Chapter 1

Introduction

Many actions occurring in nature and human existence incorporate some form of optimization. Species survive by continually optimizing parameters describing their environment. Various strategies are employed to ensure the survival of the species, for example, some animals aggregate in flocks, herds and schools. Such structures serve as a defense against predators, are instrumental in locating food sources, and ensure accurate navigation on migration. To survive, individuals in a flock, herd or school are not only dependent on their own devices, but can also rely on the collective knowledge of the entire population.

Particle swarm optimization (PSO) is a population-based optimization strategy inspired by the behaviour of bird flocks that wheel and swoop in unison. Personal as well as social memory are employed to guide the movement of particles, collectively referred to as a swarm, in order to converge on a better position. PSO has been proved to be an effective, efficient and robust optimization method for solving a wide array of single-solution problems including neural network training [96] [98] and function minimization [19] [28] [48] [49]. Furthermore, the algorithm is simple to implement, and requires no gradient information which is problem-dependent and may be difficult to calculate. PSO is particularly useful when the overall optimal solution needs to be located in a problem space containing many sub-optimal solutions. The entire search space is explored, thus preventing the swarm from settling on a suboptimal solution.

While the standard PSO locates the overall best position in the search space, many problems require the location of sub-optima as well. The quality of such solutions often differ very slightly from one another or may even be similar. The user then has a choice between multiple, equally acceptable solutions in a single search space. Population-based algorithms designed

to locate multiple optima, also glean their terminology from nature. In natural ecosystems, animals survive in different ways, and different species evolve to fill each role, referred to as an ecological niche. In PSO, a species is represented by a subswarm. Algorithms designed to locate multiple optima are referred to as *niching* or *speciation* algorithms.

Originally, genetic algorithms were adapted for niching [4] [40] [56]. Since the inception of the PSO paradigm, attempts have been directed towards the development of PSO niching algorithms, and modifications to the standard PSO were put in place to facilitate niching. To this end, various strategies have been applied, for example, objective function stretching [72] where the function landscape is modified each time a position is discovered where the function has a minimum value, the species-based PSO [57] that uses a problem-dependent niche radius parameter set in advance, nBest [12] that redefines the fitness function for multiple solutions, and NichePSO [13] [14] that has a high success rate, but requires finely-tuned parameters. A number of other algorithms yielding good results have been developed [5] [41] [58], but often these algorithms are computationally complex. The need for a simple, elegant and efficient niching algorithm with minimal user-specified parameters and low complexity that would be robust enough to be used in a large variety of situations, was identified. In addition, the idea to exploit the basic characteristics of the PSO paradigm to facilitate niching, was ultimately appealing. Thus, the purpose of this thesis is the development of a niching method encompassing these characteristics as far as possible.

This thesis investigates existing niching strategies that have been developed using the particle swarm paradigm. A novel niching algorithm requiring minimal knowledge of the search space, is developed, tested, and compared with existing algorithms. The algorithm is extended to incorporate dynamic optimization problems where multiple optima are tracked in changing environments.

1.1 Motivation

Single-solution particle swarm optimizers are relatively easy to implement, and a large number of PSO variations have been developed for various problem types. However, since the number of optima in the search space, as well as the shapes and sizes of the peaks forming the function landscape are usually unknown, PSO algorithms that locate multiple optima are much more complicated. Such multimodal PSO algorithms often require prior knowledge of the search space to produce significant results. Although some parameters have to be set in advance to facilitate niching, the challenge to develop a PSO niching algorithm requiring minimal prior

knowledge of the function landscape, remained. An additional challenge comprised using and extending the principles on which the original PSO algorithm rests to facilitate niching, resulting in an efficient and elegant solution. Such a strategy should also be robust enough to operate successfully in convoluted function landscapes where the shapes, sizes, and placing of niches differ considerably.

1.2 Objectives

The main objectives of this thesis can be summarized as follows:

- To extend the essential building blocks of the basic PSO paradigm to facilitate niching.
- To develop an efficient, elegant, and robust niching algorithm that identifies candidate solutions in an unknown problem space, calculate niche boundaries, and optimize subswarms occupying the niches to yield multiple optima.
- To ensure that the algorithm requires minimal prior knowledge of the function landscape.
- To compare different niching approaches using a diverse problem set.
- To conduct a scalability study of the niching algorithm.
- To extend the niching algorithm to incorporate dynamic optimization problems.

