University of Birmingham

Doctoral Thesis

Analysis and Development of the Bees Algorithm for Primitive Fitting in Point Cloud Models

---

Author:
Luca Baronti

Supervisor:
Dr. Marco Castellani

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Department of Mechanical Engineering
Birmingham, United Kingdom

31 January 2020

Abstract

This work addresses the problem of fitting a geometrical primitive to a point cloud as a numerical optimisation problem. Intelligent Optimisation Techniques like Evolutionary Algorithms and the Bees Algorithm were here adapted to select the most fit primitive out of a population of solutions, and the results compared.

The necessity of understanding the dynamics of the Bees Algorithm to improve its performances and applicability led to an in-depth analysis of its key parts. A new mathematical definition of the algorithm led to the discovery and formalisation of several properties, many of which provided a mathematical answer to behaviours so far only observed in empirical tests. The implications of heuristics commonly used in the Bees Algorithm, like site abandonment and neighbourhood shrinking, were statistically analysed. The probability of a premature stalling of the local search at a site has been quantified under certain conditions. The effect of the choice of shape for the local neighbourhood on the exploitative search of the Bees Algorithm was analysed. The study revealed that this commonly overlooked aspect has profound consequences on the effectiveness of the local search, and practical applications have been suggested to address specific search problems.

The results of the primitive fitting study, and the analysis of the Bees Algorithm, inspired the creation of a new algorithm for problems where multiple solutions are sought (multi-solution optimisation). This new algorithm is an extension of the Bees Algorithm to multi-solution optimisation. It uses topological information on the search space gathered during the cycles of local search at a site, which is normally discarded, to alter the fitness function. The function is altered to discourage further search in already explored regions of the fitness landscape, and force the algorithm to discover new optima.

This new algorithm found immediate application on the multi-shape variant of the primitive fitting problem. In a series of experimental tests, the new algorithm obtained promising results, showing its ability to find many shapes in a point cloud. It also showed its suitability as a general technique for the multi-solution optimisation problem.

Declaration

I, Luca Baronti, declare that this thesis titled "Analysis and Development of the Bees Algorithm for Primitive Fitting in Point Cloud Models" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Contents

List of Figures

List of Tables

Glossary

BA Bees Algorithm. xii, xv, 1, 2, 4, 6, 7, 9, 11–13, 15, 16, 18, 21, 23, 26, 33, 41, 43, 44, 48, 49, 67, 77–80, 84, 85, 87, 88, 98, 101, 104–112

EA Evolutionary Algorithm. xi, 1, 4, 7, 9–11, 18, 19, 21–23, 25, 26, 28, 34, 36, 108, 109, 111

LORRE Local Optimum Region Radius Estimator. xiii, xv, 2, 6, 9, 17–19, 77–80, 84–88, 92, 94, 95, 98, 99, 101, 103–108, 110–112

PC Point Cloud. 1–3, 10, 24, 29, 36, 37, 77, 80, 81, 104–106, 110

PF Primitive Fitting. xi, 1–3, 6, 7, 9, 21, 26, 77, 78, 104, 108–112

RANSAC RANdom SAmple Consensus. 1, 3, 4, 7, 9–11, 21, 27, 28, 34, 36, 37

SN Sequential Niching. 19, 79, 89, 98–101, 103, 104, 108, 110

VNS Variable Neighbourhood Search. 14, 15, 73, 75, 109

Chapter 1

Introduction

The Primitive Fitting (PF) problem consists of finding the geometrical primitive (e.g. a sphere or a box) that best fits a Point Cloud (PC) (i.e. an unordered set of points in a Carthesian space). PCs are typically generated by 3D range scanners in machine vision applications. Several techniques have been developed to tackle specifically this problem. These techniques are very successful, but also work under specific assumptions on the nature of the PC (e.g. the level of noise).

The first research question addressed in this thesis is whether it is possible to solve this problem with a general technique that doesn't make any ad-hoc assumption. To answer this question, the PF problem was tackled as a numerical optimisation problem. In the proposed approach, a primitive is identified by a number of parameters describing its *position* (i.e. centre), *rotation* (when applicable) and *size* (e.g. radius, height, etc..). These parameters are encoded as an array of real numbers. The fit of a candidate solution is checked via a *fitness function* that takes into account two aspects: a) the distance of the points contained in the PC from the primitive, and b) the orientation of the normals to the points in the cloud respect to the normal to the closest surface of the primitive.

Several datasets containing PCs representing spheres, boxes and cylinders with different size, rotation and different levels of point-perturbations have been generated. The datasets have been used to test the capability of the Bees Algorithm (BA) to tackle this optimisation problem, and the results were compared with an implementation of the Evolutionary Algorithm (EA) and a RANdom SAmple Consensus (RANSAC) approach.

To better understand the dynamics of the BA, the algorithm was re-

defined under a rigorous mathematical framework, and a thorough statistical analysis of its search mechanism was performed. In the analysis, the local search stalling probability (i.e. the probability that local search stagnates until site abandonment is triggered) was studied for implementations of the BA where the neighbourhood shrinking procedure is used or not. Given limited sampling opportunities for the local search, there is a trade-off between how many candidate solutions to sample at every BA cycle (i.e. how many foragers are employed per cycle at a site), and how many cycles of stagnation to allow before the site is abandoned. This study revealed the statistical motivation behind many empirical observations in the literature on how to address best this trade-off.

The above analysis revealed also the implications on the local search of the choice of neighbourhood shape, in particular for high dimensional problems.

The analysis on the BA allowed also further observation that not only the forager that landed on the fittest solution, but also the other foragers generated during one cycle of local search can provide useful information on the topology of the search space. This latter information, normally discarded in the BA, is the basis for a new algorithm designed to find multiple local optima in multimodal functions. That is, whilst the sequence of the best solutions found at each BA cycle forms a path towards the best solution found by local search at a site (i.e. in a flower patch), the sequence of *worst* solutions provides a good indication of the radius of the local optimum attraction basin. The Local Optimum Region Radius Estimator (LORRE) algorithm use this information to alter the fitness surface, discouraging future search in that region. The chances of finding other (better) local optima are increased as a result. The presentation of LORRE includes also suggestions on how to prune the set of found local optima from spurious or duplicate candidate solutions.

The LORRE algorithm found immediate application to the PF problem, when multiple shapes are present in one PC. In this new case, LORRE was used to find every shape included in the PC. Preliminary results on a selected number of test cases show that LORRE outperforms the BA on general multimodal benchmarks as well as the PF problem.

1.1 Background

In this section the different key parts of this thesis will be illustrated in a more extensive way.

1.1.1 Primitive Fitting as a Numerical Optimisation Problem

Point Clouds are widely used in machine vision and robotics to represent 3D scenes and objects sensed through laser scanning devices. Understanding PC models, and extracting concise and meaningful high level descriptions such as the shape and properties of objects, is necessary for many industrial applications like robotic grasping and pick-and place (Bjorkman et al., 2013; Mavrakis et al., 2016). This ability is naturally acquired by humans and animals, but difficult to reliably automate (Spiers et al., 2016), particularly in real-time applications where time and hardware limitations require very efficient procedures.

One of the applications of the work undertaken in this thesis is concerned with the identification of the shape of objects in 3D PCs for robot manipulation. In many industrial applications, man-made artefacts can be associated with good approximation to a set of geometrical primitive shapes like spheres, boxes, and cylinders. The problem becomes then to fit these primitive shapes to clusters of points in PC models (i.e. PF problem). Since PCs are composed of a very large number of points, the efficiency of the identification algorithms is of primary importance. At the same time, for the sake of generality problem-specific assumptions should be limited.

The primitive fitting problem is well known in the literature, and many of the solutions are based on two popular and very successful algorithms: the Hough Transform (Levine and Levine, 1985) and RANSAC (Bolles and Fischler, 1981). Given a 3D scene, the Hough Transform looks for parameterisations of primitive shapes that fit the largest number of data points. To increase the efficiency of the HT, the space of the parameterisations may be quantised, and in that case the granularity of the quantisation becomes an important parameter (Mukhopadhyay and Chaudhuri, 2015). RANSAC randomly picks from the PC a minimal sets of points that are used to parameterise candidate primitive shapes. RANSAC verifies the candidate shapes against the remaining points in the scene, and picks the instance that fits the largest number of points. An approximation tolerance is usually set to decide

whether a point is an inlier or outlier to a given shape. The main limitations of the Hough Transform and RANSAC are their computational complexity, and the sensitivity of the results to the algorithm parameterisation (the quantisation of parameters in the Hough Transform, the approximation tolerance in RANSAC).

