frac{\tau_{max} * (1 - \sqrt[n]{P_{best}})}{(AVG - 1) * \sqrt[n]{P_{best}}} \tag{13.2}$$

**3.** For the $ACOTSP_{v1.0}$ algorithm, the initialisation occurs with $\tau_{min}$ as defined in eq. (13.2), however without implementation of the local search, $\tau_{min}$ updates according to the following relation eq. (13.3), which differs from that defined in [66].

$$\tau_{min} = \frac{\tau_{max} * (1 - P_{dec})}{(\frac{nn_{ants}+1}{2}) * P_{dec}} \tag{13.3}$$

where $nn_{ants}$ is the number of nearest neighbour ants, which then takes into account that the average number of solutions that an ant can choose from no longer corresponds to the entire list, but restricted to the candidate list[4] of size $nn_{ants}$. Similarly, $P_{dec}$ is defined such that it takes into account the number of nodes in the problem, which is due to its inherent problem specific meaning. This is given by eq. (13.4).

$$P_{dec} = EXP(log(0.05)/n) \tag{13.4}$$

**4.** Regarding the (re)initialisation stages of the algorithm, each ant is initialised at a random node (city) irrespective of how many ants have been already initialised at this location. This corresponds to that described for the $MMAS$, however this differs from the suggested initialisation of at most one ant per city for the ACS as described by Dorigo and Gambardella [64]. This could be significant, as ACS implements a local pheromone update, which could then allow the amplification of components that would otherwise not be amplified.

solution component to be used).

[4]Candidate lists are where the nodes surrounding the current node are restricted to a reduced number, in order of closest to furthest.

The re-initialisation scheme used in $ACOTSP_{v1.0}$ also differs somewhat. Unlike that reported in [66], re-initialisation of pheromones where no local search is implemented is made in the following manner regarding $MMAS$: If more than 250 cycles have passed, where no improvement is observed (compared to the last re-initialisation) and the branching factor has reached a certain threshold (0.05), then the pheromone is re-initialised to $\tau_{max}$. Furthermore, every 25 ($f^{gb}$) cycles, the restart-best (*rbest*)[5] rather than the global best update is used in the pheromone update.

**5.** Further observations on the $ACOTSP_{v1.0}$ algorithm include its chosen random number generator (referenced as having origins in Numerical recipes C by Press et al. [103]). This is the one, first proposed by Lewis, Goodman and Miller in 1969, and is described in [103] to have survived the test of time. Within the in-house algorithm, the Keep It Simple Stupid (KISS) random number generator (RNG) is used with respects to the Gfortran compiler (an algorithm written by George Marsaglia)[104].

**6.** The sorting algorithm used by $ACOTSP_{v1.0}$ for the purpose of determining nearest neighbour lists appear to be either the straight insertion or shell method (not described). Such methods can again be found in Numerical Recipes in C[103]. A straight insertion sorting algorithm is also implemented here to sort the nearest neighbour candidate lists (Numerical recipes in Fortran 90 by Press et al. [10]).

**7.** Finally, the heuristic information is calculated in $ACOTSP_{v1.0}$ with the addition of a small constant to take into account cases in which a divide by zero would occur. This then gives a formulation for the heuristic information in TSP as:

$$heur_{i,j} = \frac{1.0}{distance_{i,j} + 0.1} \tag{13.5}$$

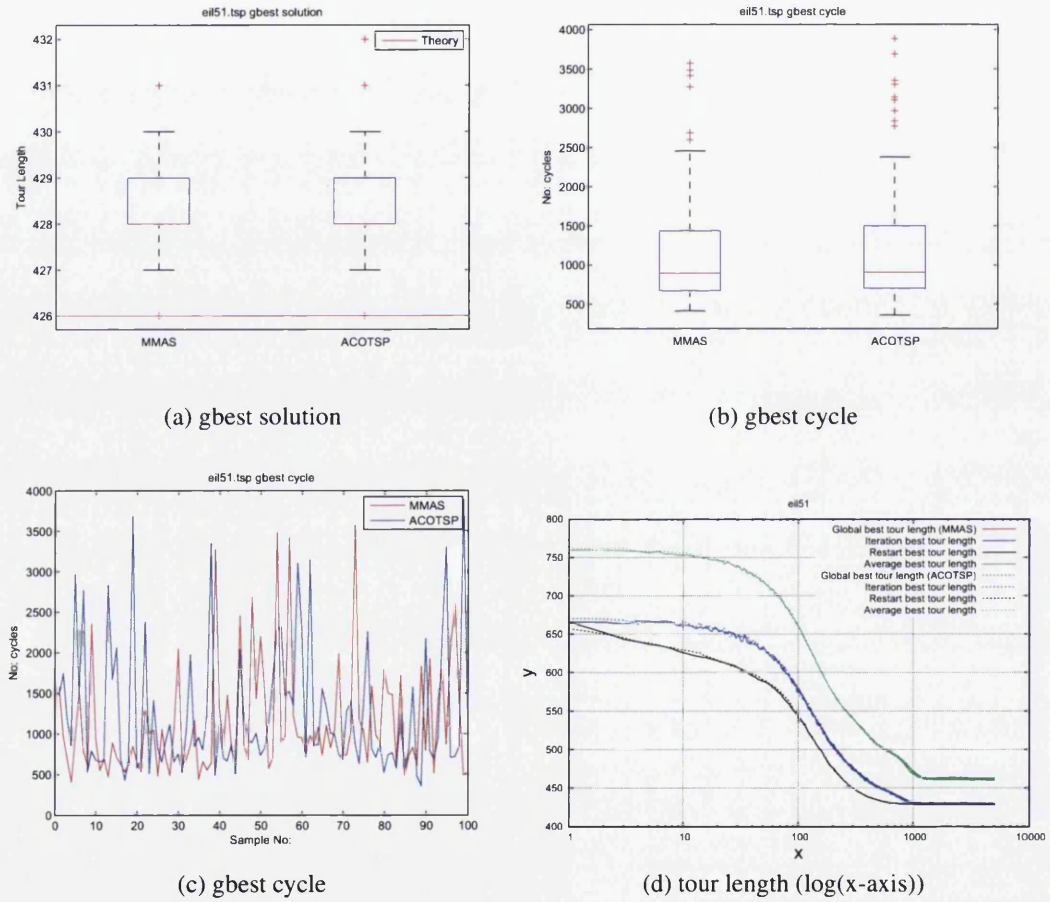
This should have little effect on the behaviour of the algorithm since this is significantly different in magnitude from a non-zero component, but a difference nether-less.

With regards to closing remarks, all differences reported here are implemented in the algorithm coded here in order to offer a fair verification of the algorithms.

[5]The restart-best solution is the gbest solution which is reset if a restart occurs. That is, the gbest solution since the most recent restart.

## 13.3 MMAS algorithmic comparison

For the purpose of this comparison, $IHACO_{MMAS}$ will simply be called $MMAS$ and $ACOTSP_{v1.0,MMAS}$, as simply $ACOTSP$. The chosen problems for comparison are those that are provided with the $ACOTSP_{v1.0}$ software. Specifically, 'eil51', 'kroa100' and 'd198'. These problems are chosen since they are not too large and verification of the algorithm at this stage is the only concern. Upon verification of the two algorithms, a range of problems which vary in both scale but also in homogeneity are chosen. It should be noted that the problems which are included as part of the $ACOTSP_{v1.0}$ software, originate from [100].




(a) gbest solution

(b) gbest cycle

(c) gbest cycle

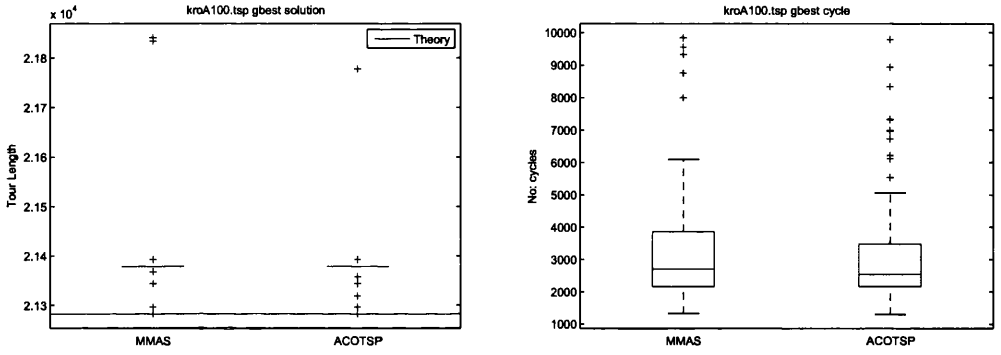
(d) tour length (log(x-axis))

Figure 13.4: TSP problem eil51, mean results over 100 samples.

It can be clearly observed that the same behaviour is exhibited in both algorithms for eil51 (fig. 13.4). Figure 13.4d clearly shows how the average tourlength of the population of ants is maintained at a level significantly higher than the gbest solution

and this level being equal for both algorithms. This diversity remains high due to the explicit pheromone limits in MMAS. The result of these limits is that there is a non-zero probability of any solution component being chosen. However, the iteration-best solution, converges to the gbest solution after some time. This occurs at the same average cycle in both algorithms. Referring to the final solution quality and the number of cycles required to achieve it (as shown in figs. 13.7a and 13.7b), both parameters are comparable from the $MMAS$ and $ACOTSP$ results. It should be noted that there are some samples which deviate significantly from the mean, but this is due to the algorithm being a heuristic, learning the search-space with stochastic properties.

Results for problem kroA100.tsp are shown in fig. 13.5



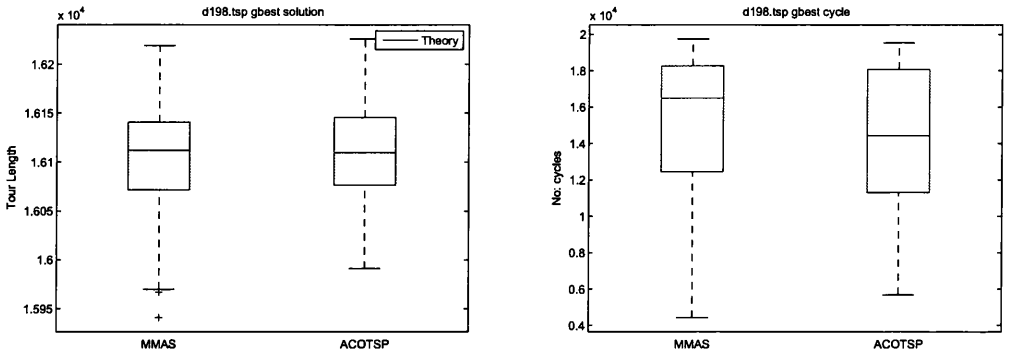

(a) gbest solution

(b) gbest cycle

Figure 13.5: TSP problem kroA100, mean results over 100 samples.

A significant difference with eil51, appears to be with its added complexity (size of the problem), the final solution found is rather un-refined. Tour lengths are found which are significantly far away from the considered extreme tour solutions (extreme points according to a box-plot). The interquartile range (IQR) is then significantly small compared to the total range of data across 100 samples, as is the $25^{th}$ and $75^{th}$ percentile (upper and lower quartile (Q1 & Q3), respectively). The IQR signifies that 50% of the gbest solutions found are at a sub-optimal tour lengths and in fact, since Q1 and Q2 are located at the same region, the vast majority of final solutions are found to be located at the same sub-optimal region. The number of cycles required as shown in fig. D.5b is in agreement for the two algorithms, as is history of the solutions found during the search.

Finally, the results for problem d198 are shown in fig. 13.6. MMAS is described as being under a disadvantage with very large problems, (larger than 100 of nodes




(a) gbest solution

(b) gbest cycle

Figure 13.6: TSP problemd198, mean results over 100 samples.

[5, 66]). For this reason, it is not unexpected that the mean number of iterations required to achieve the best found solution to be in conflict (to an extent) between the two algorithms. On no occasion is the true global minimum found by either algorithm. This accounts for the limitation and indeed erratic finding of the best solution found during their search. It should be noted that the search itself exhibits the same characteristics in both algorithms as shown in fig. 13.6. It should be re-emphasised that ACO does not become competitive until it is hybridised with a local search[5]. However, to further analyse this problem requires a further depth of understanding of the algorithm search.

The reader is referred to appendix D.1 for a full set of plots relating to this section.