1.3 Methodology

The main purpose of this thesis is the development of a novel strategy to induce niching in multimodal function landscapes. The assumption that the vector dot product of the social and cognitive direction vectors can be used to determine niche boundaries is explained and motivated. A technique to calculate separate niche radii and form subswarms occupying the niches, is presented. Three algorithms that have been developed consecutively, are presented. The sequential vector-based PSO identifies niches sequentially and optimizes the subswarms contained in the niches in turn. The parallel vector-based PSO also identifies niches sequentially, but subswarms occupying the niches are optimized in parallel while duplicate subswarms are merged at specific intervals. The enhanced vector-based PSO includes a strategy to prohibit unnecessary merging of niches.

All three algorithms are tested for a number of one-dimensional and two-dimensional benchmark functions with well-known characteristics, using a system-supplied random number generator as well as Sobol sequences to calculate initial particle positions. Selected benchmark functions were also used to test the sensitivity of the algorithm for a range of granularity values. An upper bound for the granularity was derived for each function.

A scalability study was undertaken relating the required swarm size to the number of optima in a specified search space. Three scalable benchmark functions were used. For each setting the function was tested with a range of swarm sizes, and optimal swarm sizes were deduced from graphs of the results. A linear relation between optimal swarm size and number of optima was established for each of the three functions.

To assess the performance of the vector-based PSO in comparison to that of two other niching algorithms, NichePSO and the species-based PSO, all three algorithms were tested on seven two-dimensional functions using the same settings. The functions were chosen to represent a range of landscapes in order to observe how specific situations impacted on the performance of the algorithms.

The vector-based PSO was extended to locate and track optima in changing environments. The moving peak benchmark [25] was used to set up a number of scenarios including spatial movement of 3 and 5 peaks across a defined two-dimensional search space, variation of the value of the function at stationary positions, peaks that are obscured during movement, and new peaks that appear at unknown positions. Each of these situations were used to test the performance of the dynamic vector-based PSO. To conclude, the influence of severity (the spatial change in the position of the optimum) on performance was tested. The movement of one peak across a landscape containing three peaks was tracked using a range of severity values to deduce an upper bound for the severity in a specific scenario.

1.4 Contributions

The main contributions of this thesis are:

- Proposing a strategy to address the problem of finding niche radii in a multimodal function landscape. Niches could be located and demarcated with minimal prior knowledge of the objective function.

- Developing a niching algorithm for particle swarm optimization that uses the above strategy and compares well to other niching paradigms.

- An empirical analysis of state-of-the-art PSO niching methods.
- Extending the niching algorithm to locate and track multiple optima in dynamic environments.

1.5 Thesis outline

Chapter 2 presents an overview of various aspects of optimization. A brief discussion of mathematical optimization is followed by different viewpoints on optimization in nature, and how it is related to the development of population-based optimization algorithms. To conclude, the main features of evolutionary computation and its various forms are discussed.

Chapter 3 describes the inception and development of the original particle swarm optimization paradigm. Several aspects concerning the improvement and refinement of the algorithm are reviewed. Particle trajectories and the concept of neighbourhoods in a swarm are briefly discussed. A classification of single-solution PSO algorithms is given followed by the description of a number of relevant strategies. Information sharing strategies and subswarm-based algorithms are emphasized.

Chapter 4 describes and elaborates on the concept of niching or speciation. Niching algorithms locate more than one optimum of an objective function in a search space. Genetic algorithm niching techniques are briefly reviewed. A number of techniques that adapt particle swarm optimization to locate and optimize functions with multiple optima are discussed in detail. These algorithms use different strategies to identify candidate solutions, estimate niche boundaries, and optimize separate subswarms to converge on different optima. Algorithms where these processes take place sequentially as well as in parallel, have been developed. In the case of parallel niching algorithms, a merging component is incorporated. Improvements and refinements of some of the algorithms are also discussed.

Chapter 5 presents the development of a new niching strategy, the vector-based particle swarm optimizer (VBPSO). The principles underlying this approach are discussed, and three consecutive versions of the algorithm are described in detail. Extensive test results obtained by applying the various versions of the algorithm to a number of one- and two-dimensional functions are presented in chapter 6. The final version of the VBPSO is analysed by empirically testing the sensitivity of the algorithm for different parameter values, its ability to scale to functions in higher dimensions, and the relationship between the initial swarm size and the algorithm's performance. Chapter 6 is concluded by presenting the results of a compara-

tive study of the performance of three diverse PSO niching algorithms, namely the VBPSO, NichePSO [13] [14] and the species-based PSO [57]. These algorithms are applied to a number of two-dimensional functions with varying characteristics.