The approach proposed in this work is to tackle primitive fitting as a parameter optimisation problem. Similarly to the Hough Transform approach, the goal of the procedure is to manipulate the parameters of a given type of primitive to maximise its fit to the point data. The fit is measured by a primitive-specific *fitness function* which is used to guide the optimisation process. The BA (Pham and Castellani, 2009) will be used as the parameter optimisation routine, and the results compared with those achieved using an EA (Fogel, 2006) and RANSAC.

The main advantage of metaheuristics like swarm (BA) and EA over standard primitive fitting techniques is the generality of the approach, which does not require prior scene knowledge (e.g. the noise level to set the approximation tolerance in RANSAC). These metaheuristics can be used to fit any kind of shape, as long as its goodness of fit can be defined via a fitness function. In addition to their intelligent sampling of the solution space, swarm and evolutionary algorithms are also computationally efficient.

1.1.2 Bees Algorithm: Towards a Formal Definition

The BA (Pham et al., 2006) is a nature-inspired intelligent technique that has found application in a wide range of complex optimisation problems (Pham et al., 2014; Pham et al., 2018). The main idea motivating this algorithm is to model the foraging behaviour of honey bees to address the *exploration* vs *exploitation* trade-off. According to that model, agents simulating *scout bees* randomly explore the solution space looking for areas of high fitness. The scouts that found the most promising solutions recruit (through performing a *waggle dance*, Frisch 1968) other agents (*forager bees*) for local exploitative search. Local search is conducted in parallel at different *sites*, that is, within neighbourhoods centred on the solutions marked by the scouts.

Despite the initial idea being to maintain a clear separation between the global explorative and local exploitative search efforts, it soon became clear (Pham and Castellani, 2009) that other factors such as the number of parallel local searches influence the exploration vs. exploitation balance. A number of empirical studies (Pham and Castellani, 2014; Pham and Castellani, 2015)

tried to shed light on the properties of the Bees Algorithm and related optimisation techniques, and how their parameterisation affects the nature of the search effort. However, to the best of the author's knowledge, a theoretical analysis of the Bees Algorithm behaviour has never been published.

This work addresses the above gap in the literature. Understanding in detail the dynamic behaviour of a population-based optimisation algorithm on arbitrarily complex fitness landscapes is extremely complex. For this reason, the literature on nature-inspired algorithms overwhelmingly relies on qualitative biological analogies and empirical comparisons. However, by investigating well-defined cases under a theoretical framework, important insights on the algorithm behaviour can be gained (Auger and Doerr, 2011). This study focuses on the local search procedure of the Bees Algorithm, using the clearly delimited boundaries of the site neighbourhood to infer important properties.

The proposed study is timely, as a large number of variants in operators and parameterisations have been developed for this popular optimisation algorithm (Hussein et al., 2017). In the light of the *No Free Lunch Theorem* (Wolpert and Macready, 1997), it is essential to unravel the implications of these different choices of operators and parameters.

1.1.3 LORRE: A new Approach to the Multi-Modal Problems

In contrast with *single-solution* optimisation techniques, that are designed to find a good approximation of the global optimum, multi-solution techniques aim to find all the best (i.e. with highest fitness score) local optima. In many cases, especially in presence of a single strong attractive basin in the search space, many optimisation techniques tend to rapidly converge to the same local optimum. A rapid convergence and consistent convergence is certainly a desirable property for a single solution optimisation technique, although it can pose problems if the optimum is not the global best. If many (all the) local optima are sought, the algorithm must be able to always search new areas of the solution space.

A solution to the problem is to build a mechanism in the search procedure that avoids further search in the area around already found optima, similar to the *tabu* regions in Tabu Search (Glover and Laguna, 1998). This general approach entails the solution of a number of sub-problems. That is, the

following elements need to be defined:

(i) a method to find a good approximation of one (or more) local optimum(a);

(ii) a way to estimate the local optima attraction basins;

(iii) a method to exclude those sub-regions in future searches;

The first sub-problem of locating the fitness peaks is solved using the search mechanism of the standard BA. The second and third sub-problems are solved using an innovative method created by the author. The combination of standard BA search and routines to diversify the search constitutes the LORRE algorithm, which can be regarded as a BA variant.

LORRE is specifically designed to address multi-solution problems, and dynamically modifies the fitness landscape once local BA search at a site reaches a peak and is exhausted. The fitness landscape is modified via a *derating function*, which removes the attraction basin around the optimum (addressing thus the third sub-problem). The action of the derating function can be visualised as (partly) 'filling up' a hole in the fitness landscape (minimisation problems), or (partly or completely) flattening a peak (maximisation problems). By actually removing one attractor basin, the derating function prevents further local search in the area. The two key aspects of this technique are the *smoothness* of the derating functions, which can be tuned by a parameter, and the *dynamic* estimation of the *fields radius*. The latter aspect is achieved analysing the distribution of the *worst solutions* found during the local search at a site. This addresses the second sub-problem.

1.2 Aims of the Thesis

The aims of the thesis can be described as follows:

- to devise a novel approach to tackle PF as a numerical optimisation problem;
- to investigate the use of metaheuristics, and in particular the BA, for the solution of the PF problem, initially focusing on single-shape fitting (single-solution optimisation problems).

- to perform an experimental comparison of the performance of the metaheuristics approach with the state-of-the-art on the PF problem;
- to focus on the best performing metaheuristics, and carry out an analytical study with the aim of better understanding it mechanisms and improve its performances;
- to use the results and knowledge so far acquired to extend the metaheuristics approach to the solution of multi-shape fitting (multi-solution optimisation) problems;
- to experimentally compare the performance of the novel metaheuristics approach with the state-of-the-art in multi-solution optimisation on standard benchmarks and the PF problem;

1.3 Outline of this Thesis

The remaining chapters of the thesis are organised as follows:

- Chapter 2 contains the literature review of the known PF techniques, relevant variants of the BA and multi-solution techniques for multimodal problems;
- Chapter 3 illustrates the proposed BA solution for the PF problem, along with the comparative results with the EA and RANSAC techniques;
- Chapter 4 presents the statistical analysis of the BA, with a focus on the stalling probability of a site. In this chapter, formal proofs of key propositions are also coupled with empirical tests;
- Chapter 5 describe the new technique designed to find the best local optima of a multimodal function. The algorithm will be validated on benchmark functions as well as a multi-shape variant of the PF problem;

Chapter 2

Literature Review

This chapter reviews the literature related to the topics of this thesis. The first section is centred on the PF problem, with particular focus on RANSAC variants. It also includes the literature on approaches tackling PF as a numerical optimisation problem, and using global search techniques such as EA as solvers. The second section analyses the main variants of the BA, particularly those relevant to the analysis of Chapter 4. Two optimisation techniques that share close similarities with the BA local search are reviewed in detail. Finally, the last section of this chapter reviews the literature on approaches to the multi-solution optimisation problem, discussing the main techniques and how the LORRE algorithm compares to the state-of-the-art.

2.1 Primitive Fitting

The Hough Transform aims to fit primitive shapes to sets of points in the scene. Primitive fitting is regarded as a search problem in the space of the shape parameters, and sequential search algorithms are typically used to find the instances that include the largest number of points. The Hough Transform is widely used in machine vision (Ballard, 1981), but is computationally demanding and becomes rapidly inefficient as the number of parameters needed to define the shape increases. As a consequence, the Hough Transform has been mainly used to fit elementary shapes such as lines and circles (Dalitz et al., 2017). Only a few implementations of the Hough Transform were proposed for 3D primitive shape recognition, either based on parameter search heuristics (Khoshelham, 2007), or customising the search to detect one

particular instance of shape (Rabbani and Van Den Heuvel, 2005). Alternative methods to the Hough Transform (Roth and Levine, 1993) have been developed to fit geometric primitives to point data, often based on robust statistical estimation of the shape parameters.

The RANSAC algorithm randomly picks from the scene minimal sets of points that uniquely define a given type of geometric primitive. Candidate shapes are tested against all points, and the shape that approximates the largest number of points is extracted. The procedure is then sequentially repeated on the remaining data. The algorithm steps will be discussed in greater details in Section 3.1.5. Different RANSAC approaches have shown promising results for 3D scenes in terms of accuracy and efficiency (Sveier, 2016; Schnabel et al., 2007), and have been shown able to fit candidate primitives in environments of 90% noise with little error (Schnabel et al., 2007). Advanced subroutines can be applied to preemptively terminate bad hypotheses (Raguram et al., 2008). Much effort has been dedicated to optimise RANSAC sampling and shape evaluation efficiency: in the former case, Optimal RANSAC has shown "substantial speedup for highly contaminated sets" with as much as 96% noise (Hast et al., 2013), whilst in the latter case Optimal Randomized RANSAC has been shown to run 2–10 times faster than standard RANSAC (Chum and Matas, 2008). By combing these optimisation strategies, some forms of RANSAC were able to fit primitive shapes to a field of millions of points in less than one minute (Schnabel et al., 2007).