## 13.4 ACS algorithmic comparison

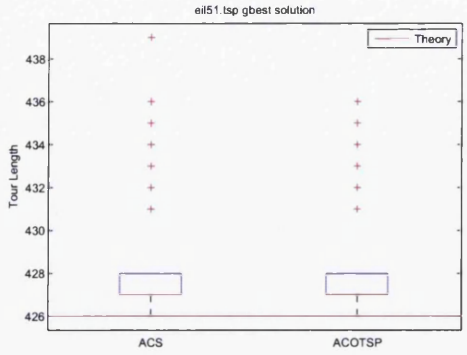
For the purpose of this comparison, $IHACO_{ACS}$ will simply be called $ACS$ and $ACOTSP_{v1.0,ACS}$, as simply $ACOTSP$. Again, the same three symmetric TSPs are chosen in order to conduct a behavioural comparison between $IHACO_{ACS\,v0.1}$ and $ACOTSP_{v1.0}$. The results of which are shown in fig. 13.7. It is proficient to mention that behaviour between the two algorithms is the same and where it is may be suggested to deviate (fig. 13.7e,13.7f), the curve shape relates between the two algorithms, indicating that the $IHACO_{ACS\,v0.1}$ algorithm is somewhat slightly biased towards better solutions on this particular occasion. This is suggested here as being due to the differing sorting functions used in both algorithms. The node listings within the candidate list result in some biasing within the problem file if for example the first occurrence of a

node at a minimum distance is found rather than the last occurrence (see section 13.2.3) for further details.

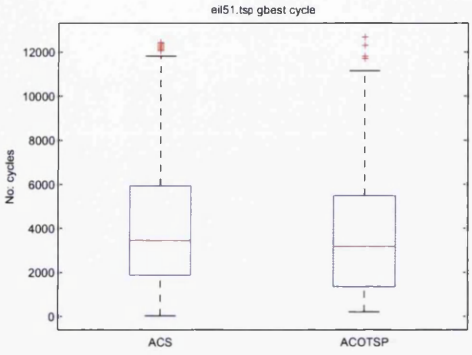
Again, the reader is referred to appendix D.2 for a full set of plots relating to this section.

## 13.5 Summary

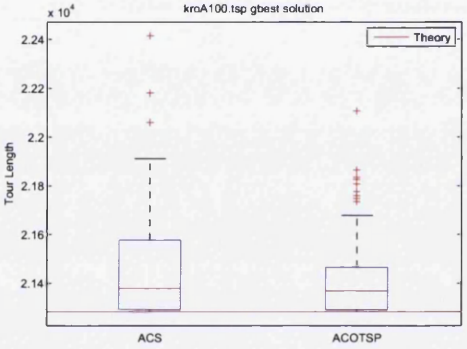
Now that verification has been made with only slight discrepancies observed between the two algorithms, those components which are common to the $PACO$ algorithm are now considered to be correctly implemented, such as the pseudo-random action choice rule of $ACS$ or the pheromone limits of the $MMAS$.



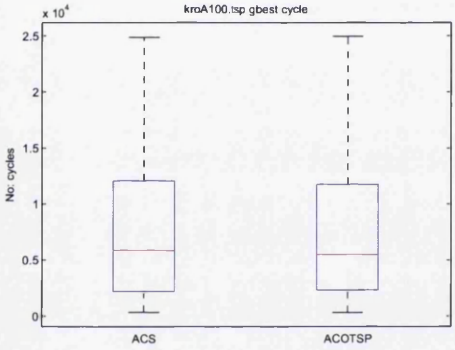

(a) gbest solution



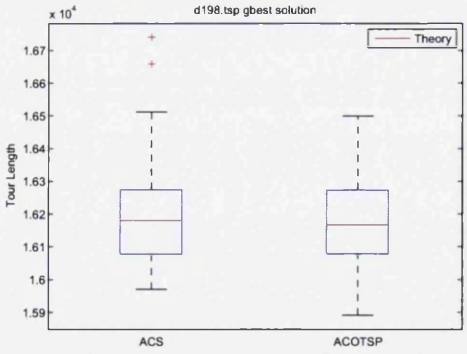

(b) gbest cycle



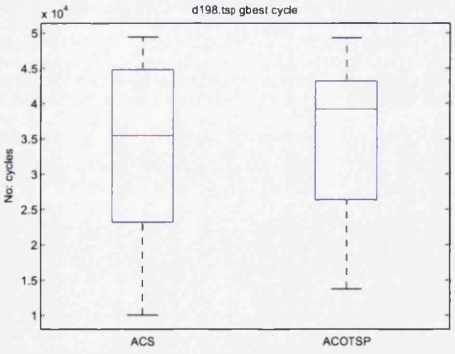

(c) gbest solution




(d) gbest cycle




(e) gbest solution




(f) gbest cycle

Figure 13.7: All three problems for ACS verification, mean results over 100 samples.

# Chapter 14

# PACO parameter study

## 14.1 Experimental

The parameter analysis of PACO is made, so as to gain an insight into this chosen algorithm. For this study, the following parameter ranges were chosen as a result of the literature review of this algorithm [5, 70, 71, 74, 76, 77] (see section 12.4.4).

| Param | Range |
|---|---|
| $\alpha$ | { 0.0, 0.1, 0.5, 1.0, 1.5, 3.0, 5.0, 10.0 } |
| $\beta$ | { 0.01, 0.1, 0.25, 0.5, 1.0, 2.0, 5.0, 10.0 } |
| $q_0$ | { 0.0, 0.25, 0.5, 0.75, 0.9, 0.95, 1.0 } |
| $w_e$ | { 0.0, 0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 1.0 } |
| $\tau_{max}$ | { 1.0, 2.0, 5.0, 10.0 } |
| $nn$ | { 2, 5, 10, $n-1$ } |
| $k$ | { 1, 2, 3, 5, 10, 25 } |
| $m$ | { 1, 5, 10, 25, 50, 100, 300 } |

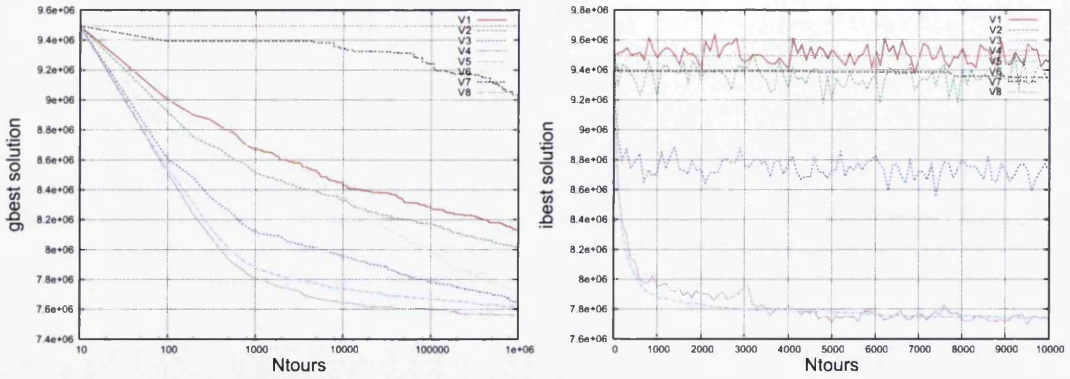
Table 14.1: Parameter ranges to consider for the parameter study of PACO.

It should be noted that the only plots printed here are of pge100 and pgc100 with respect to its gbest and ibest solutions, as a function of the iteration number. However, a full set of plots regarding all problems can be found in appendix E.

## 14.2 PACO parameters analysis ($\alpha$)

It should be noted that the parameter $\alpha$ (and indeed $\beta$) has a major influence on the running/performance of the algorithm and is due to the following relationship between the probabilities associated to two edges $i$ and $j$ at a particular time:

$$\frac{P_i}{P_j} = \left[\frac{\tau_i}{\tau_j}\right]^{\alpha} \left[\frac{\eta_i}{\eta_j}\right]^{\beta} \tag{14.1}$$




(a) (b)

Figure 14.1: Parameter study: $\alpha$

In fact, it is shown in fig. 14.1 that the algorithm is in fact highly sensitive to it. The suggested parameter which achieves best performance appears to be where $\alpha = 1.0$ or $\alpha = 1.5$ depending on the distribution of the problem. For the more clustered problems, it is clearly observed that a slightly higher $\alpha$ results in a better performance and this can be reasoned by the fact that the following formulae (for the most part) determines the solution generated:

$$\frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}} \text{ if } j \in N_i^k \tag{14.2}$$

Considering two situations: one where a clustered distribution is apparent and another, a uniform distribution, which is in agreement with observations by Pellegrini et al. [85] for the $MMAS$ algorithm. The parameter $\alpha$ determines the relative amplification of the difference of past experience between nodes (arcs) compared to the heuristic information. In the more uniform distribution of cities, the heuristic values on arcs of neighbours are likely to be very close to one another. A too high value of $\alpha$ would result in too strong an emphasis being given to past experience. A higher emphasis on past experience is however necessary where the heuristic values are distributed in the case of a clustered problem, since the differences between possible neighbours maybe very large and so the possibility of choosing some arcs (which maybe be in the $s^{gb}$) are greatly reduced. i.e. the strongly suggested heuristic values are more inclined to force

the search around these arcs which may or may not be that close to the real optimum solution.

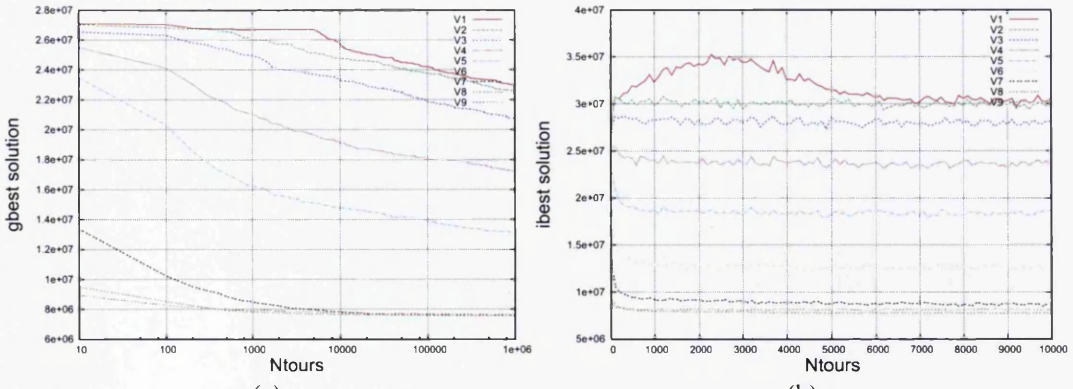
Other observations are made here, where a small $\alpha$ results in an algorithm which behaves much like a stochastic multi-greedy algorithm (no memory influencing the solution construction process). As $\alpha$ increases, an ever stronger reliance on the previous solutions is made with very a high $\alpha$, eventually resulting in the case where the same solution is constructed repeatedly (indicated by a very low tour standard deviation for $\alpha > 3.0$).

One very important observation to be made here is that with $\alpha = 0.0$ (no memory), the global best solution is stagnated but not the ibest solution since a stochastic solution construction remains, which can theoretically generate any solution (given enough time). A smaller distance between solutions stored within the archive and also for those generated is found with higher $\alpha$ until $\alpha = 1.5$. For $\alpha > 1.5$, the distance increases again, signifying that again there is too much emphasis on previous experience (exploitation). The likely possible solutions to be generated are those locally best (i.e. exploration is lost, also indicated by a decrease of branching factor for increasing $\alpha$).

A trade-off is apparent here: the emphasis between past experience and a tendency to follow the heuristically suggested solution paths.

## 14.3 PACO parameters analysis ($\beta$)

Similarly for $\beta$, this parameter appears to be highly sensitive to the performance of the algorithm and perhaps more so than $\alpha$ as shown in fig. 14.2.




(a) (b)

Figure 14.2: Parameter study: $\beta$

With $\beta = 0.0$, pheromone values are still likely to change since solutions generated are initially made at random (since the same pheromone $\tau_0$ will be present on all arcs). Deposition by the elitist ant is lower than the population members (on this particular occasion as with $\alpha$) and so even with no heuristic information, a slow convergence to a solution is apparent with no heuristic information. Unlike the situation with $\alpha$, there does not appear to be a clear distinction between clustered problems and uniformly distributed problems over those investigated here. A value of $\beta = 5.0$, which agrees with that in the literature appears to be 'best', closely followed by $\beta = 10$, which appears to have a much faster convergence but worse final solution (indicating a too strong emphasis on the heuristic differences between arcs compared to the past experience).