Chapter 7 investigates the behaviour of the vector-based particle swarm optimization paradigm in a changing environment. The positions as well as the quality of multiple optima can change dynamically over time. The vector-based PSO that locates multiple optima in static environments, is adapted and extended to track multiple optima over time. In chapter 8 a number of scenarios illustrating a variety of dynamic changes is set up and tested. The chapter concludes with an investigation of the influence of severity on dynamic changes.

A summary of the developments and results obtained in this thesis, is presented in chapter 9.

Appendix A presents empirical results of tests that were run to determine specific settings used by the vector-based PSO algorithms. These settings include minimum subswarm sizes, merging interval sizes and minimum sizes of subswarms capable of tracking moving optima.

A list of publications is presented in appendix B. These papers have led to, or are derived from the work presented in this thesis.

Chapter 2

Optimization

This chapter presents an overview of various aspects of optimization. A brief discussion of mathematical optimization is followed by different viewpoints on optimization in nature and how natural phenomena relate to the development of population-based optimization algorithms. To conclude, the main features of evolutionary computation and its various forms are discussed.

2.1 Introduction

The term **optimization** is derived from the Latin root *optimus* meaning *best*. The optimum refers to the best or most favourable set of conditions. Therefore, optimization can be described as the process that has to be followed in order to obtain a set of conditions where a specific desirable feature will have the best or most effective possible value within a certain environment.

Problems for which the solution lies in finding an optimal value for some objective, are known as *optimization problems*. Typically, an optimization problem has many candidate solutions, each represented by a set of possible conditions. Depending on the characteristics and requirements of the problem, an acceptable solution should be found, which may be the best overall solution, or one of a set of feasible solutions.

Many optimization problems can be found in everyday life, ranging from navigating through rush-hour traffic to cost-effective economic policies, production scheduling in manufacturing, and process optimization in large petro-chemical plants.

Optimization is commonly used in a mathematical context, where it refers to the minimization or maximization of mathematical functions by choosing, by means of some strategy, values from within an allowed set in order to find the best solution.

However, optimization is a much broader concept. In order to survive, man had to find ways and means to overcome obstacles in a hostile environment. Strategies had to be devised to maximize the availability of food sources and favourable conditions for procreation, while minimizing any danger to the continued existence of the species. Going even further back, plant and animal species have evolved over millions of years by adapting to their environment in the most profitable manner. Features have been developed to facilitate finding food, defense against predators, and procreation of the species.

The remainder of this chapter is organized as follows: Classical mathematical optimization is briefly reviewed in section 2.2. Basic concepts, problem classification, and mathematical optimization methods are discussed. Section 2.3 covers stochastic local search algorithms, while some background on optimization in nature is given in section 2.4. The chapter is concluded by a brief summary of evolutionary algorithms in section 2.5.

2.2 Mathematical optimization

Classical mathematical optimization methods form a necessary background to any study involving some form of optimization. Formal definitions of basic concepts and maximization are given, followed by the classification of optimization problems according to specific features. A subset of the most important mathematical optimization methods, divided into derivative and non-derivative methods, are discussed.

2.2.1 Basic concepts

Mathematical optimization can be described formally as the process of

1. the *formulation* and
2. the *solution* of a constrained optimization problem of the form

$$minimize\ f(\mathbf{x}),\ \mathbf{x} = [x_1, x_2, \ldots, x_n]^\tau \in \Re^n \tag{2.1}$$

subject to the constraints:

$$g_j(\mathbf{x}) \leq 0,\ j = 1, 2, \ldots, m$$

$$h_j(\mathbf{x}) = 0,\ j = 1, 2, \ldots, r$$

where $f(\mathbf{x})$, $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$ are scalar functions of the vector $\mathbf{x}$ and τ refers to some tolerance [93].

The components of equation (2.1) can be described as follows:

- The *objective function*, $f(\mathbf{x})$, represents a mathematical formulation of the problem to be optimized.
- The variables, x_i (the design variables), affects the value of the objective function.
- The inequality constraints, $h_j(x)$, and equality constraints, $g_j(x)$, restrict values that can be assigned to the variables. If no constraints or only boundary constraints are specified, the problem is referred to as an unconstrained minimization problem.

2.2.2 Maximization

Maximization of a function can be effected by modifying the standard form given in equation (2.1) as follows:

$$maximize\ f(\mathbf{x}) = minimize\ (-f(\mathbf{x})) \tag{2.2}$$

Any inequality constraints should then be modified accordingly.

Both minima and maxima can be referred to as optima, depending on how the problem is described and how the optimization process is implemented.