The success of RANSAC greatly depends on the trade-off between accuracy and computational complexity, namely, by the number of candidate shapes evaluated. The results obtained using RANSAC are also sensitive to the setting for the tolerance threshold used to judge whether a data point is an inlier or an outlier to a given candidate shape. Some RANSAC implementations do not utilise tolerance threshold and score candidate shapes based on histogram analysis (Liu and Wu, 2014). However, these implementations were usually tested only on PCs of varying levels of background noise (henceforth called *noise*, Figure 2.1) instead of the more deceptive case of local error (henceforth called *error*, Figure 2.1) which may arise from granularity or low precision of sensors.

Some examples of EA methods for primitive fitting have been proposed. Lutton and Martinez (1994) and Roth and Levine (1994) used a population of many possible shapes. Each individual (*candidate solution*) in the population represents the minimal set of points that uniquely define the primitive, whilst the fitness function counts the number of points inside a fixed

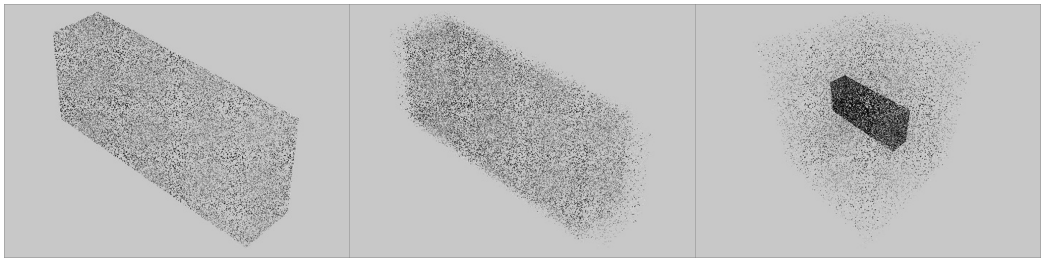


Figure 2.1: (Left) clean point cloud, no noise or error. (Middle) point cloud with error, no noise. (Right) point cloud with noise, no error. Image taken from (Alston, 2019).

boundary around the primitive. Similarly, Gotardo et al. (2003) used an EA for tackling a sub-task of the surface extraction problem. In this case the population of candidate solutions is composed uniquely of planes, and each individual represents the minimum set of points (three) needed to define a plane, sampled in a sub-region of the cloud. The optimisation strategy of the above EA approaches is clearly based on the RANSAC procedure, and still needs a careful setting of the approximation tolerance. Ugolotti et al. (2014) tested one instance of swarm algorithm (Eberhart and Kennedy, 1995) and one instance of evolutionary algorithm (Storn and Price, 1997) for object fitting. Each candidate solution encoded the six parameters defining a rigid transformation (rotation+translation) of a template point cloud shape. The fitness function evaluated the difference between the rotated and translated template and the target shape. The main drawback of this approach is the computational complexity, which required the implementation on Graphics Processing Units (GPU).

2.2 The Bees Algorithm

This thesis is centred on the BA which will be briefly described in this section to provide the reader an introduction to the algorithm.

The BA is a popular intelligent optimisation technique that found wide application in optimisation problems (Pham et al., 2018). Inspired by the honeybees foraging behaviour, it performs multiple simultaneous local searches at different sites of the solution space. The BA considers candidate solutions as food sources, and employs artificial bees to evaluate their quality (fitness)

and exploit the most promising regions of the search space. The algorithm begins sending ns artificial scout bees to randomly sampled locations (candidate solutions) in the search space. The scout bees evaluate the quality of the food sources where they landed using Equation (3.1). Each visited solution becomes the centre of a neighbourhood delimited by a hypercube of side $ngh = 2\delta^I$. The algorithm then enters the main loop, which consists of a number of steps. The first step is called *waggle dance*, in analogy with the waggle dance behaviour of honey bees where foragers are recruited for harvesting the richest food sources. In this step, the neighbourhoods around the fittest nb solutions visited by the scouts are selected for local search.

The second step is where the simultaneous exploitative searches are performed, that is the neighbourhoods are harvested (local search). Namely, nre forager bees are sent to exploit the neighbourhood of the very best $ne \leq nb$ visited solutions, and $nrb \leq nre$ foragers are sent to the remaining $nb - ne$ sites. Each forager lands on a food source in the assigned neighbourhood, and evaluates its quality. The landing site of the foragers is determined using the procedure described in Section 3.1.3. That is, in the local search step the BA concurrently samples the neighbourhoods around the most promising solutions. The forager that visited the fittest solution within a neighbourhood becomes the new scout, and the centre of the neighbourhood is moved to that solution.

If no forager finds a solution that is fitter than the centre of the neighbourhood, the scout remains unchanged, and the local search is said to stagnate. In this case, the size of the neighbourhood is reduced (*neighbourhood shrinking* procedure). After *stlim* consecutive stagnation cycles the neighbourhood is abandoned and the scout is re-initialised at a new randomly picked location in the search space (*site abandonment* procedure).

Global explorative search (third step) is performed by the remaining $ns - nb$ scouts, which keep on randomly sampling the solution space looking for new promising regions. At the end of one cycle of the main loop, nb scouts mark the neighbourhoods resulting from local search, and $ns - nb$ scouts mark the neighbourhoods found through global search. The algorithm terminates after a given number of iterations returning the best solution found.

For more details on the BA and its capabilities, the reader is referred to (Pham and Castellani, 2009; Pham and Castellani, 2014; Pham and Castellani, 2015).

2.2.1 Main Variants of the Bees Algorithm

Besides the standard procedure described in Section 4.1, many different variants of the Bees Algorithm exist. A recent survey (Hussein et al., 2017) separated three main branches of the Bees Algorithm, namely the Basic BA, the Shrinking-based BA, and the Standard BA.

The Basic BA refers to the basic form of the Bees Algorithm, which is mentioned in several articles (Pham et al., 2008; Packianather et al., 2009; Pham and Castellani, 2009; Yuce et al., 2013) and performs parallel local searches using *fixed* neighbourhoods (i.e. no neighbourhood shrinking).

The Shrinking-based BA (Pham et al., 2006) includes the neighbourhood shrinking procedure. The heuristics behind the neighbourhood shrinking procedure is to intensify the exploitation effort as the local search progresses, focusing the sampling on increasingly smaller regions of the solution space.

Finally the Standard BA, so termed in numerous articles (Pham and Castellani, 2009; Castellani et al., 2012; Pham et al., 2012), includes the neighbourhood shrinking and site abandonment procedures. The heuristics behind site abandonment is to abandon a site once the local search stagnates, to avoid being trapped into local fitness peaks.

Many recruitment, neighbourhood alteration, and site abandonment heuristics were proposed in the literature.

Ghanbarzadeh (2007) proposed two methods for setting the number of recruited foragers proportionally to a) the fitness or b) the location of the sites. Other authors proposed recruitment schemes where the number of foragers was proportional to the fitness of the site, and decreased it progressively by a fixed amount (Packianather et al., 2009), or according to a *fuzzy logic* policy (Pham and Darwish, 2008). Pham and Darwish (2010) used Kalman filtering to allocate number of bees to the sites selected for local search. This strategy was used to train a Radial Basis Function neural network, and improved the learning accuracy and speed of the neural network. Finally, Imanguliyev (2013) proposed a recruitment scheme where the number of foragers for a site was computed on the *efficiency rate* of the site, rather than its fitness score.

In its basic instance (Ghanbarzadeh, 2007), the search scope of a site is changed (reduced) when local search fails to improve. Ahmad (2012) proposed two different methods to change dynamically the neighbourhood of a site: a) BA-NE where the search scope is *increased* if a better solution is found and kept invariant otherwise, and b) BA-AN, where the neighbour-

hood is *asymmetrically* increased along the direction that led to the last improvement and decreased otherwise.

When a site is abandoned, the best-so-far local solution is usually kept in memory (Pham and Castellani, 2009). However, in some cases (Packianather et al., 2009; Pham and Koç, 2010) all the local solutions found before abandoning a site are retained for later use. In Hierarchical Site Abandonment (Pham and Darwish, 2008) when a site s is abandoned, all the other sites with fitness lesser or equal to s are abandoned too.

2.3 Related Techniques

Variable Neighbourhood Search (VNS) (Mladenović and Hansen, 1997) and LJ Search (Luus and Jaakola, 1973) have important similarities with the local search strategy of the Bees Algorithm, and will be discussed in detail in this section.