It should be noted that with $\beta = 0.0$, there is a strange increase in the $s^{ib}$ and $s^{av}$ solutions generated. However, it was discovered that this 'anomaly' results from the fact that a nearest neighbour strategy is used here to limit and consequently re-order the matrices. Since a candidate list of $n - 1$ is used, this was not initially considered, but since $\beta = 0.0$ and a candidate list is used, this still orders the matrices according to a nearest first procedure. This results in the choosing of components that are closer together, even though heuristic information is not used directly i.e. a search is made for the maximum total information, which itself is in order of nearest neighbour first. With $\beta = 0.0$, when the pseudo action-choice rule is triggered ($q \leq q_0$) arcs of equal deposit (length) can no longer be distinguished since the following argument is used ($\geq$). Since a population of 3 is used, 3 initially random deposits are made and after which, arcs are chosen amongst the 3 possible paths (though most likely two) which are closest in neighbour. As soon as $\beta$ is increased, arcs of equal length can be distinguished from one another and this increase in tourlength does not appear. This 'hump' appearance is then a result of the distribution of solutions found within the initial period of the search.

## 14.4 PACO parameters analysis ($q_0$)

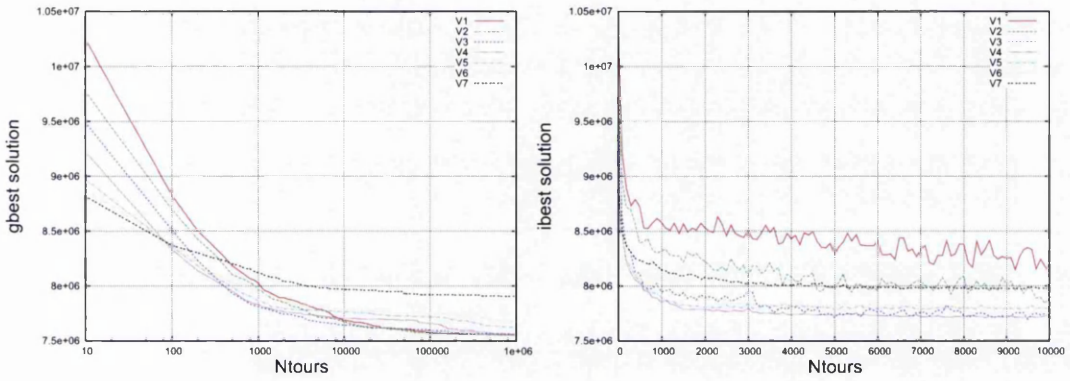
This parameter is important in whether the following rule is used or not:

$$\arg \max_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta \quad \text{if } q \leq q_0 \tag{14.3}$$

The smaller the value, the less likely that the deterministic choice of arcs with highest *total information*[1] are to be used and the less exploitation behaviour is present. It is

[1]total information is described here as the quantity $[\tau_{il}]^\alpha [\eta_{il}]^\beta$ on arcs.

suggested here that a lower $q_0$ (or in fact a value of zero) would results in the best solution quality in the long run, however, it would require a greatly increased computational expense in order to achieve similar levels of solution quality. For this reason, a greater than 0 value for $q_0$ is suggested. This is apparent from fig. 14.3.




(a) (b)

Figure 14.3: Parameter study: $q_0$

A value of $q_0 > 5.0$ then results in the emphasis becoming too strong on this total information. $q_0$ can be seen as then a speed-up parameter for the algorithm, giving emphasis on the level of exploitation during the search. On the extreme case of $q_0 = 1.0$, the very highest arcs of total information are chosen every time. It should be noted however that the algorithm still shows a slow convergent behaviour since a random initial starting point for the ants results in slightly different solutions constructed. This can be shown by the following illustration (fig. 14.4).

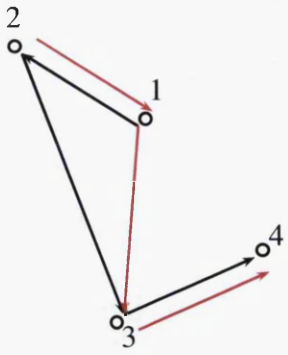



Figure 14.4: Illustration of the numerous solutions generated at initialisation due to random initial placement. Red line for one possible tour and black line for another.

As shown in the above illustration, even though the same total information in both

tour constructions is apparent, a random initialisation can result in varying tourlengths. This explains how the solution may improve over time even when arcs are chosen deterministically during the solution construction phase ($q_0 = 1.0$). This also then allows alteration to the solution construction through deposition on different arcs. It is noticed that the updates to gbest are also rather infrequent for this case. One important point to emphasise here is that the distance measures and average branching factor reduce as a function of increasing $q_0$. This again re-emphasises the fact that increasing $q_0$ produces a greedier algorithm. As a final note, it is noticed that the algorithm is not as sensitive to this parameter to others including $\alpha$ and $\beta$.

## 14.5 PACO parameters analysis ($w_e$)

From here-in, the algorithms pheromone update has been modified to encompass the original formulation. That is, $\Delta = w_e(\tau_{max} - \tau_0)/k$ is replaced with $\Delta_e = (1.0 - w_e)(\tau_{max} - \tau_0)/k_e$. A mistake which does not matter in the previous parameters studied, but now does, since the weighting of the elitist deposit is adjusted. $w_e$ signifies the amplification of the elitist deposit, with a weight of $w_e = 0.0$ signifying that no deposit occurs when a new global best solution is found. In the case of the study of other parameters, a zero weighting is a disabling of the elitist ant and so the global best solution updates by the ibest solution.

This parameter determines the emphasis of the elitist ant through its quantity of deposition compared to the other solutions. The deposition of the elitist is based on the following relation:

$$\tau_{ij} = \tau_{ij} + \Delta \text{ if normal member deposit/removal} \tag{14.4}$$

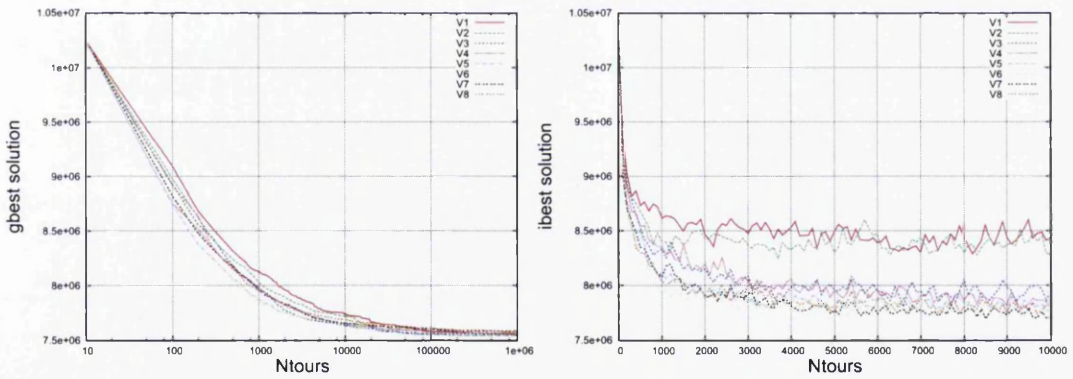
$$\tau_{ij} = \tau_{ij} + \Delta_e \text{ if elitist deposit/removal} \tag{14.5}$$

where $\Delta = w_e(\tau_{max} - \tau_0)/k$ and $\Delta_e = (1.0 - w_e)(\tau_{max} - \tau_0)/k_e$

A description of the function of this parameter from its formulation would be, that it controls the exploitation of the algorithm by emphasising the gbest solution. However, the parameter $q_0$ also controls the level of exploitation, but through a different means. $q_0$ gives control to this behaviour by statistically choosing the 'best so far' overall arcs which may or may not be due to the elitist member i.e. the first emphasises the effect of the globally best solution found so far whilst the later causes the deterministically most suggestible solution which may or may not agree with the elitist (global best) solution.

Looking at the distance between solutions within the archive and indeed the solutions generated, it becomes clear that increasing the weight on the elitist member of the population results in a decrease in the distance between members of the archive. This indicates that the ibest solutions are closer together (a convergence toward a better solution and a search around this solution). Similarly, of the solutions generated by the ants, the higher the weight on the elitist member, the more similar (smaller distance) between tours generated. This is a rather obvious consequence of the higher probability of further solutions being generated around the global best solution. The better performing weight is then dependent on the length of the run. For pge100, a very small weight is suggested, but not small enough that the search does not reach such a refined solution within the given time. It is clear from the different problems, that features of the problem dramatically alter which level of weight is more optimised for the given time-frame given to the run, however it is clear that a higher weighting is necessary if the problem is of increased complexity and computational cost remains restrictive.

With a weight of $< 0.1$ it is obvious that the speed of convergence is rather slow, though the algorithm does not seem so sensitive to this parameter compared to $\beta$ and $\alpha$ as shown in fig. 14.5.




(a) (b)

Figure 14.5: Parameter study: $we$

The distance between solutions generated is greatest when the elitist weight is lowest but the standard deviation of tour lengths generated is lowest without elitist weight. This is due to the fact that with a high elitist weight, it acts as another population member. With low or no weight assigned to the elitist member, the population archive is in effect reduced since the likelihood of following the path of the elitist member is greatly reduced. Similarly, if the elitist weight is increased considerably then the elitist member

takes a domineering effect over the other population members, in effect, reducing the population. By following the ibest members (non elitist members) of the archive, the search is much more exploratory as indicated by both the ibest, average and distance between constructed and archived solutions.

Looking at the pgc100 problem (clustered), the differences between the weightings becomes clearer with the obvious advantage of a higher weighting for shorter computation times. With a low weight, the distance between members of the population archive are shown to increase for some time. This is likely due to the fact that the elitist member gbest solution not receiving enough deposit to result in a significant chance of a search being made in their neighbourhood. The following solutions generated, and ibest solutions found are unlikely to be related to the best solutions found so far. A further interesting point to be made here is that a weight of $w_e = 1.0$ (elitist deposit only) results in some interesting features in the gbest curve, indicating a too heavy a reliance on the elitist member has varying results depending on the computational cost allocated to the running of the algorithm. It should be noted that the standard deviation of the solutions somehow remains quite high. It is interesting to note however, that the algorithm appears to perform rather well with elitist only as a member. This does coincide with observations of Guntsch, in his PhD thesis [76] (indicated by the choosing of a population of one member). Overall it is considered here that some level of elitism is a good idea in order to speed-up the search but the level weighting chosen is unclear and results are very variable.

Since the performance of the algorithm is not too strongly effected by this parameter compared to others, it is discounted when considering the upcoming parameters. One final observation made here is of the seemingly strange increase in distance between the population members for a weight of $< 0.1$. This occurs as the relative weight between population members compared to the elitist member, results in the significance (or lack of) of the deposit. This in turn, results in future solution components that depend on this weighting. That is, a weight of $> 0.1$ is required to allow the elitist solution to play a more significant role in the solution construction through the deposits made or the $s^{gb}$ solution does not stand out from those components which have yet to be visited (with deposit $\tau_0$). To emphasise this problem, an example deposition comparison between the elitist and other member solutions are shown below:

$$\text{for } n = 400, \tau_{max} = 1.0$$

$$\begin{aligned}\Delta_e &= \frac{w_e(\tau_{max} - \tau_0)}{ke} \\ &= \frac{0.01 * (1.0 - 0.002506)}{1} \\ &= 0.01 \end{aligned} \tag{14.6}$$

$$\begin{aligned}\Delta &= \frac{(1.0 - w_e) * (\tau_{max} - \tau_0)}{ke} \\ &= \frac{0.99 * (1.0 - 0.002506)}{2} \\ &= 0.49 \end{aligned} \tag{14.7}$$

It is clearly observed that it is necessary to have a large enough weighting, so that solution construction has a significant enough attraction toward the elitist member. For this reason, this parameter is intertwined with the chosen ranges for the pheromone.

## 14.6 PACO parameters analysis ($\tau_{max}$)

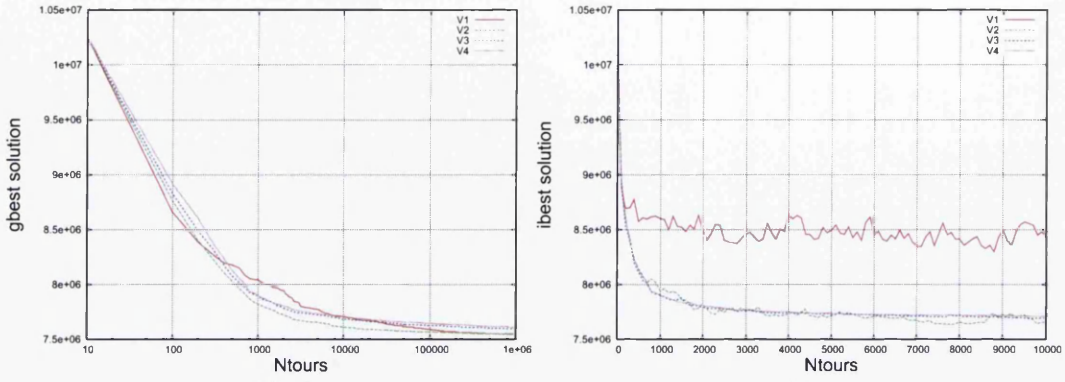
$\tau_{max}$ has the effect of changing the probabilities of choosing past visited solutions. The minimum pheromone is kept constant and since $\tau_{max}$ can be scaled, $\tau_0$ is not considered as anything other than a fixed parameter in this study.