2.2.3 Problem classification

Optimization problems can be classified according to specific features. The choice of an optimization algorithm will, to a large extent, depend on the particular type of problem, and should be appropriate for the specific application. The following points should be taken into consideration when deciding on a strategy.

Linear versus non-linear optimization problems: For some optimization problems the objective as well as all the constraints, are linear functions [93] [98]. These problems are called *linear programming problems*. Figure 2.1 depicts a linear programming problem in two dimensions. Efficient techniques have been developed to solve these problems, one of the most famous being the simplex method proposed by Dantzig in 1947 [93].

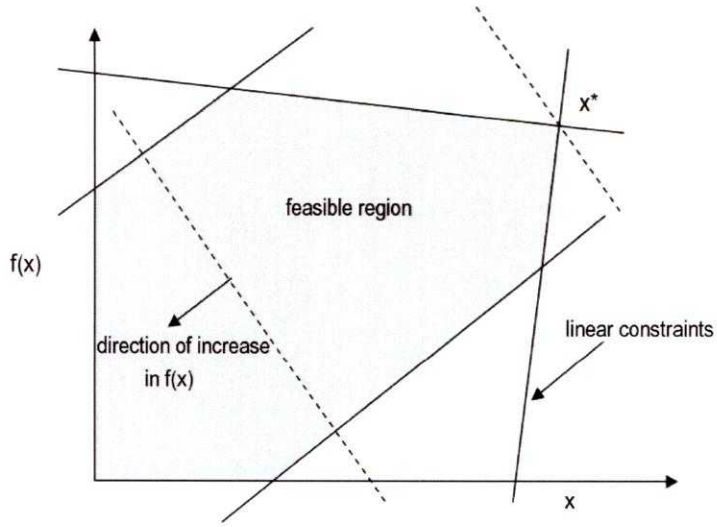

Figure 2.1: Two-dimensional linear programming problem

Non-linear optimization problems are generally more difficult to solve, and numerous techniques have been developed for this purpose.

Dimensionality of the objective function: The number of variables in the objective function may vary from one to several. Figure 2.2 and Figure 2.3 depict functions of one and two variables with single minima at $\mathbf{x}^*$.

Local and global optima: Some optimization problems are such that more than one optimum exists in the search space. The values of the objective function at these positions may differ significantly, differ very slightly, or can even be the same. The optimum with the best value is the global optimum, while the other optima are referred to as local optima. More formally, given that optima imply *minima*, local and global minimizers are defined as follows: A *local minimizer*, $\mathbf{x}^*_B$, of the region B, is defined so that

$$f(\mathbf{x}^*_B) \leq f(\mathbf{x}), \forall \mathbf{x} \in B \tag{2.3}$$

where $B \subset S \subseteq \Re^n$ and S denotes the search space.

A *global minimizer*, $\mathbf{x}^*$ is defined so that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in S \tag{2.4}$$