2.3.1 Variable Neighbourhood Search

Variable Neighbourhood Search iterates cycles of local search around a seed solution using neighbourhoods of different size. This idea has been successfully used (Hansen and Mladenović, 2003) in numerous applications, like the Traveling Salesman (Mladenović and Hansen, 1997) and the Vehicle Routing (Rousseau et al., 1999; Bräysy, 2001) problems. The main steps of the VNS algorithm are:

Variable Neighbourhood Search: Main Steps

1. Initialise k neighbourhoods $S_1, \ldots, S_k$ of variable size around a randomly generated centre x, initialise $i = 1$;
2. Take neighbourhood S_i:
   (a) sample a solution v uniformly inside the neighbourhood S_i;
   (b) apply a local search procedure using v as seed to find a new solution v';
      i. if v' is fitter that x, set $x = v'$ and $i = 1$;
      ii. else set $i = i + 1$;
3. If $i > k$, terminate the algorithm and return the best found solution, otherwise iterate from step 2;

VNS is akin to local neighbourhood search at a BA site. A variant of this approach, called Reduced Variable Neighbourhood Search (Whitaker, 1983; Mladenović et al., 2003), skips the local search at step 2b, to keep $v' = v$. Reduced Variable Neighbourhood Search is equivalent to a Basic BA where $nb = 1$, $nr = 1$, and $ns = 0$ (no global search). Parallel Variable Neighbourhood Search (Garcia-López et al., 2002; Djenić et al., 2016) resembles more closely the BA, since it performs a number of concomitant local VNS search procedures.

The main difference between VNS and the BA is that the former uses a randomly generated sequence of neighbourhood sizes for local search, whilst the latter uses a fixed (Basic BA) or deterministically shrunk (Shrinking-based BA) neighbourhood size. In addition, VNS uses a fixed number of foragers per optimisation cycle, whilst the number of foragers is determined by the quality of the neighbourhood in the BA.

2.3.2 *LJ* Search

The *LJ* Search Method was successfully used to optimise feedback control in nonlinear systems (Luus, 1974b), as well as time-optimal (Luus, 1974a)

and time-delay (Oh and Luus, 1976; Nair, 1979) systems. Given an N-dimensional minimisation problem, the LJ Search Method pseudocode is:

LJ Search: Main Steps

1. Let s_1^e be the N-dimensional vector of the spans ($max - min$ value) of the interval of definition of the N decision variables, s_1 the seed solution, and $t = 1$ an index;
2. Sample nr solutions $v_1, \ldots, v_{nr}$ as follows:
$$v_i = s_t + u \cdot s_t^e \qquad i \in \{1, \ldots, nr\}$$
where u is a vector of N real values independently sampled with uniform distribution in $[-0.5, 0.5]$;
3. Compute the next solution as:
$$s_{t+1} = \arg\min_v \{\mathscr{F}(v) \mid v \in \{s_t, v_1, \ldots, v_{nr}\}\}$$
4. Reduce the ranges of a given factor α:
$$s_{t+1}^e = \alpha s_t^e$$
and increment the counter $t = t + 1$;
5. check if the stop criterion is met. If so return s_t, otherwise go back to step 2;

Except for the initialisation of the neighbourhood, the LJ Search algorithm closely resembles the BA local search procedure at one site with neighbourhood shrinking. That is, the Shrinking-based BA can be described as a multi-LJ Search method. Surprisingly, to the best of the authors' knowledge, the similarity between LJ Search and the Bees Algorithm has so far been overlooked in the literature.

A mathematical analysis of the properties of the LJ Search method was published by Gopalakrishnan Nair (1979). In particular, Gopalakrishnan Nair proved that the succession of solutions $s_1, \ldots, s_n$ converges to a local

optimum. Unfortunately, the proof of convergence is only valid for an infinite number of iterations. Moreover, in his analysis Gopalakrishnan Nair took in consideration hyperspherical neighbourhoods instead of the hypercubic ones that are actually used in the LJ Search algorithm. As will be shown in Section 4.4, the use of different neighbourhoods has important consequences on the properties of the search.

2.4 LORRE: a Comparison with Similar Techniques

The idea of avoiding already explored regions to increase the chances of finding better solutions in different areas of the search space is not an innovative aspect of the LORRE algorithm. The literature on the topic includes different techniques that will be reviewed in this section. Many of these techniques were introduced in the context of single-solution problems, where only the global optimum is sought, in order to increase the exploration capability of the main search algorithm. In other cases, some of the techniques here reviewed were developed to solve multi-solution problems.

A widely known meta-heuristic designed to avoid already explored regions, in single-solution problems, is the Tabu Search algorithm (Glover and Laguna, 1998). Tabu Search memorises the visited solutions in a *tabu list*. Solutions in the tabu list are not considered in future local searches, unless all other reachable solutions are included in that list.

The Tabu Search algorithm has been used to boost the exploration capability of other meta-heuristics and algorithms, such as Ant Colony Optimisation (Dorigo and Blum, 2005) or the Scatter Search algorithm (Glover, 1977). Both algorithms use Tabu Search to memorise and avoid already sampled solutions in the context of single-solution search. In other cases, the dynamically updated list of visited solutions (sometimes only the local optima) is used to dynamically shift the balance between exploration and exploitation. These techniques are grouped under the umbrella term of Adaptive Memory Programming by some authors (Taillard et al., 2001). The general Adaptive Memory Programming approach was summarised as follows: (i) to sample a new solution s taking into account the history of solutions sampled, (ii) to find a better solution s' than s performing some kind of local search, and (iii) to update the list of sampled solutions adding s'.

The common problem with Adaptive Memory Programming approaches is that they often work under the assumption of dealing with a discrete solutions space. In a continuous solutions space, techniques like Tabu Search are difficult to apply.

The BA can be regarded as an Adaptive Memory Programming technique since it stores in memory the best site centres, and performs local searches around these centres. The main difference is that the Bees Algorithm keeps in memory only the currently sampled local peaks, without any memory of *past* searches. The LORRE algorithm is closer to Adaptive Memory Programming since it also takes into account information on past local search results. Notably, this extra information is not taken into account until the search has been exhausted at a site, and is shared globally by all new local searches (i.e all scouts generated after the site has stalled).

In the context of BA applications to single-solution optimisation problems, (Shatnawi et al., 2013) used local and global memories of the visited regions of the search space. The local memories were used by the single bees to avoid sampling the same location twice, whilst the global memories were used to drive local search away from already explored regions.

Another BA variant (Imanguliyev, 2013) uses *shift* operators to identify planar sub-regions in the space, and includes these regions in a tabu list, similarly to the Tabu Search approach. In this case, a planar sub-region is identified when the solutions sampled inside differ in fitness score by less than a fixed, user-defined, threshold. Unfortunately, the paper analyses only one-dimensional problems, and the authors did not mention how to expand the approach to higher dimensional cases. Imanguliyev (2013) also did not mention any solution to prevent the tabu list from becoming unmanageably large as the search progresses.

In the context of EA-based multi-solution optimisation (Shir, 2012) two well-known diversification techniques are *fitness sharing* and *crowding*. Fitness sharing (John, 1992) reduces the fitness score of an individual proportionately to its distance to other members of the population. This mechanism aims to promote *niching* dynamics, where individuals fall into the attraction basin of different local optima. This technique is usually complemented with some (Sareni and Krahenbuhl, 1998) heuristics, and uses a *niching radius* that defines the maximum distance of interaction between the fitness of different individuals.

The crowding method (De Jong, 1975) is a variant of the EA where, for each child generated, a sub-population of cf individuals, where cf is the

crowding factor defined by the user, are selected from the parent population. The child replaces the most similar individual, according a certain metric, of this sub-population if its fitness score is greater. Following studies (Thomsen, 2004; Wong et al., 2012) incorporated this principle to tackle multi-solution problems, using the population size as crowding factor and the Euclidean distance as metric, with competitive performances compared with the state-of-the-art (Wong et al., 2012).

A meta-heuristic for the multi-solution problem is Sequential Niching (SN) (Beasley et al., 1993). This approach adopts a simple, yet effective, strategy to find the multiple local optima of a function. SN makes use of a single-solution optimiser, which is repeatedly run on a progressively modified objective function. At first, the optimiser is run on the original objective function until a stopping criterion is reached, and the fittest found solution o (i.e. the best approximation of the global optimum) is saved. Then the objective function is modified using a *derating function*, that locally decreases the score of the objective function, centred in o and with a pre-fixed radius. A new instance of the optimiser is ran using the modified objective function, and another local optimum is found. The process is iterated, progressively modifying the objective function with further derating functions. The process stops when a sufficient number of local optima are found.

The LORRE algorithm borrows the idea of modifying the objective function using several derating functions from the SN technique. Both SN and the EA with fitness sharing use the feature of locally reducing the objective function score, as well as the problem of setting the optimal radius for such reduction. In contrast, the LORRE algorithm estimates the borders of the local basins of attraction, to adaptively set the radius of the derating functions.