$\tau_0$ is arbitrarily set as $\frac{1}{n-1}$ (so that all rows and columns add to one) [71].

In the roulette wheel selection, consider the case where there are 4 feasible arcs from the current node from which to choose from. The pheromone deposit in this case with no elitist ant can have integer multiples ($k_n$) of a quantity of pheromone given by: $\tau_0 + \frac{(\tau_{max} - \tau_0)}{k}$. Using roulette wheel selection, to change the parameter $\tau_{max}$ results in varying importance of the $\tau_0$ deposits (unvisited arcs) to the solution construction. With a larger $\tau_{max}$, a reduction in exploration might be expected, since the likelihood of choosing unvisited ($\tau_0$) arcs reduces with increasing $\tau_{max}$. On the other end of the scale, to reduce $\tau_{max}$, the importance of the previous experience is reduced (decrease exploitation). Consider when $\tau_{max} \rightarrow \tau_0$, there is then very little difference between them and so exploitation $\rightarrow 0$. Again, this is only the case as $\tau_0$ has been fixed (minimum pheromone over all arcs).

These comments are justified by the observations made in fig. 14.6, where it is shown that the lower the parameter $\tau_{max}$, the greater the exploration (or lack of ex-

ploitation to be more accurate).




(a) (b)

Figure 14.6: Parameter study: $\tau_{max}$

This is indicated by the higher average solution generated, ibest solution generated and standard deviation. It is also clear that the branching factor and distance between solutions stored in an archive follow a similar trend with increasing $\tau_{max}$ (i.e. they decrease as $\tau_{max}$ increases). When observing clustered problems, it is apparent that such a low $\tau_{max}$ does not result in a catchup of the algorithm in solution quality over the computational restraints given to the algorithm. The algorithm is however better performing with small $\tau_{max}$ with its better gbest solutions at the beginning of the search. On this occasion there appears to be no clear trend amongst the standard deviation of tours and it is likely due to the characteristics of the problem influencing any trend. Similar trends to the uniformly randomly generated problem (pge100) are apparent where $\tau_{max} = 2.0$ appears to be close in result to $\tau_{max} = 1.0$. This signifies that a much stronger exploitation of previously found solutions is necessary with the clustered problems. Overall, the best setting for this parameter appears to be an initially low value, then increasing values toward 5.

To further clarify these observations, problems pge200, pgc200, pge400 and pgc400 are briefly described here where any differences of trends are identified. For pge200, it is clear that within the time-scale, the use of $\tau_{max} = 1.0$ results in a very slow convergence behaviour. The most suitable parameter in fact appears to be $\tau_{max} = 2.0$. Similar trends to pge100 are observed with the one objection. The standard deviation of tours appears to be maximum with use of $\tau_{max} = 2.0$, with $\tau_{max} = 1.0$ closely following but with average solution quality with $\tau_{max} = 1.0$ being the worst. Again, pgc200 indicates that a higher $\tau_{max}$ results in a better performance with $\tau_{max} = 5.0$ ap-

pearing to be optimum. On the much larger problems with the computational limits set (pge400), the performance of the $\tau_{max} = 2.0$ was not able to catch-up with the $\tau_{max}$ set-up, but it is strongly suggested from the curve that it would indeed do so if given enough time. Regarding pgc400, results indicated a small difference between $\tau_{max} = 5.0$ and $\tau_{max} = 10.0$, which could go either way in respect to their average performance, if left to continue.

To summarise, the parameter $\tau_{max}$ dictates the importance of the deposited (counted) solution components visited previously. Since $\tau_0$ remains fixed, it is clear that an increasing $\tau_{max}$ results in a decreasing importance of the arcs with $\tau_0$ deposit (decreasing exploration). For this reason, it is suggested that an initially low value of $\tau_{max}$, with increasing value as the search progresses may be advantageous. On a final note, the distance between constructed tours and also those recorded in the archive decrease as a function of increasing $\tau_{max}$, where this demonstrates the effect of decreasing the statistical chance of choosing solutions which have no deposit ($\tau_0$) as the deposit of the visited arcs is increased (increasing $\tau_{max}$).

## 14.7 PACO parameters analysis ($nn$)

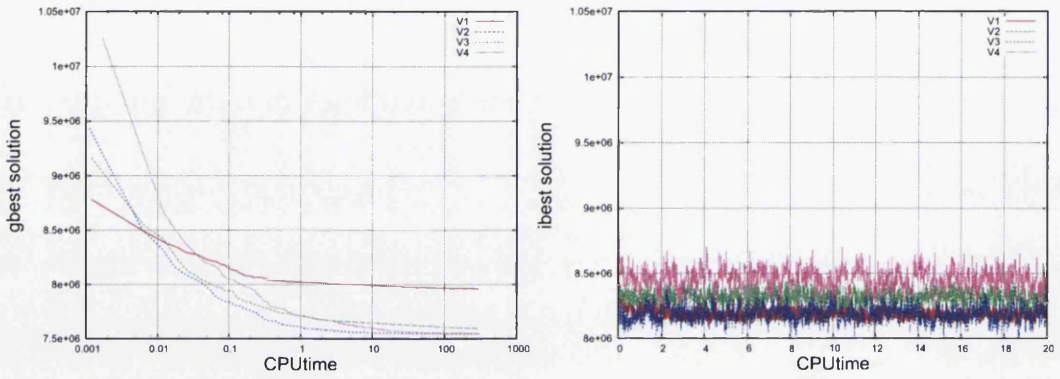
This parameter offers to significantly speed up the algorithm by means of discounting those arcs which are far apart (very distant neighbouring nodes). The reasoning behind this approach, is that there are only a certain number of arcs which have a high enough chance of being chosen and so implementing a nearest neighbour list allows a reduced number of possible neighbours to be chosen from (removing those which have negligible chance of being chosen). It is obvious that this parameter will be problem specific and indeed problem characteristic specific and so a generous number is likely to be a good choice to encompass all problems without the loss of accuracy. First, the importance of this parameter is to be discussed including its incorporation into the algorithm.

Firstly, it should be re-iterated that both $we = 0$ (disabled), $q_0 = 0$ (disabled) and that the branching factor for analysis is now determined amongst the entire matrix rather than the nearest neighbours (as this would not offer to be a fair comparison of the parameter). The following particulars of the algorithm are to be observed when incorporating the candidate list strategy.

The candidate list significantly reduced the solution construction phase of the algorithm and is implemented by means of reducing the loop from $n$ to $nn$. However, the

undesired consequence of this, is that when no unvisited node is present within the $nn$ list, a deterministic choice is made based on the arcs with maximum *total information*. This is increasingly likely as the length of the candidate is decreased. If no unvisited node (feasible) is found within the candidate list then the solution component with maximum total information is chosen (deterministic). That is, to decrease the candidate list results in an ever more deterministic solution construction occurrence.

It is clear from the plots (see fig. 14.7 for a representative plot), that a candidate list of $\leq 5$ results in a reduction of quality of the final solution with however the very initial increase in speed with better solution quality.




(a) (b)

Figure 14.7: Parameter study: $nn$

A candidate list of over 10 however appears to offer no significant benefit to solution quality but with a much increased computational expensive. As suggested previously, as the candidate list becomes too small, the number of *best next* calls during solution construction (deterministic choosing) increases. This then has the undesired downside of trapping within sub-optimal optima with the reduced exploration. Regarding plots other than the $s^{gb}$, the $s^{ib}$ indicates that the worst solution quality per iteration is with the largest candidate list ($nn = n - 1$) and this stems from the higher exploration resulting from not choosing the 'best next' solution at any time. Surprisingly, this is then closely followed by $nn = 5$ not $nn = 10$. Analysing the average branching factor, the distances between solutions generated in each cycle and also the distances between solutions stored within the archive has given a number of indication as the reasoning behind the performance changes of the algorithm. The highest average branching factor (as expected) is with the largest candidate list. However, again a candidate list of length 5 appears to have a higher exploration than a length of 10. It is hypothesised that

this is due to the features of the problem, though the differences are perhaps not that significant. Regarding the distances between solutions in the archive, the same trend as with the branching factor is observed, though of the solutions generated the distance between them appears to be least for a candidate list of 10 indicating that the search is made in a more localised manner around the archived solutions. It is interesting to note that a much reduced candidate list is better in the case of the clustered pgc100 problem closely followed by a length of 10. This is likely due to the characteristics of the problem (the way in which the nodes are distributed). It is not well understood as to specific reasons of performance of this parameter on this particular occasion, only that it is characteristic dependent. What is clear however is that a candidate list of 10 seems to perform rather well across the problems investigated here. This parameter appears to offer to significantly decrease the computational cost per step but at the cost of increasing bias toward the highly suggestible (high heuristically suggestible) arcs in the most extreme case (very small candidate lists).

## 14.8 PACO parameters analysis ($k$)

This parameter is the size of the population (number of solutions recorded in the solution archive). Each solution in this archive has at most $k$ iterations before its influence is removed. This is due to the fact that the first-in-first-out (FIFO) method like the original PACO algorithm has been implemented here. The original PACO algorithm being called FIFO-QUE. With $k = 1$, there is one solution member and one elitist. In the case where $we = 0.0$, there is no elitist and the case where $k = 1$ would then signify a single ibest update.

One might expect that to increase the size of the solution archive will result in increased exploration. This is due to there being deposits on multiple solutions arcs giving more choice during the solution construction phase. For a population of $> 5$, it was found that the individual deposit was then not significant enough to result in its attraction towards its inclusion in the solution construction i.e. as the number of solutions recorded in the archive increase and $\tau_{min}$ and $\tau_{max}$ remain fixed, the attractiveness of a visited solution over the unvisited solution becomes diminished. It is for this reason that few updates in the $s^{gb}$ solution are apparent with $k > 5$. Since this parameter is strongly correlated with the range of pheromone allowed, it is difficult to differentiate the real effect of introducing extra solutions into the archive while keeping a consistent attraction compared to unvisited solutions. To more fairly deduce a relationship

between storing extra solutions into the archive, the amount of deposit made by each solution is kept fixed by rearranging the formulation for the pheromone update:

$$\text{set } \Delta = 0.3 - \tau_{min} \tag{14.8}$$

$$\tau_{ij} = \tau_{ij} + \Delta \tag{14.9}$$

$$\tau_{ij} = \tau_{min} + \frac{\tau_{max} - \tau_{min}}{k} \quad \text{for the first deposit} \tag{14.10}$$

$$0.3k = k\tau_{min} + \tau_{max} - \tau_{min} \tag{14.11}$$

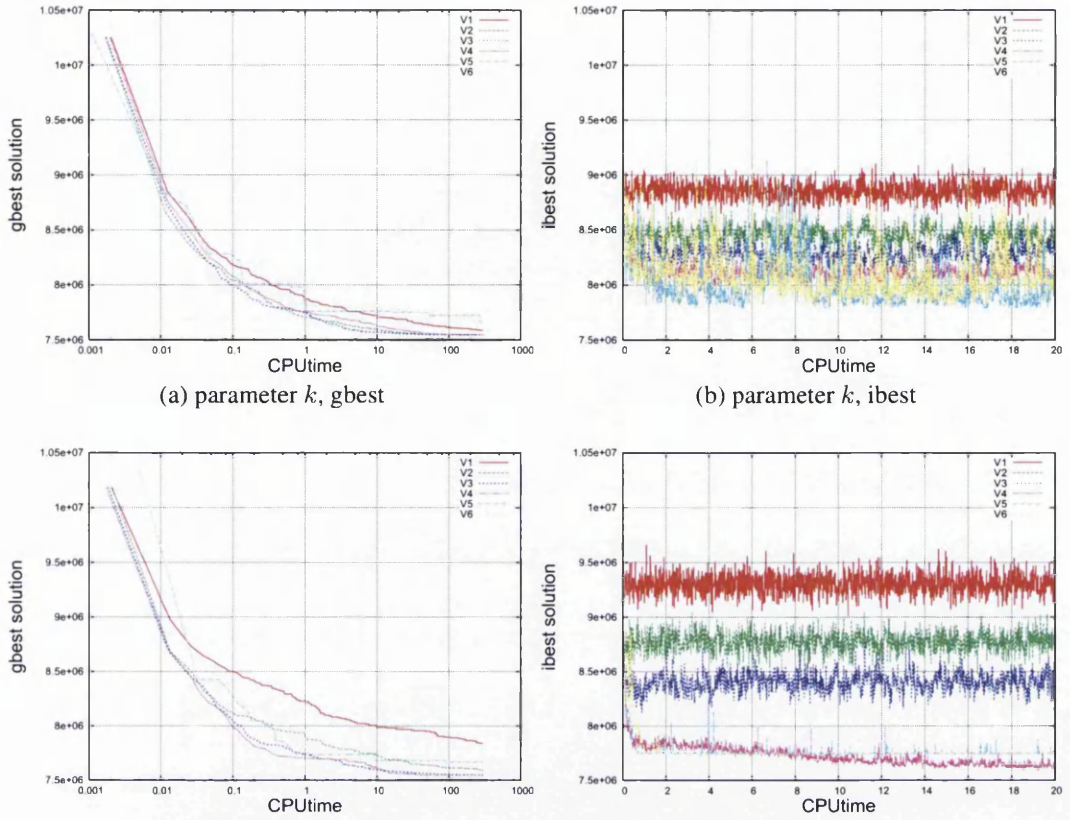
$$\tau_{max} = \tau_{min} + k(0.3 - \tau_{min}) \tag{14.12}$$

It must then be considered that more solutions may visit the same arcs and therefore offer a much greater attraction toward them in the solution construction phase compared to the unvisited $\tau_{min}$ arcs. To this end, this experiment with both altering $\tau_{max}$ and $k$ is made in order to gain further insight into the workings of the algorithm.