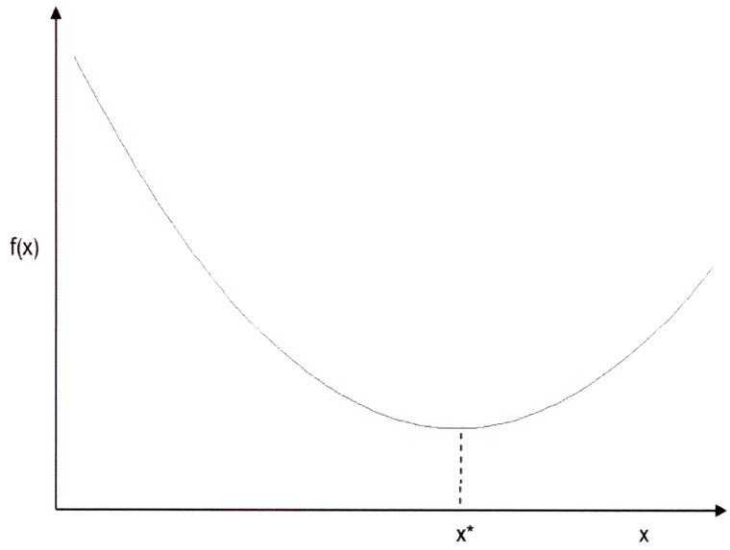

Figure 2.2: Function of single variable with minimum at x^*

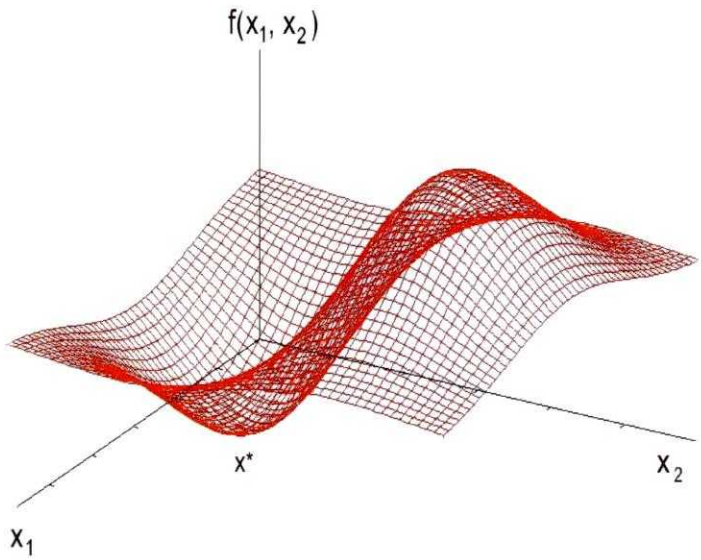

Figure 2.3: Function of two variables with minimum at x^*

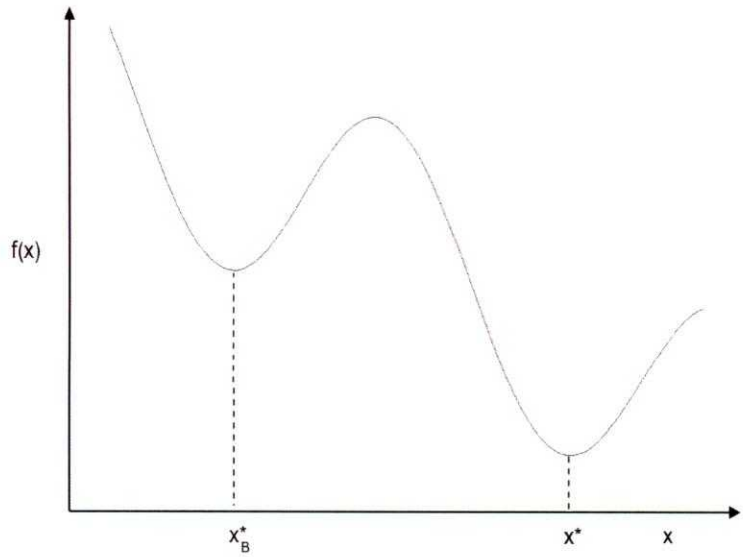

Figure 2.4: Function of a single variable with global minimum at x^* and local minimum at x^*_B.

where S is the search space.

Figure 2.4 depicts an objective function with a global minimum and one local minimum.

Constraint functions: Optimization problems can be divided into *unconstrained* and *constrained* problems. Unconstrained minimization takes place when an objective function is optimized over the entire search space. Although the boundaries of the search space are also constraints, these problems are referred to as unconstrained optimization problems.

With constrained optimization problems, one or more constraint functions are defined to restrict the region where optimization should take place. If $g(\mathbf{x})$ denotes the inequality constraint function, such that $g(\mathbf{x}) \leq 0$, the contour $g(\mathbf{x}) = 0$ divides the search space into a *feasible region* and an *infeasible region* [93] [98].

In the case of the equality constraint function, $h(\mathbf{x})$, the feasible values will only be found on the contour or surface where $h(\mathbf{x}) = 0$.

Multi-solution optimization: An objective function may, in addition to global optima, contain several local optima. As these local optima may represent alternative feasible so-

lutions, it is often necessary to find all or most solutions in a particular search space. Multi-solution optimization is also referred to as *multi-modal optimization.*

Multi-objective optimization: The simultaneous optimization of several objective functions is necessary in many real-world problems, where objectives may work against each other [33]. The result is a set of optimal solutions representing trade-offs between different, often conflicting, objectives.

Optimization in dynamic environments: Most real-world optimization has to be carried out in a dynamic environment, that is, the objective function changes over time. Such environments require algorithms that locate and track changing optima. Single or multiple solutions can be tracked, depending on the problem requirements.

This thesis focuses on multi-solution optimization, using particle swarm optimization. Algorithms are developed to locate multiple solutions in a search space. These algorithms are also adapted to track multiple optima in a dynamic environment. Throughout the thesis, only unconstrained, single-objective functions are considered.