Chapter 3

Bees Algorithm Applied to Primitive Fitting Problem

In this chapter the approach that considered the PF as an optimisation problem will be illustrated in detail. In the Section 3.1, the representation scheme, fitness function, and local search heuristics used in the BA and EA will be presented first. The EA and the RANSAC implementation used in this study will be then described. In Section 3.2 the dataset used in the tests is described and the validation function used to assess the solutions is defined.

3.1 Primitive Fitting Methods

In this work the PF is tackled as an optimisation problem, and solved using the BA a biologically inspired technique. The results obtained using the BA will be compared to those obtained using a standard RANSAC procedure, and another popular metaheuristics, namely an EA. This section describes the three algorithms in detail. These algorithms will be tasked to recognise instances of three types of shape primitives in point clouds: *spheres*, *boxes*, and *cylinders*. The BA and EA encode a primitive using the same representation scheme, and use the same fitness evaluation function to assess how a candidate solution fits the data points. Therefore, it can be said that they operate in the same *fitness landscape*. The BA and EA also share the same local search operator. They differ in the kind of metaheuristics they employ, which determines the way the results of the local search (the heuristics) are used.

3.1.1 Representation Scheme

A primitive in a 3D scene is unequivocally described by a finite set of parameters that determine its position (the geometric centre, or centroid, henceforth simply referred to as *centre*), rotation and other geometrical properties. A solution I is thus naturally encoded as a vector of real values representing the primitive parameters. The size of the vector is shape-specific:

Sphere 4 parameters: 3 to locate the centre, and one to represent the radius;

Box 10 parameters: 3 to locate the centre, 4 to describe the orientation [1] and 3 parameters to encode the width, depth and height;

Cylinder 9 parameters: 3 to locate the centre, 4 to describe the orientation and 2 parameters to encode the radius and height;

For the sake of clarity, henceforth a distinction will be made between *pose parameters* which include the position and, when applicable, the orientation, and the *size parameters*, namely: the radius of the sphere; the width, depth and height of the box; and the radius and height of the cylinder.

3.1.2 Fitness Function

The term 'fitness function' is adopted from the EA terminology, and is widely used in the wider metaheuristics literature. In the proposed application, the fitness function quantifies the goodness of fit of a primitive shape I to a given point cloud $\mathcal{PC} = \{p_1, \ldots, p_N\}$ of N elements. The evaluation criterion for the goodness of fit takes into account two factors: the distance $\delta(p_i, I)$ between each of the individual points $p_i \in \mathcal{PC}$ and the surface of the primitive I; and the concordance $NC(p_i, I)$ of the normals calculated at each point $p_i \in \mathcal{PC}$ and its projection $\pi(p_i, I)$ on the closest surface of the primitive:

$$\mathscr{F}(I, \mathcal{PC}) = \frac{1}{N} \sum_{i=1}^{N} \frac{NC(p_i, I)}{1 + \frac{\delta(p_i, I)}{\delta_{max}}^2} \tag{3.1}$$

where the normalisation factor δ_{max} is the distance between the centroid and the outmost element of the point cloud. Given a point $p_i \in \mathcal{PC}$, the projection $\pi(p_i, I)$ is the closest part of the candidate primitive surface to the point

[1] Rotations are described using quaternions.

p_i. The calculation of $NC(p_i, I)$ necessitates of a method to calculate the normals to the elements of a PC. If the normals are not known, the reader is referred to the literature (Mitra and Nguyen, 2003; Boulch and Marlet, 2012) for a suitable extraction method. The function in Equation (3.1) is the same for each type of primitive, whilst the concordance of normals $NC(p_i, I)$ and distance $\delta(p_i, I)$ are type-specific.

For a given sphere S, the distance $\delta(p_i, S)$ between an arbitrary point p_i and $I = S$ is computed as:

$$\delta(p_i, S) = |d(p_i, S_c) - S_r| \tag{3.2}$$

where the function $d(A, B)$ measures the Euclidean distance between points A and B, and S_c and S_r are respectively the centre and radius of S. For the box and cylinder, $\delta(p_i, I)$ is computed as the distance between point $p_i \in \mathcal{PC}$ and its projection $\pi(p_i, I)$:

$$\delta(p_i, I) = |d(p_i, \pi(p_i, I))| \tag{3.3}$$

The concordance of normals $NC(p_i, I)$ is computed using the *cosine similarity* between the normal $N(p_i)$ to point $p_i \in \mathcal{PC}$ and the normal $N(\pi(p_i, I))$ of its projection on the candidate primitive surface:

$$NC(p_i, I) = \max\left(\frac{N(p_i) \cdot N(\pi(p_i, I))}{\|N(p_i)\| \, \|N(\pi(p_i, I))\|}, 0\right) \tag{3.4}$$

where $\cdot$ denotes the *dot product* between two vectors. Using Equation (3.4), only normals that agree in direction contribute to the calculation of the goodness of fit.

From Equation (3.1), it is easy to see that $\frac{\delta(p_i,I)}{\delta_{max}}^2$ is minimum and equal to zero when the shape fits perfectly the data. In this case, the denominator of Equation (3.1) is minimum and equal to 1. The numerator is maximum and equal to 1 when the shape fits perfectly the data, and all the normals concord. Hence, the parameter fitting problem consists of maximising the output of $\mathscr{F}(I, \mathcal{PC})$.

3.1.3 Local Search Operator

Mutation in the EA and the local search in the BA are performed using the same shape modification operator. Each of the parameters g_i^I describing a

candidate primitive shape I may be modified in the following way:

$$g_i^I = g_i^I + 0.1(u_i - l_i)\rho \tag{3.5}$$

where

$$\rho \sim U(-\delta^I, \delta^I) \tag{3.6}$$

is a random sample drawn with uniform probability in the $[-\delta^I, \delta^I]$ range, u_i and l_i are respectively the upper and lower bound of the i-th parameter, and $i = \{1, ..., n\}$. The number n of parameters is equal to four if I is a sphere, ten if it is a box, and nine if it is a cylinder (Section 3.1.1). Should one parameter g_i^I be modified to a value outside the $[u_i, l_i]$ interval, it will be placed on the closest extreme.

The shape modification procedure works as follows. The first step is to establish which features (centre, orientation, or size) of the shape are to be modified. If the sought primitive is a sphere, all the features (centre and size) are modified with probability p^f. If it is a cylinder or a box, only one feature is changed: either the centre, or the orientation, or the size. The probabilities of changing each of the features is given in the left-hand side of Table 3.1.

Once it has been established which features to change, the second step of the procedure is to determine which parameters (the g_i^I) are to be changed. Each parameter g_i^I is modified with a probability p^p. For each parameter, the mutation probability is given in the right-hand side of Table 3.1. For example, if the centre of a box is to be changed, each of its X, Y, and Z coordinates will be changed with probability $p^p = 0.7$. If a change of orientation is drawn, the four values describing the orientation are all modified with probability p^p. Since orientation is expressed in the form of a unit quaternion, the modified quaternion vector is then normalised.

Preliminary tests have shown that in many cases the optimisation process tended first to fit some surfaces of the candidate primitive to the PC (e.g. the four lateral faces of a box), and then to adjust the remaining ones (e.g. the top and bottom faces). This often led the algorithm to become stuck in shape configurations (i.e. local fitness optima) that could not be further optimised with one single modification event. For example, a box of width λ that fits a box-shaped PC of width $\lambda+\delta$ perfectly on five faces, is short on the sixth face. One single change of width would very unlikely fit the sixth face to the data, and at the same time would certainly destroy the alignment of the opposite face to the data (if the width modification is applied symmetrically respect

Feature	p^f			g_i^I	p^p		
	Sphere	Box	Cylinder		Sphere	Box	Cylinder
Centre	1	0.3*	0.33*	X	1	0.7	0.7
				Y	1	0.7	0.7
				Z	1	0.7	0.7
Rotation	-	0.3*	0.33*	$w_1 i$	-	0.7	0.7
				$w_2 j$	-	0.7	0.7
				$w_3 k$	-	0.7	0.7
				w_4	-	0.7	0.7
Size	1	0.4*	0.33*	height	-	0.33*	0.7
				width	-	0.33*	-
				depth	-	0.33*	-
				radius	1	-	0.7

Table 3.1: Shape modification probabilities. Probabilities marked with an asterisk are mutually exclusive (e.g. either the centre, or the orientation, or the size of a cylinder is changed)

to the centre). To avoid this problem the modification procedure keeps one face of the shape fixed. For example, the height of a box or cylinder may be changed keeping the bottom or top face fixed, and modifying the height.

The differences in the way the shape modification procedure is applied to the different shapes reflect the different levels of disruptiveness the procedure has on said shapes.