Before this additional study is described, it should be noted that when increasing the number of solutions which enter the archive, the standard deviation of generated tours is considerably more erratic, while the distance between these generated tours are a little less than those with smaller populations. Another point made here is that a larger than 1 archive is of definitive importance to the performance of the algorithm (likely due to the added exploration gained from the influence of multiple solutions).

To further study this parameter, the relation between altering both $k$ and $\tau_{max}$ is observed, keeping the single deposit constant (arbitrarily chosen). Now, a much greater difference between the parameter changes are observed. The two parameters were too strongly coupled to be able to understand the effect of one alone. Parameter $m$ may also be considered to be strongly coupled, but it is assumed to be to a lesser extent, especially since an insight into the variation of the archive size is to be studied here (i.e. will not incorporated into the study of this parameter).

From observing the plots concerning the varying $k$ and $\tau_{max}$ as shown in fig. 14.8, it now becomes clear that contrary to that described in the literature, a larger solution archive $k$ appears advantageous. In fact a value of between 5 and 10 appears to be best over the chosen problems. With the later indicating a better mean $s^{gb}$ solution in larger/more complex problems. To re-emphasise what was described earlier, by increasing the size of the population, the length of influence of the particular solution stored within is increased, but also the number of solutions having influence on future solution construction. That is, more exploration around the solution found is apparent if the




(a) parameter $k$, gbest

(b) parameter $k$, ibest

(c) parameter $k\tau_{max}$, gbest

(d) parameter $k\tau_{max}$, ibest

Figure 14.8: Parameter study: $k$ and combined $k$ with varying $\tau_{max}$

solution archive is longer. However, the larger the archive, the greater choice between more numerous influential solutions are apparent. The level of exploration is based on not only the number of solutions in the archive but also the level of exploitation by the number of perhaps similar solutions within this archive (the distance between these solutions). The first observation to be made is that the ibest solutions decrease with increasing archive size, and further still, the ibest solutions with an archive of over 5 appear not so erratic. This in indicated by the smoother curve. Looking at the standard deviation, it appears (as expected) that the solutions generated are highly different and ever more erratic for larger archive sizes. A highly exploratory search is made with increasing archive size. Since the ibest curve appears not so erratic, it can be concluded, that this signifies the local optimum as being located in a reduced time in comparison to the lower archive sizes. As the archive size is increased, the similarity between solutions will reduce, as ibest solutions found are highly likely to be next to previous ibest

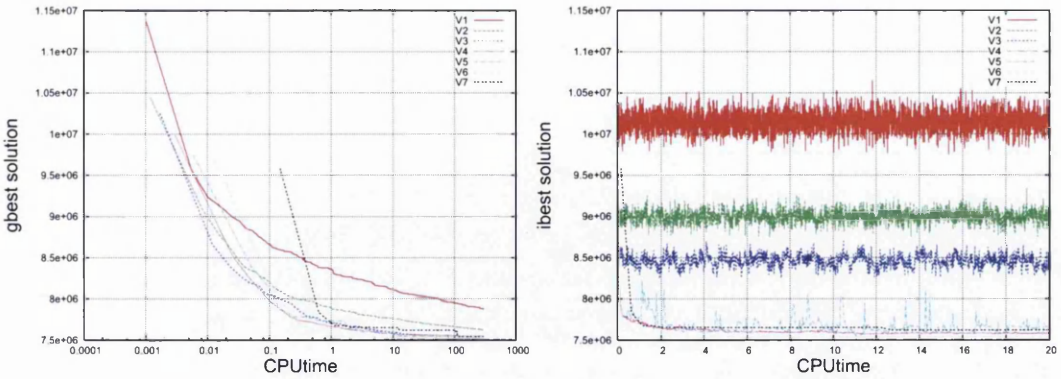
solutions due to the expect 'basin' structure of the TSP. If the local optima are located quickly by the greater exploration of the population, then it is not so unexpected to see reduced activity in the ibest curve of larger archive set-ups contrary to the increased number of influential solutions. Looking at the location of the update for $s^{gb}$ ($s^{gb}$ cycle location), to terminate the search early would be rather undesirable since the search with larger archives (10) explore the area of the solutions found so far in great detail, resulting in stages of no better solutions being found. For this reason, a population archive of 5 is more recommended as a general parameter while a population of 10 is perhaps more appropriate if time constraints are not too restricted. As a final note, an interesting point to consider would be the case where ibest solutions are unlikely to be next to each other (separated by great distance). In such a case, the greater size of archive would inevitably result in a much better exploration and more thorough search of the search space, encompassing rather different characteristics to the effect of this parameter on the TSP. This could very well be the case in scheduling problems. Such problems are well known to exhibit UV-curves [105], in which a greedy algorithm performs rather badly due to the sheer number of local optima and uncorrelated fitness-distance relation.

## 14.9 PACO parameters analysis ($m$)

This parameter has the effect of defining the number of independent solutions constructed per cycle (iteration). At the end of the cycle, the deposition and or removal of pheromone occurs resulting in a new matrix of attracters for the next round of solution construction.

As $m$ is increased, it might be expected for exploration to increase, as a greater number solutions are constructed. With $m = 1$, only one solution per cycle is constructed, resulting in the minimum level of exploration. It is observed that the larger the number of ants (tours generated per cycle), the better the solution if given enough time, due to the increased level of exploration, as shown in fig. 14.9.

The most significant difference to solution quality however appears for populations of less than 10. With a population of 10, very similar performance to much greater populations are observed with the unnecessary computational costs associated with it. In the most extreme case considered (population of 300), the solution takes a considerable amount of time to refine and reach similar quality to a population of only 10 and in fact, it does not converge to a similar quality of solution within the bounds imposed in this study. Observing the $s^{ib}$ curve, it is clear that the best solution found per iteration




(a) (b)

Figure 14.9: Parameter study: $m$ as a function of CPU time

fluctuates greatly and that it is considerably worse than with greater populations. This can be reasoned by the fact that a greater number of solutions generated will no doubt generate a statistically better and more consistent ibest. With a population of greater than 10, it is observed from the ibest curve that a rather consistently good ibest solution is found thus signifying that a much narrower number of solutions will enter the storage archive. This is further indicated by the distance between solutions stored in the archive decreasing with population increase. Interestingly enough, the solutions generated by the algorithm have greater similarity as the population size is increased (up to a point). Observing the tour standard deviation, it becomes clear that as the population size exceeds 25, a decrease in the standard deviation of tours in apparent, similarly for that exceeding 50. However, as the population reached 100, another increase is observed. It is hypothesised that this is due to two underlying reasons behind exploration. The first is that exploration is increased as the number of solutions per cycle is increased. The second reason that causes greater exploration is to have fewer solutions per cycle as those that enter the archive are more likely to be different from one-another if the ibest solutions found are not so consistently good (ergo, likely similar). i.e. a double edged sword is described, where exploration is increased by a larger population, but too large a population confines the search in the very close neighbourhood of very good solutions.

Looking at the clustered pgc100 problem, it appears that a much greater population is advantageous and in fact, smaller populations do not catchup with the higher population within the time-scale (no catchup occurs within the time-scale defined for the run). With larger populations, the probability of finding an improved solution within

the neighbourhood is greatly increased, reducing the likelihood for adding distant solutions to the archive, this is a sign of the importance to the performance of the algorithm, where a clustered characteristic is indicated (i.e. a more thorough search of the neighbourhood is helpful for 'difficult' and or clustered problems). Again, the average solutions produce much better quality with larger population size, indicating that a more refined search is made with larger populations. The branching factor indicates that predominantly solutions constructed follow the same path for very large populations: also indicated by the distance between archived solutions. pge200 performs best it seems for a population of 50, with fair performance over the medium term (runtime) but also in terms of slightly better solutions than a larger or smaller population. It could very well be due to the difficulty of problem and the fact that the algorithm is applied for the same time as the smaller problem, so that the larger population set-ups have yet to catch up.

It is recommended that a small population (5) is advantageous in the short term, with increasing population size as the search progresses with a medium sized population (25) indicating fair performance in the medium length time-scale with finally an even larger population indicating the better performance in the longer time-scales (50).

To understand the erratic behaviour of both the distance between solutions in the archive together with the average branching factor for $m \geq 25$, we must realise that a high $m$ ensures a much greater and thorough search of the neighbouring solutions around the attracted solutions within the archive. This then results in consistently good $s^{ib}$ solutions, and ones which are rather similar. As discussed previously, two good solutions are highly likely to be very similar to one-another due to the characteristics of the search space of the TSP (the fitness-distance correlation of this problem). As solutions which enter the archive are ever similar as $m$ is large (indicated by the distance between them), then the deposits made by entering the archive a more likely to be made on the same arcs, resulting in a reduced average branching factor. It then follows logically that a similar trend be observed with the average branching factor to the distance between solutions within the archive. Regarding the erratic yet infrequent increase of both the distance between solutions in the archive and the average branching factor, this is likely due to the sheer number solutions made per cycle. What is meant, is that due to the sheer number of solutions constructed per cycle, the likelihood of choosing solution components which have not been selected is greatly increased.

To summarise, a higher $m$ results in a more thorough search of the neighbourhood solution ensuring that the best $s^{ib}$ enters the archive. Decreasing $m$ however has the

effect of decreasing the likelihood of finding the locally best $s^{ib}$, resulting in greater exploration. From this observation, it is clear that there are two different types of exploration through the increase or decrease of this parameter. Increasing it, increases the level of exploration within a cycle, which in turn results in a more exploitative approach due to those solutions which enter the archive. A low value $m$ however results in actual exploration by not necessarily adding the best possible solution in the neighbourhood to be entered and influence upcoming solution construction.

## 14.10 Summary

Firstly, the parameters studied are characterised here in a table, according to certain effects or groupings, to help clarify the results of the previous section. These are shown in table 14.2.