2.2.4 Mathematical optimization methods

Numerous methods for solving optimization functions of many variables have been developed, tested and applied to practical problems [93]. It is often claimed that some optimization techniques are superior to others, a statement that is contradicted by the "No Free Lunch" (NFL) theorem of Wolpert and Macready [103]. This theorem states that all optimization techniques have the same behaviour over all $f : X \rightarrow Y$ where X and Y are finite sets. However, differences in the behaviour of optimization techniques have been observed [26]. For several optimization scenarios specific techniques have been found to be superior to general ones. It can also be reasoned that although the NFL theorem has been proved for the set of *all* functions, it does not necessarily hold for all subsets of this set [98]. Most real-world functions have some structure and compact descriptions, forming a small subset of all functions. Therefore the choice of a method depends on the specific problem and will, to a large extent, determine the efficiency and accuracy of the solution. The advent of the computer has played an important part in the development of numerical methods using an iterative approach to find better approximations to the optimum until the desired accuracy is achieved.

A primary method to find local extrema of a function is to locate points where the derivative of the function is zero, provided that the function is differentiable. An alternative method is

to evaluate the function many times and search for a local minimum. Therefore, numerical optimization methods can be divided into derivative and non-derivative methods.

Non-derivative methods

Non-derivative optimization methods are methods that do not require any derivative of the objective function. For a non-derivative optimization method to be efficient, the number of function evaluations should be reduced while an accurate solution must be reached in reasonable time. Two well-known and efficient methods are discussed below [64]:

The golden ratio method: This method is suitable for functions that are unimodal, that is, one minimum occurs in a specific interval. The method works as follows: Assuming that the initial interval is [0,1], it is divided into three subintervals [0,1-r], [1-r,r] and [r,1]. Depending on whether $f(x)$ is decreasing or increasing at the interval boundaries, a new interval, consisting of the two leftmost or the two rightmost subintervals, is formed and this smaller interval is divided into three subintervals. The process is repeated until the boundaries of the subinterval where the minimum occurs, differ less than a given error margin, in which case the minimum can be approximated.

The Nelder-Mead method: Nelder and Mead devised a simplex method to find the minimum of a function of n variables [67]. The term *simplex* describes a generalized triangle in n dimensions. For two variables, a simplex will be a triangle. Initially, function values at the three vertices are found. The search generates a sequence of triangles that are formed by replacing the worst vertex of each triangle with a new vertex in such a way that the function values at the vertices become smaller while approaching the minimum point.

Derivative methods

Derivative optimization or direct-search methods use the derivative of the objective function at certain points in the search space to direct the search for the position where the function has a minimum or maximum value.

A large number of direct-search algorithms for unconstrained minimization of functions have been developed. All such algorithms require an initial estimate of the optimum [93]. From this starting point a sequence of estimates $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$, are generated. Assuming the optimum is

a minimum point, search for a better estimate takes place in the direction of *descent*; that is, where the derivative of the function becomes smaller. The process is repeated until no further descent is possible, or if the condition for a minimum, $\nabla f(\mathbf{x}) = 0$, is sufficiently accurately satisfied in which case $x^* \approx 0$.

For smooth functions, a sub-class of the direct search methods, namely the *line search* descent methods, is suitable. The direction of descent u^{i+1} is selected at each iteration x^i such that

$$\frac{df(\mathbf{x}^i)}{d\lambda}|_{u^{i+1}} = \nabla^\tau f(\mathbf{x}^i)u^{i+1} < 0 \tag{2.5}$$

where λ indicates the minimizer, ∇ is the gradient and τ refers to some tolerance.

Within the sub-class of line search descent methods, a number of methods exist that differ in the way in which the descent directions, u^i, are chosen, as well as the way in which the line search is performed.

The *method of steepest descent* is a simple and well-known method for the minimization of a function of n variables. It was first proposed by Cauchy in 1847 [93] [64]. This method is a *first order method* because first order partial derivatives are employed to compute the search direction at each iteration. For an n-dimensional function, $f(\mathbf{x})$, partial derivatives are evaluated in turn, each time starting with the previous minimum and changing the search direction. Iteration will produce a sequence, $\{\mathbf{x}_k\}$, of points with the property $f(\mathbf{x}_0) > f(\mathbf{x}_1) > \ldots > f(\mathbf{x}_k) > \ldots$. If $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_B^*$ then $f(\mathbf{x}_B^*)$ will be a local minimum for f.