3.1.4 Evolutionary Algorithm

EAs are global search techniques modelled on the process of natural selection of species as described by Darwin, and on the laws of inheritance of traits postulated by Mendel and Wilson (Wilson, 1900; Ayala and Kiger, 1984). In analogy with biology, in EA terminology the vectors encoding the solutions (Section 3.1.1) are called *chromosomes*, and their elements (the parameters) are called *genes*.

The optimisation process, illustrated in Figure 3.1, is started randomly initialising a population of p candidate solutions. The algorithm then enters the main loop, which consists of a number of steps. In the first step, the fitness of the population is evaluated using Equation (3.1). In the second step (*selection scheme*), the population is ranked in decreasing order of fitness,

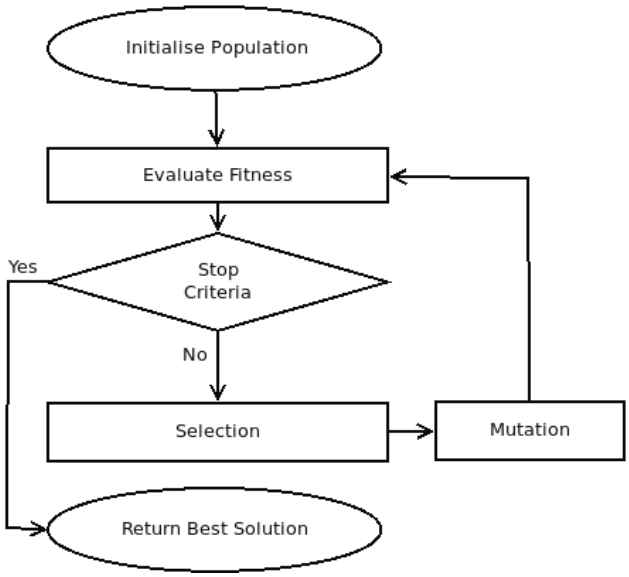


Figure 3.1: Flowchart of the EA used for the PF problem. Notably, in this implementation a crossover operator is not used.

and sampled with replacement to select $p-1$ seeds (parents). Population sampling is done allocating selection probabilities proportionally to the position in the ranking (Fogel, 2006). In the third step (*mutation*), the $p-1$ selected parents are used to generate $p-1$ offspring using Equation (3.5). The mutation procedure employs the parameter modification operator described in Section 3.1.3. The main difference between the EA and the BA in the use of the parameter modification operator is that the former encodes the neighbourhood size (δ^I) in one extra chromosome, and lets it undergo evolution. The EA thus adapts the scope of the local search using the same evolutionary process used to evolve the solutions. The BA progressively shrinks δ^I via the neighbourhood shrinking procedure.

A new population is formed from the $p-1$ offspring and the fittest individual of the current population, according to the *generational replacement with elitism* procedure (Fogel, 2006).

The EA used in this study iterates *gen* cycles (*generations*) of evaluation, selection, and mutation. Genetic crossover (Fogel, 2006) is not implemented; for this reason, and the fact that the mutation width δ^I undergoes evolution, the EA is close to the Evolutionary Programming approach (Fogel, 2006).

3.1.5 RANSAC

As seen in Section 2.1, a number of RANSAC implementations were proposed in the literature. The main variations over the standard procedure concerned the use of heuristics for faster execution, rather than changes in the search procedure. Often, the precise algorithmic details of these RANSAC variants were not fully reported, and for this reason the standard procedure was implemented in this study (Bolles and Fischler, 1981). RANSAC is an iterative process, where one new candidate shape is created and scored every cycle. Each iteration comprises of three primary subroutines.

The first RANSAC subroutine creates a minimal subset P_{ms} of points, where $P_{ms} \subset \mathcal{PC} = \{p_1, \ldots, p_N\}$ and $ms < N$. The subset P_{ms} contains the minimum number of points needed to fully define a candidate shape. Namely, four points are needed to define a sphere, five to define a cylinder and six to define a box (Figure 3.2).

For the sphere, all four points are randomly sampled at once, and resampled if they are coplanar. For the cylinder and box, one point is picked from the PC at a time, and a set of tests are performed to determine if the newly sampled point is on the same or a different shape face to the points already populating P_{ms} (i.e. all normals must be perpendicular or of opposite direction to each other). The newly sampled point is added if it lies on a different face respect to all points in P_{ms}, otherwise is discarded. In the case of the box, the goal of the sampling procedure is to have a point on every face. In the case of the cylinder, the goal is to have three points on the cylindrical outer surface, and one point on each of the end faces.

The second RANSAC subroutine defines a candidate primitive shape from P_{ms}. The parameters required to fully define a shape are similar to those defined in Section 3.1.1, with the sole difference of the use of Euler's angles (i.e. roll, pitch, yaw) instead of quaternions to define orientation. For spherical primitives, the parameters were found using Schmitt's technique (Alston, 2019). For cylinders and boxes, the orientation of the shape is found using the normals to two of the points in P_{ms} (once the orientation of two perpendicular faces is found, the third dimension can be retrieved from the right-hand rule). After the orientation of the candidate shape is found, the size is determined from the whole set of points in P_{ms} (pairs of points in P_{ms} on opposite faces delimit the boundaries of the candidate shape). Finally, the centre is calculated from the reconstructed shape.

The third and final step is to score the candidate primitive shape. The

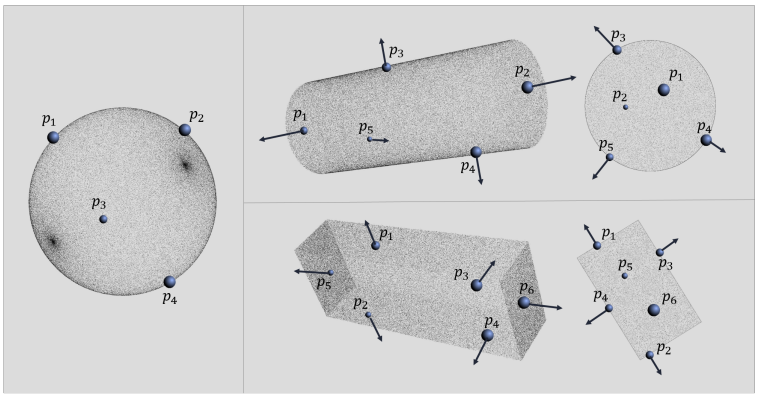


Figure 3.2: Four non coplanar points are used to define a sphere (see left). Five points are used to define a cylinder, with one on each end face and three on the outer cylindrical surface (see top right). Six points are used to define a box, one point for each side (see bottom right). Note that the cylinder and box require estimated surface normals to the PC to validate the minimal set of points. Image taken from (Alston, 2019).

score is calculated as follows:

$$Score(I) = \sum_{i=1}^{N} \min\{dist(p_i, I), \epsilon\} \tag{3.7}$$

where N is the number of points in the PC, $\delta(p_i, I)$ is the shortest distance between p_i and the surface of the candidate shape I, and ϵ is the approximation tolerance. In the RANSAC implementation used in this chapter $\epsilon = 0.3$. A perfectly fitting shape scores zero.

3.2 Experimental Method

The performance of the Bees Algorithm, the EA, and RANSAC was evaluated on three data sets, using a purpose-built error function. This function is different from the goodness of fit function used in the individual algorithms, and this guarantees an unbiased evaluation of the results. For each data set,

forty independent runs of each algorithm were performed, and the results statistically analysed.