To conclude on this study, a number of parameters are strongly coupled and the use of a population archive is the likely cause of such a strong coupling. Furthermore, the deposit which is not based on the quality of solution, also has the effect of exaggerating the effects of parameter adjustment, making such parameters as $\alpha$ and $\beta$ perform well, over only a small interval. The result of this study leads to the suggested offline tuning of the algorithm with use of a simple PSO, for which a proof of concept is made here for the PACO algorithm.

| Param | Notes |
|---|---|
| $\alpha(8)$ | Dramatically changes the time by which the algorithm converges to the optimum.<br>Together with $\beta$, controls the emphasis of previous experience.<br>Optimal value: 1.0 for uniform and 1.5 for clustered. |
| $\beta(10)$ | The most sensitive parameter to the performance of the algorithm<br>Determines the amplification of heuristic info. over previous exp.<br>Optimum: $\beta = 5.0$ overall and $\beta = 10.0$ for faster convergence. |
| $q_0(4)$ | Strongly controls the level of exploitation.<br>Determines how often the deterministically 'best' decision is made (total inf.).<br>Optimum: User defined, recommended $0.0 < q_0 < 1.0$, keep a default $q_0 = 0.5$ |
| $w_e(3)$ | Strongly controls the level of exploitation.<br>Emphasis of the elitist $s^{gb}$ member.<br>Optimum: $0.01 - 0.90$, recommended 0.1 |
| $\tau_{max}(3)$ | Changes the attraction of deposits relative to the unvisited arcs with $\tau_0$<br>Optimum: $1 - 10$, problem specific |
| $nn(3)$ | Acts as a speed-up by reducing the possible choices to those most likely ($nn$ most likely)<br>Reducing the candidate list gives higher bias towards heuristically best.<br>Optimum: 10 overall (problem dependent) |
| $k(4)$ | Increases the length of influence of solution $s^{ib}$<br>Controls the number of solutions of influence.<br>higher local exploration with increasing $k$<br>Optimum: $5 - 10$ depending on time constraints (10 for the more highly constrained) |
| $m(5)$ | Is equivalent to the number of solutions constructed per cycle.<br>Increasing $m$ increases exploration per cycle (localised search).<br>Increasing $m$ decreases exploration overall with consistent $s^{ib}$ solutions enter archive.<br>Optimum $\geq 5$ depending on problem, Default 10. |

Table 14.2: Estimated sensitivity rating of the parameter

# Chapter 15

# Offline parameter tuning

## 15.1 Set-up

Some offline tuning methods are described in section 12.5. In order to tune the parameters of the ant colony algorithm, the most basic form of particle swarm is implemented with chosen variables heavily influence by [7]. For the basic PSO, the GBEST model is considered, with the parameters presented in table 15.1. The constraint handling technique chosen is a preserving feasibility method, where the solution coordinate is set to the bound if it leaves it. With regards to initialisation, a standard uniform random distribution is used. For additional information to the paradigm known as PSO, the reader is referred to chapter 7.

| Parameter | Description |
|---|---|
| $iw = 2.0$ | Individuality weight |
| $sw = 2.0$ | Social weight |
| $w^1 \rightarrow 0.9, w^{t_{max}} \rightarrow 0.0$ | Linearly changing inertia weight |
| $t_{max} = 100$ | Maximum number of time-steps |
| $runs = 8$ | Number of repeat runs of the PSO |
| $n = 10$ | Swarm size |

Table 15.1: Parameter set-up for the basic GLOBAL PSO algorithm.

With regards to the problem tackled, an objective function is required to be defined for the PSO, signifying the performance of the ACO algorithm across the 3 problems (eil51, kroA100 and d198) with 3 samples considered on each, making it 9 problems in total (with each sample having its own seed re-called each PSO time-step). The objective variables of the PSO correspond to the input parameter set of the ACO. Two simple

methods are possible: One method is where offline tuning occurs for each problem individually and some trend determined from this; the other, is where the objective function is defined as some form of average across the range of problems. The latter method is approached here for the reason of simplicity alone, where the objective function of the PSO is formalised as a mean penalised and normalised fitness across the problem set.

The mean normalised fitness across 8 PSO runs is determined, with the original non-normalised fitness corresponding to the gbest tourlength found. This is normalised by using both known bounds: the first (the minimum bound), corresponds to the known global minimum; the upper bound (maximum solution), corresponds to the longest length tour possible using a nearest neighbour heuristic (determined a priori to this tuning phase). The upper bound is determined by applying a nearest neighbour heuristic at each starting node, then tacking the maximum tourlength found. The mean normalised fitness of each problem is then given by eq. (15.1).

$$f(\mathbf{X})_i = \begin{cases} \frac{\text{Normalised mean fitness}}{\text{success-rate}}, & \text{if success met on at least 1 run} \\ \frac{\text{Normalised mean fitness}}{\frac{1}{3*runs}}, & \text{otherwise} \end{cases} \tag{15.1}$$

where success-rate $= \frac{\text{success}}{\text{runs}}$

This takes not only the tourlength into account but also the success-rate. In the case where no success is met on any run, the fitness is penalised by an arbitrary factor of 3, by reasoning that some poor solutions are better than all poor solutions. Regarding the termination conditions for the ACO, the standard limit for these three problems is set as originally defined by the International Contest on Evolutionary Optimisation[93] which is $10000 \times n$, where $n$ is the number of dimensions of the problem.

The objective function for the PSO is then determined by the mean value of eq. (15.1) across the set of problems as shown in eq. (15.2).

$$F(\mathbf{X}) = \frac{\sum_{i=1}^{p} f(\mathbf{X})_i}{p} \tag{15.2}$$

With respect to the chosen bounds to each parameter (limits of the objective variables of the PSO), the parameter study from chapter 14 together with the resources of Guntsch and Middendorf [74], Guntsch [76] (originator of the PACO algorithm), allows suitable ranges to be set. The more recent parameter study by Oliveira et al. [77] on PACO also has its influence on the chosen parameter ranges. These ranges are shown in table 15.2.

| Parameter | Note |
|---|---|
| $0.1 \leq \alpha \leq 5.0$ | |
| $0.1 \leq \beta \leq 10.0$ | |
| $0.0 \leq q_0 \leq 1.0$ | |
| $0.0 \leq w_e \leq 1.0$ | |
| $1.0 \leq m \leq 50.0$ | Restricted to integers (NINT) |
| $1.0 \leq k \leq 25.0$ | Restricted to integers (NINT) |
| $1.0 \leq \tau_{max} \leq 100.0$ | |
| $1.0 \leq nn \leq 20.0$ | Restricted to integers (NINT) |

Table 15.2: Parameter bounds for the offline tuning method.

Those parameters which are discrete are simply rounded to their nearest integer value. These include parameters $m$, $k$ and $nn$, as shown in table 15.2.

## 15.2 Results

Since only 8 PSO repeat runs were conducted, it cannot be assumed that the best performing parameter set over the given problems is found. Nonetheless, the best performing parameter set is presented and it is demonstrated to result in an improved performance on the larger problems (kroA100 and d198). This is with respect to the computational requirements together with reduced mean final solutions. With this increase in performance over the two problems, this has resulted in only a small reduction in efficiency (CPU time) on the smaller eil51 problem. This then indicates the parameter set to be more suitable than the default set as defined by Guntsch [76] over these given problems. Furthermore, the converged parameter set is somewhat similar to those described by Guntsch [76], indicating that the converged solution set to belong to the same valley to which the default set is located, but now better refined.

Comparison between the four algorithms including the offline parameter tuned PACO (PACO_t), PACO with default parameter set (PACO_d), ACS and MMAS is now made. These results indicate the good overall performance of PACO_t, PACO_d and MMAS on eil51 with the best indicated by PACO_t. With kroA100, PACO_t has the lowest mean solution with a rather small number of tours required to achieve these solution. It is closely followed by MMAS in terms of solution quality, while achieving these solutions in a much reduced number of tours. Finally, MMAS and PACO_t achieve a similar mean solution quality with respect to d198, while MMAS requires on average a smaller number of tours to achieve such a solution. However, PCAO_t is observed to

have a clear advantage over both PACO_d and ACS. This is not surprising as MMAS and PACO_t have their parameters designed according to the limits set by [93]. For the readers interest, the statistical significance of these results are shown in appendix F.

| eil51 | | | | kroA100 | | | | d198 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PACO_t | tourlength | gpos | cputime | PACO_t | tourlength | gpos | cputime | PACO_t | tourlength | gpos | cputime |
| best | 426 | 42 | 0.016 | best | 21282 | 274 | 0.240 | best | 15908 | 7418 | 19.377 |
| worst | 433 | 467 | 0.156 | worst | 21650 | 753 | 0.672 | worst | 15997 | 17680 | 46.775 |
| mean | **427.9** | **1554.5** | 0.537 | mean | **21310.0** | 5777.4 | **5.171** | mean | **15964.5** | **19249.1** | 51.017 |
| stdev | 2.2 | 1943.7 | 0.669 | stdev | 83.1 | 5990.1 | 5.328 | stdev | 21.5 | 10632.3 | 28.966 |
| sterr | 0.5 | 434.6 | 0.150 | sterr | 18.6 | 1339.4 | 1.191 | sterr | 4.8 | 2377.4 | 6.477 |
| PACO_d | tourlength | gpos | cputime | PACO_d | tourlength | gpos | cputime | PACO_d | tourlength | gpos | cputime |
| best | 426 | 639 | 0.060 | best | 21282 | 2946 | 0.708 | best | 15893 | 75794 | 47.659 |
| worst | 432 | 369 | 0.036 | worst | 21600 | 1998 | 0.480 | worst | 16680 | 67899 | 42.379 |
| mean | **427.0** | 8098.5 | 0.795 | mean | 21340.0 | 27318.2 | 6.757 | mean | 16002.7 | 116695.8 | 73.517 |
| stdev | 1.1 | 8023.4 | 0.786 | stdev | 104.9 | 30876.6 | 7.622 | stdev | 160.3 | 54643.8 | 34.525 |
| sterr | 0.2 | 1604.7 | 0.157 | sterr | 23.4 | 6904.2 | 1.704 | sterr | 35.8 | 12218.7 | 7.720 |
| ACS | tourlength | gpos | cputime | ACS | tourlength | gpos | cputime | ACS | tourlength | gpos | cputime |
| best | 427 | 2807 | 0.288 | best | 21282 | 7255 | 1.584 | best | 15872 | 24333 | 7.245 |
| worst | 434 | 922 | 0.096 | worst | 21952 | 15277 | 3.292 | worst | 16931 | 80951 | 24.770 |
| mean | 428.3 | 6685.8 | 0.655 | mean | 21467.3 | 26678.0 | 5.808 | mean | 16192.2 | 125474.6 | 41.146 |
| stdev | 2.5 | 5156.2 | 0.484 | stdev | 194.7 | 21872.7 | 4.767 | stdev | 218.1 | 56692.1 | 18.944 |
| sterr | 0.6 | 1153.0 | 0.108 | sterr | 43.5 | 4890.9 | 1.066 | sterr | 48.8 | 12676.8 | 4.236 |
| MMAS | tourlength | gpos | cputime | MMAS | tourlength | gpos | cputime | MMAS | tourlength | gpos | cputime |
| best | 426 | 463 | 0.276 | best | 21282 | 841 | 1.480 | best | 15886 | 2315 | 23.985 |
| worst | 431 | 549 | 0.332 | worst | 21389 | 756 | 1.908 | worst | 16019 | 2693 | 27.986 |
| mean | **427.3** | 2499.0 | 1.447 | mean | 21316.5 | 2971.8 | 7.361 | mean | **15963.5** | 3335.6 | **25.862** |
| stdev | 1.1 | 2183.3 | 1.299 | stdev | 47.0 | 2908.1 | 7.296 | stdev | 24.5 | 2159.1 | 19.396 |
| sterr | 0.2 | 488.2 | 0.290 | sterr | 10.5 | 650.3 | 1.631 | sterr | 5.5 | 482.8 | 4.337 |

Figure 15.1: Comparing the effectiveness of the parameter tuning of PACO (PACO_t) against other available algorithms, including PACO with default parameter set (PACO_d), ACS and MMAS. Significant results shown in bold. Results are conducted over 20 sample runs with $10000 \times n$ iterations.

## 15.3 Conclusion

The offline tuning of the PACO algorithm has demonstrated its ability to achieve a much greater performance compared to the default parameter set. Additionally, both PACO_t and PACO_d alongside MMAS have outperformed ACS over the limits set by [93] for these function. This was not unexpected, but the offline tuning through hybridisation of ACO with PSO for parameter tuning has resulted in the performance increase of PACO toward MMAS and in fact achieving a better mean solution in one of the three problems (kroA100). With the added pheromone restart procedure described by Oliveira et al. [77] for PACO, it is expected to utilise even greater performance with the offline parameter tuning implemented here. This method is a proof of concept only for the PACO algorithm, since the algorithm is somewhat underutilised amongst the community. The algorithm takes elements from EAs, with its population archive of solutions, allowing information collected during the search to be better utilised for the remainder of the search. For this reason, the successful improvement in performance

of the algorithm is justified for the continuation of research of this algorithm, to tackle multi-modal problems, MOPs and niching for example (see section 12.4.4).

# Part IV

# Conclusion

# Chapter 16

# Conclusion

## 16.1 Summary

Firstly, an investigation in the application of a local search (SQP) combined with the in-house particle swarm optimiser (GP-PSO) was made. A literature review of the paradigm, considering both origins and other local search hybrid approaches was made. Following this, the local search was applied to the solutions provided by the GP-PSO, resulting in a considerable refinement of solutions being observed, especially in the case of the constrained problem suite (CEC06). Additionally, an investigation into early termination was conducted, concluding the method chosen as being suitable as a stagnation (termination) measure for unconstrained problems. However, for the constrained problem suite, the additional requirement for feasibility of the swarm to be met, resulted in the effective use of these measures as an early switching technique. That is, for considerable computational savings to me made. On average, 80% of the number of function evaluations are required for the GP-PSO-SQP to meet succes as defined by the suite (utilising the derived early switching method) compared to the point at which the GP-PSO finds the global optimum. This then functions not only as a stagnation measure, but as an early switching technique.