2.3 Stochastic local search algorithms

Derivative and non-derivative optimization methods depend, to a large extent, on a well-chosen starting point. Roughly, these methods can be classified as local methods, as the search process is initialized at some point in the search space and proceeds by iteratively moving from one location to a neighbouring one depending on local knowledge only [44]. In contrast to systematic search algorithms where the search space is traversed systematically, there is no guarantee that a local search algorithm will find a solution. Local search methods can also visit the same location more than once or get stuck in some part of the search space from which they cannot escape.

Local search algorithms may generally seem inferior to systematic algorithms, but due to time constraints, using systematic algorithms is often not viable. Ideally, search algorithms should deliver reasonably good solutions within a limited time. However, depending on the

problem, a good solution may be reached by combining local and systematic algorithms.

Local search algorithms often make use of stochastic mechanisms and are then known as *stochastic local search* algorithms. These algorithms have been proved useful for solving both NP-complete decision problems and NP-hard combinatorial problems [44]. The concept of hill-climbing, as well as algorithms such as simulated annealing [54] and tabu search [38] can be classified as using stochastic search mechanisms. In addition, population-based algorithms, finding their inspiration in the behaviour of natural populations, all use some form of stochastic search.

2.4 Optimization in nature

Optimization is not a new concept. Inhabitants of the world in which we live have evolved over millions of years into populations best suited to survive in their various environments. The amazing diversity of these populations and the myriad of ingenious ways in which they have adapted to their environment suggest some stochastic processes as well as the principle that came to be known as survival of the fittest [22]. To ensure their survival, these populations had to optimize the parameters describing their existence. Survival strategies have evolved in these populations in different ways. Some animals and insects have developed special features to protect themselves against predators and utilize their environment to their best advantage. Other creatures, especially birds and insects, form colonies, nests or swarms. It is from these, often highly specialized systems, that researchers gleaned their inspiration to develop optimization methods that model the way in which the systems work, as perceived by the researcher. The optimization system is usually more simplified than the natural system it purports to model - natural systems being much more complicated and not always well understood by man. However, this does not prevent the researcher from building an artificial system with the broad principles observed in the natural system as a starting point.

Many of these artificial systems are population-based, having found their inspiration in the behavioural patterns of populations of animals like flocks of birds, schools of fish, and termite and ant colonies. Interesting theories of group behaviour and the forces that seem to govern it, have been formulated over the years [63] [75] [80]. Some of the first studies in the behavioural patterns of animals and insects in their natural environment were undertaken in the early years of the 20th century by Eugène Marais, a South African writer, lawyer and naturalist [63]. Among others, the behaviour of termites inhabiting large termite nests, were carefully

observed and documented. Experiments suggested that the queen influences the movements of all the termites, even when they are far away from the nest. On the other hand, synchronous movements of swarms of locusts were observed where no central influence could be identified. Marais called this phenomenon the "group soul", an influence keeping the group or swarm together even if no fixed centre exists.

Different explanations for the highly coordinated movements of flocks of birds or schools of fish were put forward. Research done by Wayne Potts in 1984 showed that maneuver waves spread through a flock of birds much faster than a single bird's mean reaction time [75]. Potts called this the "chorus line hypothesis" and attributed it to an individual anticipating an approaching wave.

In 1986 Craig Reynolds showed that complex behaviour can be generated by applying simple rules. He illustrated the concept by creating "boids", computer-generated flocking organisms [80]. Three basic laws govern the movement of each boid, namely separation, alignment and cohesion. A mechanism was thus developed to create computer-generated flocks of creatures for use in films and multi-media. A current popular view is to attribute the phenomenon to emergent behaviour where the flock's movements are determined by decisions of individual birds following simple rules when responding to movements of their neighbours.

Many advantages for being part of a flock exist. A flock's collective intelligence will serve as a defense against predators, find food sources more easily and avoid being lost, while moving in groups even reduces energy expenditure as birds are able to glide more often. Therefore, it can be concluded that group behaviour can be seen as a form of optimization. The best food sources, adequate defense, and profitable energy expenditure all ensure the survival of the group as an entity, while the survival of the group, swarm, or nest takes precedence over the fate of the individual.

In general, computer systems can now be built where collections of independently acting entities exhibit collectively intelligent behaviour in an environment that these entities can sense and alter. This behaviour is known as swarm intelligence, an example of biology as a source for computing.