3.2.1 Data Sets Used

Each data set comprised of 591 3D models of 10^3 data points each. Each model represented one primitive shape (sphere, box, or cylinder) of different proportions and orientation. Namely, each data set was composed of:

- 181 individual models of spheres. where the radius was varied from 1 to 10 units in steps of 0.05;
- 220 individual models of boxes, where the width, height and depth were varied from 1 to 10 units in steps of 1, and took all possible combinations of these levels (full factorial design). The orientation was randomly determined;
- 190 individual models of cylinders, where the radius was varied from 0.5 to 5 units in increments of 0.25, the height from 1 to 10 units in steps of 1, and radius and height took all possible combinations of these levels (full factorial design). The orientation was randomly determined;

The centre of the shapes was set at the origin. Each PC was created first forming and rotating the primitive shape, and then uniformly sampling 10^3 data points from its surface. The three data sets differed for the amount of noise (error, see Figure 2.1) in the sampling of the points. Namely:

- *Clean* set: there is no error, the data points lie exactly on the surface of the primitive shape;
- *Error* set: the position of each point was randomly perturbed with uniform probability within a 0.1 unit radius;
- *Double Error*: the position of each point was randomly perturbed with uniform probability within a 0.2 unit radius;

In the tests, it is assumed that the type of sought shape is known, and the goal is to find the shape size and orientation. That is, each algorithm is run three times on each data set, each time to fit one specific kind of shape. Each time an algorithm is run on one data set, it is shown only the subset

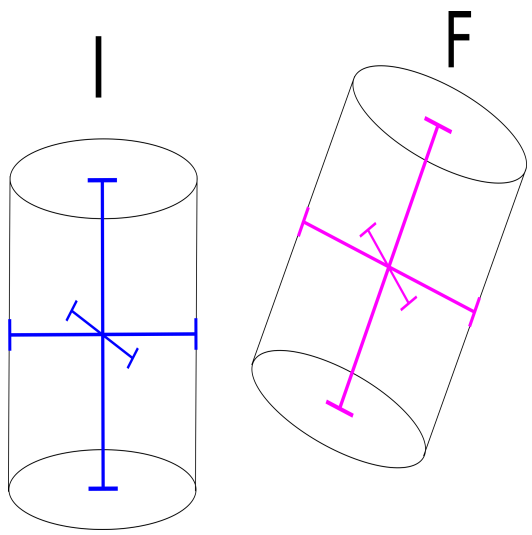


Figure 3.3: The found shape F is compared to the real shape I projecting its height, depth, and width onto the height, depth, and width of the real shape.

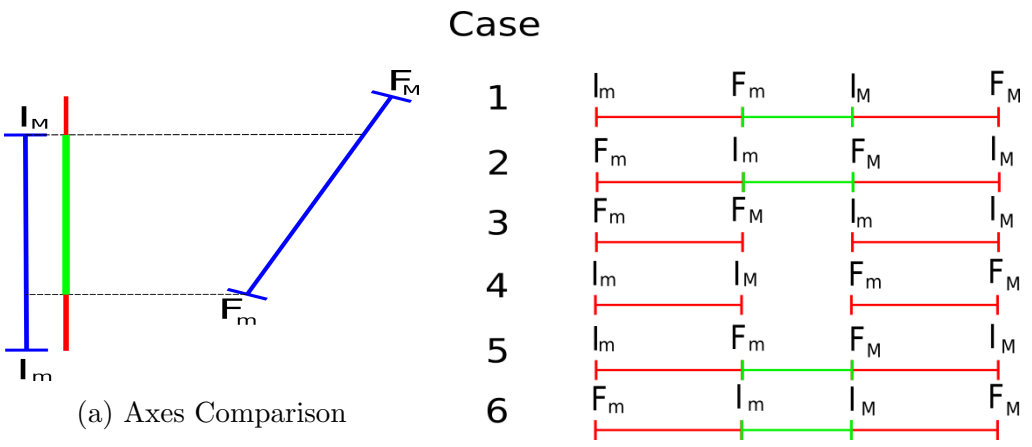
of shapes that needs to be fitted: for example, if the algorithm is required to find the size and orientation of cylinders in the *clean* data set, only the subset of 190 cylinder models will be used.

3.2.2 Error Evaluation Function

The best scoring solution F found by an algorithm in a point cloud (i.e. the *found shape*) is compared with the *real shape* I, namely the reference shape used to generate said point cloud.

A fair evaluation of the solutions needs to take into account several issues. First, it involves the comparison of parameters of mixed units (angle degrees for orientation, linear units for size and position), and thus standard metrics (e.g Euclidean distance) would not be appropriate. Second, differences in the centre or orientation have a larger impact on the matching of the two shapes (and hence on robotic manipulation) than a comparable offset in the size parameters. Finally, the data sets include shapes with differences in size up to one order of magnitude, and the evaluation function should be invariant to size.

The error evaluation function considers the match and alignment of the



(a) Axes Comparison

(b) Different Cases

Figure 3.4: Projection of the i^{th} segment of F (F^i) onto the j^{th} segment of I (I^j). That is, the intersection of the projections of F^i and I^j onto the j^{th} Cartesian axis. The green part of the segment marks the match (intersection) of the two projections, the red parts the mismatch. In case of perfect match, the green part will be equal to the length of I^j, and there will be no red parts. There are six possible cases of partial or no match between the two axes.

three segments corresponding to the height, width, and depth of the F and I shapes. Each of the three segments is placed along one of the principal axes of symmetry of the shape (Figure 3.3). In the case of the box, the lengths of the three segments correspond to the three size parameters of the solution, in case of the cylinder the length of one segment corresponds to the height and the other two to the diameter of the circular section, in case of the sphere they are all equal to the diameter. If the principal axes are not unique (e.g. the three axes of the sphere), they are aligned with the Cartesian axes of the reference frame.

The error is measured from the overlap between the *projection* of each segment of F onto the three segments of I. In case of perfect matching and alignment, each segment of F (e.g. the height of a box) will project exactly onto the corresponding segment of I, and on a point (i.e. zero overlapping) onto the other two segments (the width and depth) of I. In case of misalignment or incorrect dimensions of F, the projection of one segment (e.g. the height of a box) will not cover exactly the corresponding segment (the

height) of I (Figure 3.4a), and will be non-zero on the other two segments (the width and depth).

Given a solution F, let us denote as F^1, F^2 and F^3 its three segments (the width, depth, and height respectively), sorted in decreasing order of length, and as $\hat{F}^i_j$ the projection of the i^{th} segment of F on the j^{th} Cartesian axis, where $j = 1$ denotes the X axis, $j = 2$ denotes the Y axis, and $j = 3$ denotes the Z axis. Likewise, for the three axes of I. Finally, let us denote henceforth as $|A|$ the length of a given segment A.

To simplify the calculations, a rigid transformation is applied to express F and I in a new Cartesian frame that corresponds to the three principal axes of I. Note that now $|\hat{I}^i_i| = |I^i|$ and $|\hat{I}^i_k| = 0 \ \forall k \neq i$ The error $Err(F, I)$ in the alignment and match of F to I is calculated as follows:

$$Err(F, I) = \min_{i=\{1,2,3\}} \left\{ \min_{\substack{k=\{1,2,3\} \\ k\neq i}} \left\{ M(F^i, I^i) - E(F^i, I^k) \right\} \right\}$$

$M(F^i, I^i)$ denotes the matching and alignment of F^i with the corresponding segment I^i, and is calculated as the length of the intersection $\hat{F}^i_i \cap \hat{I}^i_i$ (green sub-segment in Figure 3.4a), minus the sum of the lengths of the non-intersecting parts of $\hat{F}^i_i$ and $\hat{I}^i_i$ (red sub-segments in Figure 3.4a).

$$M(F^i, I^i) = \frac{|\hat{F}^i_i \cap \hat{I}^i_i| - \left[|\hat{F}^i_i| - |\hat{F}^i_i \cap \hat{I}^i_i|\right] - \left[|\hat{I}^i_i| - |\hat{F}^i_i \cap \hat{I}^i_i|\right]}{|\hat{I}^i_i|} \tag{3.8}$$

Note that if the found shape F matches perfectly I, $|\hat{F}^i_i \cap \hat{I}^i_i| = |\hat{I}^i_i|$ (F^i is aligned with I^i), all the other terms are equal to zero, and $M(F^i, I^i) = 1$. In case of total mismatch, $|\hat{F}^i_i \cap \hat{I}^i_i| = 0$ and $M(F^i, I^i) = -\frac{|\hat{F}^i_i|+|\hat{I}^i_i|}{|\hat{I}^i_i|} < -1$.

$E(F^i, I^k)$ denotes the mismatch and misalignment of F^i, and is measured as the length of the intersection of its projection with the non-corresponding axes I^k of I. That is:

$$E(F^i, I^k) = \max_{\substack{k=\{1,2,3\} \\ k\neq i}} \left\{ \frac{|\hat{F}^i_k \cap \hat{I}^k_k|}{|\hat{I}^k_k|} \right\} \tag{3.9}$$

Note that if the found shape F corresponds perfectly to I, $|\hat{F}^i_k \cap \hat{I}^k_k| = 0$ for all $i \neq k$ (F^i is aligned with I^i), and $E(F^i, I^k) = 0$. In case F^i is

perpendicular to I^i (total mismatch), F^i will be aligned with one of the I^k and $E(F^i, I^k) = 1$.

Equation (3.8) is equal to zero in case F corresponds to I ($M(F^i, I^i) = 1$ and $E(F^i, I^k) = 0$), is greater than zero otherwise, and is maximum in case of total mismatch ($M(F^i, I^i) < -1$ and $E(F^i, I^k) = 1$). The max and min operations are meant to penalise the main mismatches in length and alignment, whilst being more forgiving on minor discrepancies.