In the discrete problem domain, the subject area at Swansea University engineering department is new. For this reason, an extensive literature review was conducted, concluding that an algorithm (PACO) which has been mostly overlooked in the literature was suitable for further study. Following this, a parameter study of the chosen algorithm was conducted, resulting in a detailed understanding and a realisation of the strongly coupled parameters present. It is likely that this algorithm has more strongly coupled parameters than the more popular MMAS and ACS algorithms. Finally, a method in

which to overcome this parameter coupling was devised, hybridising a simple PSO with the PACO algorithm for off-line tuning. The result of this off-line tuning, was the optimisation of the algorithms parameters, resulting in overall better performance than the original and indicating clear strengths in some cases over the more popular MMAS and ACS algorithms. The conclusion of this study is that the PACO algorithm is likely to perform at least as well if not better than other leading algorithms in tackling various applications. The PACO algorithm also has a population archive, making it the only one to have tackled niching. This makes it highly suitable for tackling multi-objective, multi-modal and dynamic problems. Since scheduling problems are the overall intention for future application from this thesis and such applications are often multi-modal, the PACO algorithm is a good choice for further research.

## 16.2 Contribution

This thesis has resulted in the effective hybridisation of the already advanced in-house particle swarm optimiser at the engineering department of Swansea University (GP-PSO) with SQP local search, utilising this black box method available within the optimisation toolbox of Matlab. To this end, the GP-PSO-SQP has shown to reach competitive performance with current leading algorithms presented for comparison. Early termination or stagnation measures are rather infrequently reported within the literature and so this work also contributes towards a rather underdeveloped but important area within the paradigm.

Within ACO, the PACO algorithm has been identified as a clear contender with leading algorithms, showing a distinct advantage in some cases. With the hybridisation of the algorithm with a PSO for offline tuning, PACO has clearly been observed as having benefited, indicating its suitability for further research and application.

## 16.3 Future work

The GP-PSO-SQP has proven the effective use of a local search with the GP-PSO for solution refinement, however, a much greater computational saving may be apparent if problem type patterns within the swarm measures were to be derived. Additionally, one of the most effective PSO algorithms within the literature (DMS-PSO) (see section 8.3), utilises a local search for solution refinement within the swarm on an on-going basis.

This is quite possibly a more effective approach in the hybridisation of the two methods and a suitable method for further investigation.

The work conducted with respects to the PACO algorithm is somewhat in its infancy. The underlying unique feature of the algorithm is its population archive, bridging the gap between particle swarm and evolutionary algorithms such as genetic algorithms. For this reason, the PACO algorithm is able to much more easily utilise features inherently successful in EAs for the purpose of optimisation tasks. Indications of which are clear with its application in the literature to multi-objective problems and with niching. Job Shop Scheduling problems have been identified as having multi-modal characteristics of its search-space and as such, an algorithm capable of niching could quite possibly benefit such applications. Since Job Shop Scheduling is the intended applications that are to follow from this thesis, the PACO algorithm has been identified has a highly suitable candidate to further research in this direction.

# References


[1] Daniel Merkle and Martin Middendorf. A new approach to solve permutation scheduling problems with ant colony optimization. In *EvoWorkshops*, pages 484–494, 2001.

[2] James Kennedy, Russell C. Eberhart, and Yuhui Shi. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[3] Mauro Sebastián Innocente. *Development and testing of a Particle Swarm Optimizer to handle hard unconstrained and constrained problems*. PhD thesis, Civil and Computational Engineering Centre, College of Engineering, Swansea University, Swansea, United Kingdom, 2010.

[4] Zbigniew Michalewicz and David B. Fogel. *How to solve it : modern heuristics*. Springer, December 2004.

[5] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. The MIT Press, 2004.

[6] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd edition, 2007.

[7] Mauro Innocente. Population-based methods: Particle swarm optimization - development of a general-purpose optimizer and applications - parti. Master's thesis, Civil & Computational Engineering Centre University of Wales Swansea, 2006.

[8] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67 –82, april 1997.

[9] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of life reviews*, 2(4):353–373, DEC 2005.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in Fortran 90 (2nd ed.): the art of parallel scientific computing*. Cambridge University Press, New York, NY, USA, 1996.

[11] URL http://mathworld.wolfram.com/SecantLine.html. Definition of the secant line (Jan 2011).

[12] C. Blum. Ant colony optimization: Introduction and hybridizations. In *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pages 24 –29, 17-19 2007.

[13] Keld Helsgaun. General k-opt submoves for the lin-kernighan tsp heuristic. *Mathematical Programming Computation*, 1: 119–163, 2009.

[14] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, October 2000.

[15] Camilo Rostoker, 2005. URL http://www.cs.ubc.ca/labs/beta/Courses/CPSC532D-05/Slides/tsp-camilo.pdf. (May 2011).

[16] Mauro Sebastián Innocente. Population-based methods: Particle swarm optimization & development of a general-purpose optimizer and applications. Master's thesis, Civil & Computational Engineering Centre, University of Wales Swansea, 2006.

[17] Thomas Weise. *Global Optimization Algorithms - Theory and Application*. Thomas Weise, 2009-06-26 edition, 2009.

[18] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35:268–308, September 2003.

[19] Harun Pirim, Engin Bayraktar, and Burak Eksioglu. *Tabu Search: A Comparative Study*. InTech, 2008.

[20] Michel Gendreau and Jean-Yves Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140:189–213, 2005.

[21] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization*. Springer, 2006.

[22] D.B. Fogel. What is evolutionary computation? *Spectrum, IEEE*, 37(2):26, 28 –32, feb 2000.

[23] Michael Affenzeller, Stephan Winkler, Stefan Wagner, and Andreas Beham. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications (Numerical Insights)*. Chapman & Hall, 1 edition, April 2009.

[24] D. Karaboga. An idea based on Honey Bee Swarm for Numerical Optimization. Technical Report TR06, Erciyes University, October 2005.

[25] S. Walton, O. Hassan, K. Morgan, and M.R. Brown. Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos, Solitons & Fractals*, 44(9):710 – 718, 2011.

[26] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 1-6, pages 1942–1948, Perth, 1995.

[27] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. A review of particle swarm optimization. part i: background and development. *Natural Computing*, 6:467–484, 2007.

[28] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7:109–124, 2008.

[29] J. Mullen, R., S. Monekosso, S. Barman, and P. Remagnino. A review of ant algorithms. *Expert Systems with Applications*, 36:9608–9617, 2009.

[30] Marco Dorigo and Thomas Stützle. Ant colony optimization: Overview and recent advances. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 227–263. Springer US, 2010.

[31] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(9): 851–871, 2000.

[32] Michael J.B. Krieger and Jean-Bernard Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots, 2000.

[33] Israel A. Wagner, Michael Lindenbaum, and Alfred M. Bruckstein. Efficiently searching a graph by a smell-oriented vertex process. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):211–223, 1998.

[34] J. Kennedy and R. Eberhart. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, pages 39–43, Nagoya, 1995.

[35] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *SIGGRAPH '87 Conference Proceedings*, pages 25–34, 1987.

[36] F Heppner and U Grenander. A stochastic nonlinear model for coordinated bird flocks. In S Krasner, editor, *Ubiquity Of Chaos*, pages 233–238. Amer assoc advancement science, Amer assoc advancement science, 1990. 1989 Annual meeting of the american assoc for the advancement of science, San Francisco, CA, Jan 14-19, 1989.

[37] Mark M. Millonas. Swarms, phase transitions, and collective intelligence. Working Papers 93-06-039, Santa Fe Institute, Jun 1993. URL http://ideas.repec.org/p/wop/safiwp/93-06-039.html.

[38] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pages 120–127, Honolulu, 2007.

[39] M Clerc and J Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on evolutionary computation*, 6(1):58–73, FEB 2002.

[40] Y. G. Petalas, K. E. Parsopoulos, and M. N. Vrahatis. Memetic particle swarm optimization. *Annals of operations research*, 156(1):99–127, DEC 2007.

[41] Fei Han, Qing-Hua Ling, and De-Shuang Huang. An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks. *Neural Computing and Applications*, 19:255–261, 2010.

[42] T. Aruldoss Albert Victoire and A. Ebenezer Jeyakumar. Hybrid pso-sqp for economic dispatch with valve-point effect. *Electric Power Systems Research*, 71(1):51–59, 2004.

[43] J. Y. Chen, Z. Qin, and Y. et al. Liu. Particle swarm optimization with local search. In *Proceedings of the 2005 International Conference on Neural Networks and Brain*, volume 1-3, pages 481–484, Beijing, 2005.

[44] S. Das, P. Koduru, M. Gui, and et al. Adding local search to particle swarm optimization. In *2006 IEEE Congress on Evolutionary Computation*, volume 1-6, pages 428–433, Vancouver, 2006.

[45] Xi-Huai Wang and Jun-Jun Li. Hybrid particle swarm optimization with simulated annealing. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 4, pages 2402–2405, aug. 2004.

[46] Z. H. Cui, J. C. Zeng, and X. J. Cai. A new stochastic particle swarm optimizer. In *IEEE Congress on Evolutionary Computation*, volume 1-2, pages 316–319, Portland, 2004.

[47] De Freitas Vaz, Antnio Ismael, Pinto Fernandez, and Edite Manuela Da Graa. Optimization of nonlinear constrained particle swarm. *Technological & Economic Development of Economy*, 12(1):30–36, 2006.

[48] Jens Gimmler, Thomas Stützle, and Thomas E. Exner. Hybrid particle swarm optimization: An examination of the influence of iterative improvement algorithms on performance. In Marco Dorigo et al, editor, *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006*, volume 4150 of *Lecture Notes in Computer Science*, pages 436–443. Springer-Verlag, Heidelberg, Germany, 2006.

[49] J. J. Liang and P. N. Suganthan. Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In *IEEE Congress on Evolutionary Computation*, volume 1-6, pages 9–16, Vancouver, 2006.

[50] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore, 2005. URL http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC2005/Tech-Report-May-30-05.pdf.

[51] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore, 2006.

[52] X. H. Hu, R. C. Eberhart, and Y. H. Shi. Engineering optimization with particle swarm. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium (sis 03)*, pages 53–57, Indianapolis, 2003.

[53] Daniel Lichtblau and Eric W. Weisstein. Convex function. URL http://mathworld.wolfram.com/input/?i=convex+function. (Jan 2012).

[54] Angel E. Munoz-Zavala, Arturo Hernandez-Aguirre, Enrique R. Villa-Diharce, and Salvador Botello-Rionda. Peso+ for constrained optimization. In *2006 IEEE congress on evolutionary computation, VOLS 1-6*, IEEE Congress on Evolutionary Computation, pages 231–238, New York, USA, 2006. IEEE, IEEE.

[55] Maurice Clerc. *Particle Swarm Optimization*. ISTE Publishing Company, 2006.

[56] J.J. Liang and P.N. Suganthan. Dynamic multi-swarm particle swarm optimizer with local search. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 522 –528 Vol.1, sept. 2005.

[57] Yann Cooren, Maurice Clerc, and Patrick Siarry. Performance evaluation of tribes, an adaptive particle swarm optimization algorithm. *Swarm Intelligence*, 3:149–178, 2009.

[58] Leticia C. Cagnina, Susana C. Esquivel, and Carlos A. Coello Coello. Solving engineering optimization problems with the simple constrained particle swarm optimizer, 2008.

[59] C. Worasucheep. Solving constrained engineering optimization problems by the constrained pso-dd. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, volume 1, pages 5 –8, may 2008.

[60] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence from Natural to Artificial Systems*. Oxford University Press, 1999.

[61] Marco Dorigo, V. Maniezzo, Alberto Colorni, Marco Dorigo, Marco Dorigo, Vittorio Maniezzo, Vittorio Maniezzo, Alberto Colorni, and Alberto Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1991.

[62] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41, 1996.

[63] Maurice Clerc. When ant colony optimization does not need swarm intelligence, 2000. URL http://clerc.maurice.free.fr/pso/.

[64] M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53 –66, April 1997.

[65] T. Stützle and Holger H. Hoos. Max-min ant syetem and local search for combinatorial optimization problems. In *2nd International conference on metaheuristics MIC97*, 1997.

[66] Thomas Stützle and Holger H. Hoos. Max-min ant system. *Future Gener. Comput. Syst.*, 16:889–914, June 2000.

[67] T. Stützle and H. Hoos. Max-min ant system and local search for the traveling salesman problem. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 309 –314, April 1997.

[68] M. Maur, M. López-Ibañez, and T. Stützle. Pre-scheduled and adaptive parameter variation in max-min ant system. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8, 2010.