2.5 Computing paradigms inspired by natural optimization processes

Swarm intelligence forms only one branch of a class of computing paradigms that owe their existence to the observation and study of natural biological systems. The field of artificial intelligence (AI) hosts a number of these paradigms, including mechanisms with an ability to learn or adapt to new situations [32]. These systems are collectively known as computational intelligence (CI) which comprises a number of main paradigms: artificial neural networks (NN), evolutionary computation (EC), swarm intelligence (SI), fuzzy systems (FS), and artificial immune systems (AIS). The branch of evolutionary computation shows some characteristics similar to swarm intelligence, as both are population-based and exhibit adaptive behaviour in order to obtain better solutions to optimization problems. The development of various sub-branches of evolutionary computation preceded the advent of the discipline of swarm intelligence. Therefore, to place swarm intelligence in perspective, a brief description of evolutionary computation is presented.

The development of evolutionary algorithms was inspired by the behaviour of populations of individuals in nature. For such populations to survive, individuals of a population have to adapt to an often hostile environment. For centuries agriculturists have cultivated plants and bred domestic animals to acquire traits useful to man. During the great voyages of discovery in the 18th and 19th centuries, many new plant and animal species were discovered. Some isolated populations have developed in peculiar ways to adapt to their environments. In the absence of predators some species have even lost some of their features.

Charles Darwin, an English botanist, also joined an expedition of the "Beagle" in the capacity of a naturalist [22]. He visited the Galapagos archipelago in 1835. In the absence of humans and predators, some of the most unique life forms on earth, highly adapted to their harsh surroundings, have developed on the islands. Observations here and in other remote locations, emphasized the struggle for existence among these animals and plants. The idea that favourable variations would tend to be preserved and unfavourable ones destroyed, formed the basis of his theories about the formation of new species. Thus natural selection or "survival of the fittest" would ensure that the population having evolved over many generations, would be able to survive and reproduce.

These ideas as well as the science of genetics, inspired the development of several optimization algorithms [24] [40] [43] [55]. All of these use randomly chosen populations of individuals,

a fitness function to determine how good the solution is, and a strategy to produce offspring from the individuals.

The following techniques comprise the main sub-areas of evolutionary computation:

Genetic algorithms: Computer programs that resemble natural selection were pioneered in the late fifties and early sixties. The original genetic algorithm strategy involves a population of random strings of 1's and 0's that are generated to solve a particular problem. The measure of fitness depends on the type of problem. The process of selecting parents, recombining parents through crossover to form offspring and then selecting a new generation, is repeated until the population converges. From time to time a small fraction of strings are mutated to introduce diversity. Many modifications of the original algorithm have been tested in a variety of contexts [32], and used to solve a number of practical optimization problems [43].

Genetic programming: Genetic programming (GP) [55] is an evolutionary-based methodology to optimize a population of computer programs. The purpose of genetic programming is to find programs that perform a user-defined task. Reports on genetic programming methodology were published during the 1980's by Smith [91], Cramer [21], and Forsyth [37], while John R. Koza pioneered the solving of complex optimization problems by means of genetic programming [55].

Evolutionary programming: Evolutionary programming (EP), originally proposed by Lawrence J. Fogel in the 1960's, differs from genetic algorithms and genetic programming in that evolutionary programming evolves behavioural models instead of genetic models [35] [36]. An important difference between evolutionary programming and the other paradigms is that only mutation is used in order to introduce variation in a next population.

Evolutionary strategies: The ideas of adaptation and evolution also led to the development of evolutionary strategies (ES), developed by Rechenberg and Schwefel in the 1960's and 1970's [79] [86]. The focus of evolutionary strategies is on *evolution of evolution*. A set of strategy parameters is defined that influences the evolution process. While the population is optimized, the strategy parameters are adapted, thus optimizing the process itself.

Differential evolution: Differential evolution (DE) is a stochastic population-based optimization algorithm introduced by Storn [94] and Price [78] in 1996. DE is distinguished from other population-based techniques by the differential mutation mechanism where

the target vector is mutated by the addition of a random difference weighted vector. A trial vector is constructed and offspring is produced by crossover between a parent and the trial vector.

Cultural evolution: Cultural evolution, inspired by human social evolution, enhances the search process by including prior knowledge about the domain [81]. Culture can be defined as a system of symbolically encoded conceptual phenomena that are socially and historically transmitted within and between social groups [27]. Cultural changes occur much faster than biological evolution in natural systems, thus speeding up the search process.

2.6 Conclusion

A brief discussion of aspects of optimization was presented in this chapter. Principles from mathematical optimization that were deemed to be relevant to the objectives of this thesis were reviewed. As many modern population-based algorithms glean inspiration from adaptive natural systems, some background and interesting observations of emergent behaviour in nature were described. Finally, several evolutionary computing paradigms were briefly reviewed, as it exhibits some similarities to particle swarm optimization.