[69] Daniel Angus. Niching for ant colony optimisation. In Andrew Lewis et al, editor, *Biologically-Inspired Optimisation Methods*, volume 210 of *Studies in Computational Intelligence*, pages 165–188. Springer Berlin / Heidelberg, 2009.

[70] Daniel Angus. Niching for population-based ant colony optimization. In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, page 115, dec. 2006.

[71] Michael Guntsch and Martin Middendorf. Applying population based aco to dynamic optimization problems. In Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors, *Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 97–104. Springer Berlin / Heidelberg, 2002.

[72] Daniel Angus. Population-based ant colony optimisation for multi-objective function optimisation. In *Proceedings of the 3rd Australian conference on Progress in artificial life*, ACAL'07, pages 232–244, Berlin, Heidelberg, 2007. Springer-Verlag.

[73] D. Angus. Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*, pages 333 –340, april 2007.

[74] Michael Guntsch and Martin Middendorf. A population based approach for aco. In *Proceedings of the Applications of Evolutionary Computing on EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN*, pages 72–81, London, UK, UK, 2002. Springer-Verlag.

[75] Benjamín Barán and Osvaldo Gómez. Omicron ACO. A new ant colony optimization algorithm. *CLEI Electron. J*, 8(1): 1–8, 2005.

[76] Michael Guntsch. *Ant Algorithms in Stochastic and Multi-Criteria Environments*. PhD thesis, University of Karlsruhe, 2004.

[77] Sabrina M. Oliveira, Mohamed Saifullah Hussin, Thomas Stuetzle, Andrea Roli, and Marco Dorigo. A detailed analysis of the population-based ant colony optimization algorithm for the tsp and the qap. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, GECCO '11, pages 13–14, New York, NY, USA, 2011. ACM.

[78] Daniel Angus. *Niching Ant Colony Optimisation*. PhD thesis, Swinburne University of Technology, Melbourne, Australia, 2008.

[79] Daniel Angus and Clinton Woodward. Multiple objective ant colony optimisation. *Swarm Intelligence*, 3:69–85, 2009.

[80] Carlos García-martínez, Oscar Cordón, and Francisco Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. *European Journal of Operational Research*, 180:116–148, 2007.

[81] Kangshun Li, Fumei Xu, Ping Huang, and Wensheng Zhang. A new best-worst ant system with heuristic crossover operator for solving tsp. *International Conference on Natural Computation*, 4:92–97, 2009.

[82] C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(2):1161 –1172, april 2004.

[83] M. Birattari, P. Pellegrini, and M. Dorigo. On the invariance of ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 11:732, 2007.

[84] Thomas Stützle, Manuel López-Ibáñez, Paola Pellegrini, Michael Maur, Marco A. Montes de Oca, Mauro Birattari, and Marco Dorigo. Parameter adaptation in ant colony optimization. Technical Report TR/IRIDIA/2010-002, IRIDIA, Université Libre de Bruxelles, Belgium, January 2010.

[85] Paola Pellegrini, Thomas Stützle, and Mauro Birattari. Off-line vs. on-line tuning: A study on max–min ant system for the tsp. In Marco Dorigo et al, editor, *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 239–250. Springer Berlin, Heidelberg, 2010.

[86] Tiankun Li, Wanzhong Chen, Xin Zheng, and Zhuo Zhang. An improvement of the ant colony optimization algorithm for solving travelling salesman problem (tsp). In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1 –3, sept. 2009.

[87] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufmann, 2002.

[88] Prasanna Balaprakash, Mauro Birattari, and Thomas Stützle. Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In Bartz-Beielstein et al, editor, *Hybrid Metaheuristics*, volume 4771 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin / Heidelberg, 2007.

[89] Daniela Favaretto, Elena Moretti, and Paola Pellegrini. On the exploitative behaviour of max-min ant system. In *Proceedings of the Second International Workshop on Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, SLS '09, pages 115–119, Berlin, Heidelberg, 2009. Springer-Verlag.

[90] Liangjun Ke, Zuren Feng, Zhigang Ren, and Xiaoliang Wei. An ant colony optimization approach forthemultidimensional knapsack problem. *Journal of Heuristics*, 16:65–83, 2010.

[91] Paola Pellegrini, Elena Moretti, and Daniela Favaretto. Exploration in stochastic algorithms: An application on max-min ant system. Working Papers 169, Department of Applied Mathematics, University of Venice, October 2008. URL http://ideas.repec.org/p/vnm/wpaper/169.html.

[92] Paola Pellegrini and Andrea Ellero. The small world of pheromone trails. In *Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence*, ANTS '08, pages 387–394, Berlin, Heidelberg, 2008. Springer-Verlag.

[93] Hugues Bersini, Marco Dorigo, Stefan Langerman, Gregory Seront, and Luca Maria Gambardella. Results of the first international contest on evolutionary optimisation (1st iceo). In *International Conference on Evolutionary Computation*, pages 611–615, 1996.

[94] P. Lalbakhsh, B. Zaeri, and M.N. Fesharaki. Nonlinear ant system for large scale search spaces. In *Fuzzy Information Processing Society (NAFIPS), 2010 Annual Meeting of the North American*, pages 1 –6, 2010.

[95] Hongtao Bai, Dantong OuYang, Ximing Li, Lili He, and Haihong Yu. Max-min ant system on gpu with cuda. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 801 –804, Dec 2009.

[96] Marco Dorigo, Mauro Birattari, Thomas Stützle, Université Libre, De Bruxelles, and Av F. D. Roosevelt. Ant colony optimization artificial ants as a computational intelligence technique. *IEEE Comput. Intell. Mag*, 1:28–39, 2006.

[97] Thomas Stützle, Manuel López-Ibáñez, Marco Dorigo, James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith. *A Concise Overview of Applications of Ant Colony Optimization*. John Wiley & Sons, Inc., 2010.

[98] Kokolo Ikeda and Shigenobu Kobayashi. Ga based on the uv-structure hypothesis and its application to jsp. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 273–282, London, UK, 2000. Springer-Verlag.

[99] Thomas Stützle. URL http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html. Version 1.0, Developed in ANSI C under Linux (Oct 2011).

[100] Tsplib95. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. (Nov 2010).

[101] Portgen tsp instance generator from the 8th dimacs implementation challenge. URL http://www2.research.att.com/~dsj/chtsp/codes.zip. (Sep 2010).

[102] Keld Helsgaun. Lkh version 2.0.5. URL http://www.akira.ruc.dk/~keld/research/LKH/. (Feb 2012).

[103] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[104] URL http://gcc.gnu.org/onlinedocs/gfortran/RANDOM_005fNUMBER.html. (Jan 2011).

[105] Y. Nagata. Niching method for combinatorial optimization problems and application to jsp. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2822 – 2829, july 2006.

[106] Shih-Pang Tseng, Chun-Wei Tsai, Ming-Chao Chiang, and Chu-Sing Yang. A fast ant colony optimization for traveling salesman problem. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –6, 2010.

[107] David S. Johnson and Lyle A. Mcgeoch. The traveling salesman problem: A case study in local optimization. In *Local Search in Combinatorial Optimization*. 1997.

[108] Paola Pellegrini and Elena Moretti. A computational analysis on a hybrid approach: Quick-and-dirty ant colony optimization. *Applied Mathematical Sciences*, 3:1127–1140, 2009.

[109] Christian Blum Manuel Lpez-Ibeza. Beam-aco for the travelling salesman problem with time windows. *Computers & Operations Research*, 37:1570, 2009.

[110] H. Duan and Xiufen Yu. Hybrid ant colony optimization using memetic algorithm for traveling salesman problem. In *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*, pages 92 –95, 1-5 2007.

[111] Yuren Zhou. Runtime analysis of an ant colony optimization algorithm for tsp instances. *Evolutionary Computation, IEEE Transactions on*, 13(5):1083 –1092, oct. 2009.

[112] Christian Blum and Marco Dorigo. Deception in ant colony optimization. In Marco Dorigo et al, editor, *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5 - 8, 2004, Proceedings*, volume 3172 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2004.

[113] Max Manfrin, Mauro Birattari, Thomas Stützle, and Marco Dorigo. Parallel ant colony optimization for the traveling salesman problem. In Marco Dorigo et al, editor, *Ant Colony Optimization and Swarm Intelligence*, volume 4150 of *Lecture Notes in Computer Science*, pages 224–234. Springer Berlin / Heidelberg, 2006.

[114] A. Acharya, D. Maiti, A. Banerjee, R. Janarthanan, and A. Konar. Extension of max-min ant system with exponential pheromone deposition rule. In *Advanced Computing and Communications, 2008. ADCOM 2008. 16th International Conference on*, pages 1 –8, dec 2008.

[115] Michalis Mavrovouniotis and Shengxiang Yang. Ant colony optimization with direct communication for the traveling salesman problem. In *Computational Intelligence (UKCI), 2010 UK Workshop on*, pages 1 –6, 2010.

[116] M. Imani, E. Pakizeh, M.M. Pedram, and H.R. Arabnia. Improving max-min ant system performance with the aid of art2-based twin removal method. In *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pages 186 –193, july 2010.

[117] Song Zheng, Ming Ge, Chunfu Li, Chunlin Wang, and Anke Xue. New transition probablity for ant colony optimization: global random-proportional rule. In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pages 2698 –2702, july 2010.

[118] Luca Maria Gambardella and Marco Dorigo. An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS J. on Computing*, 12:237–255, July 2000.

[119] Manuel López-Ibá nez and Thomas Stützle. An analysis of algorithmic components for multiobjective ant colony optimization: A case study on the biobjective tsp. In *Artificial Evolution'09*, pages 134–145, 2009.

[120] Silvia Mazzeo and Irene Loiseau. An ant colony algorithm for the capacitated vehicle routing problem. *Electronic Notes in Discrete Mathematics*, vol 18:181–186, 2004.

[121] Marc Reimann, Karl Doerner, and Richard F. Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563 – 591, 2004.

[122] Marc Reimann, Karl Doerner, and Richard F. Hartl. Analyzing a unified ant system for the vrp and some of its variants. In *Proceedings of the 2003 international conference on Applications of evolutionary computing*, EvoWorkshops'03, pages 300–310, Berlin, Heidelberg, 2003. Springer-Verlag.

[123] Daniel Merkle and Martin Middendorf. An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In *Proceedings of EvoWorkshops 2000, volume 1803 of LNCS*, pages 287–296. Springer Verlag, 2000.

[124] A. Bauer, B. Bullnheimer, R.F. Hartl, and C. Strauss. An ant colony optimization approach for the single machine total tardiness problem. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 3 vol. (xxxvii+2348), 1999.

[125] E. Pérez, F. Herrera, and C. Hernández. Finding multiple solutions in job shop scheduling by niching genetic algorithms. *Journal of Intelligent Manufacturing*, 14:323–339, 2003.

[126] Sjoerd Van Der Zwaan and Carlos Marques. Ant colony optimisation for job shop scheduling, 1999.

[127] Thomas Stützle, Fachgebiet Intellektik, Fachbereich Informatik, and Technische Hochschule Darmstadt. An ant approach to the flow shop problem. In *In Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, pages 1560–1564. Verlag, 1997.

[128] Chandrasekharan Rajendran and Hans Ziegler. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, 155(2):426–438, 2004. Financial Risk in Open Eeonomies.

[129] Kuo-Ling Huang and Ching-Jong Liao. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Comput. Oper. Res.*, 35:1030–1046, April 2008.

[130] Xiao ian Zhuo, Jun Zhang, and Wei neng Chen. A new pheromone design in acs for solving jsp. In *IEEE Congress on Evolutionary Computation*, pages 1963–1969, 2007.

[131] Kenneth D. Boese. Cost versus distance in the traveling salesman problem. Technical report, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1995.

[132] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.

[133] Manuel López-Ibáñez and Thomas Stützle. The impact of design choices of multiobjective antcolony optimization algorithms on performance: an experimental study on the biobjective tsp. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 71–78, New York, NY, USA, 2010. ACM.

[134] Manuel López-Ibáñez and Thomas Stützle. Automatic configuration of multi-objective aco algorithms. In *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, pages 95–106, Berlin, Heidelberg, 2010. Springer-Verlag.

[135] 8th dimacs implementation challenge. URL http://www2.research.att.com/~dsj/chtsp/. (Nov 2010).

[136] R.E. Burkard, E. ELA, S.E. Karisch, and F. Rendl. Qaplib - a quadratic assignment problem library. URL http://www.seas.upenn.edu/qaplib/. (Jan 2011).

[137] L. M. Gambardella, É. D. Taillard, D. Taillard, and M. Dorigo. Ant colonies for the qap, 1997.

[138] David Corne et al, editor. *New ideas in optimization*. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[139] J.E.Beasley. Or-library. URL http://people.brunel.ac.uk/~mastjjb/jeb/info.html. (Feb 2011).