3.2.3 Parameters Used

The parameterization of the two metaheuristics has been optimised via extensive trial and error, and is shown in Table 3.2. It is different for each shape type, but the same across the three data sets (noisy, error, double error, Section 3.2.1). The two metaheuristics have been parameterized so as they sample the same number of solutions in one complete optimisation trial.

Evolutionary Algorithm			
Parameter	**Sphere**	**Box**	**Cylinder**
# Individuals	10	10	25
# Parents	3	3	8
Mutation Rate (p^f)	1	1	1
# Iterations	390	900	672
Sampling Coverage	5%	25%	25%
Bees Algorithm			
Parameter	**Sphere**	**Box**	**Cylinder**
Scout bees (ns)	2	3	4
Elite sites (ne)	1	1	1
Best sites (nb)	2	3	4
Recruited elite (nre)	9	10	10
Recruited best (nrb)	4	4	6
Stagnation limit ($stlim$)	20	30	25
Initial patch neighbourhood (ngh)	0.15	0.5	1
# Iterations	300	500	600
Sampling Coverage	5%	25%	25%

Table 3.2: Parameterization of the Bees Algorithm and Evolutionary Algorithm. These parameters are formalised in the traditional way. In Section 4.1 will be presented as an alternative formalisation for the BA parameters.

3.3 Results

The five number summary of the primitive fitting tests is shown for each algorithm in Table 3.3 for the *clean*, *error*, and *double error* data sets.

In terms of accuracy (median value), the Bees Algorithm performed particularly well on spheres and boxes, where it obtained errors that were smaller than or comparable to the errors obtained by the EA and RANSAC. On cylinders, although the performances of the Bees Algorithm and RANSAC were close, the latter obtained the best results.

In terms of consistency (first and third quartiles), RANSAC and the Bees Algorithm performed comparably on boxes and spheres, whilst RANSAC was clearly superior in the fitting of cylinders.

Overall, the EA was the least accurate and consistent of the three algorithms on all data sets and shapes. All the three algorithms proved robust to error, as their performances on the *clean* data set are indistinguishable from those obtained on the *error* and *double error* data sets.

Clean						
Shape	**Algorithm**	**Min**	**First Quartile**	**Median**	**Third Quartile**	**Max**
Sphere	Evolutionary Algorithm	1.055×10^{-3}	8.898×10^{-3}	1.410×10^{-2}	2.135×10^{-2}	8.831×10^{-2}
Sphere	Bees Algorithm	1.270×10^{-5}	1.594×10^{-4}	2.585×10^{-4}	4.132×10^{-4}	1.625×10^{-3}
Sphere	RANSAC	0	0	0	0	0
Box	Evolutionary Algorithm	8.170×10^{-3}	2.517	2.815	3.514	1.217×10^{1}
Box	Bees Algorithm	7.828×10^{-2}	1.903	2.353	2.938	1.246×10^{1}
Box	RANSAC	1.490×10^{-8}	2.500	2.714	3.250	7.000
Cylinder	Evolutionary Algorithm	8.893×10^{-2}	1.954	4.441	8.938	1.071×10^{3}
Cylinder	Bees Algorithm	2.069×10^{-1}	1.824	3.709	7.667	1.052×10^{3}
Cylinder	RANSAC	1.902	2.481	2.696	3.162	1.115×10^{2}
Single Error						
Shape	**Algorithm**	**Min**	**First Quartile**	**Median**	**Third Quartile**	**Max**
Sphere	Evolutionary Algorithm	1.075×10^{-3}	9.255×10^{-3}	1.435×10^{-2}	2.205×10^{-2}	1.013×10^{-1}
Sphere	Bees Algorithm	1.109×10^{-4}	1.022×10^{-3}	1.690×10^{-3}	2.885×10^{-3}	3.689×10^{-2}
Sphere	RANSAC	3.061×10^{-4}	3.462×10^{-3}	5.391×10^{-3}	9.289×10^{-3}	5.755×10^{-2}
Box	Evolutionary Algorithm	7.724×10^{-3}	2.496	2.802	3.517	1.212×10^{1}
Box	Bees Algorithm	3.498×10^{-2}	1.872	2.348	2.938	1.167×10^{1}
Box	RANSAC	1.175×10^{-2}	2.230	2.511	2.955	7.106
Cylinder	Evolutionary Algorithm	8.816×10^{-2}	1.954	4.478	9.032	1.054×10^{3}
Cylinder	Bees Algorithm	1.141×10^{-1}	1.813	3.758	7.711	1.140×10^{3}
Cylinder	RANSAC	6.959×10^{-2}	2.222	2.426	2.779	1.082×10^{3}
Double Error						
Shape	**Algorithm**	**Min**	**First Quartile**	**Median**	**Third Quartile**	**Max**
Sphere	Evolutionary Algorithm	9.426×10^{-4}	9.769×10^{-3}	1.522×10^{-2}	2.319×10^{-2}	1.018×10^{-1}
Sphere	Bees Algorithm	2.037×10^{-4}	1.838×10^{-3}	3.140×10^{-3}	5.610×10^{-3}	7.257×10^{-2}
Sphere	RANSAC	5.078×10^{-4}	6.800×10^{-3}	1.066×10^{-2}	1.776×10^{-2}	1.077×10^{-1}
Box	Evolutionary Algorithm	1.795×10^{-2}	2.482	2.768	3.471	1.210×10^{1}
Box	Bees Algorithm	8.176×10^{-2}	1.899	2.351	2.991	1.200×10^{1}
Box	RANSAC	2.545×10^{-2}	2.081	2.425	2.843	7.272
Cylinder	Evolutionary Algorithm	8.970×10^{-2}	1.984	4.514	9.347	1.095×10^{3}
Cylinder	Bees Algorithm	1.251×10^{-1}	1.880	3.884	8.076	1.170×10^{3}
Cylinder	RANSAC	1.562×10^{-1}	2.070	2.339	2.693	8.833×10^{2}

Table 3.3: Five number summary of the primitive fitting results obtained by the Evolutionary Algorithm, Bees Algorithm and RANSAC on the three data sets.

3.4 Discussion

The state-of-the-art RANSAC algorithm is widely used because of its ability to precisely fit shapes to PC models. The performance of RANSAC in terms of accuracy and speed strongly depends on a number of ad hoc assumptions like the tolerance threshold. The results presented in Section 3.3 proved that the Bees Algorithm is able to obtain results of quality comparable to those obtained using RANSAC, without the need of assumptions. Compared to another state-of-the-art metaheuristics (EA), the Bees Algorithm was able to fit primitive shapes to PC scenes with greater accuracy and consistency.

Although the Bees Algorithm had been optimised for speed, single shape fitting times were in the order of fractions of a second, and thus fully compatible with real-time operations. If needed, optimisation and parallelisation would boost the efficiency of the Bees Algorithm.

The Bees Algorithm showed also considerable robustness to error, which simulated imprecision in laser scanning devices. Overall, the tests presented in this chapter offer a first indication of the capability of the Bees Algorithm to solve effectively and efficiently the primitive fitting problem, with performances comparable or better than the state-of-the-art. The strength of the Bees Algorithm is that it does not need any assumption to be made on the model.

It should also be noted that in the comparison of Section 3.3 RANSAC was advantaged by the initialisation subroutine. Candidate shapes were in fact initialised with points sampled from the PC, making sure that these points lied on opposite sides of the shape in Figure 3.2. It is arguable that a similar seeding of the candidate solutions could boost the performance of the Bees Algorithm and the EA.

3.5 Conclusions

In this work the ability of the Bees Algorithm to solve the primitive fitting problem was evaluated. The performance of the Bees Algorithm was tested on the recognition of three kinds of primitive shapes from artificially generated data sets, and compared to the performance of the state-of-the-art RANSAC algorithm and the EA metaheuristics.

The tests showed that the Bees Algorithm is more precise and consistent than the EA, and performs with comparable accuracy to and consistently

as RANSAC. Although not optimised for speed, the efficiency of the Bees Algorithm was compatible with real-time applications. Like the other two algorithms, the Bees Algorithms showed considerable robustness to error in the PC models. This result indicates the suitability of the Bees Algorithm to handle data from noisy and imprecise sensors.

The main advantage of the Bees Algorithm over techniques like RANSAC is that it doesn't need ad hoc assumptions to be made on the models. In particular, RANSAC is sensitive to the choice of the error tolerance threshold, which usually requires careful optimisation for top performance. RANSAC needs also a seeding procedure to generate the candidate shape, which is then scored on its fit to the rest of the PC model. In its present implementation, the Bees Algorithm does not need any seeding of the candidate solutions.

The tests performed in this work featured only PCs representing single objects. Further tests will be carried out to investigate the ability of the Bees Algorithm to recognise multiple and possibly different shapes in a scene. One possible scheme would be to carry out parallel searches for different shapes, for example one kind of shape for each neighbourhood. At the end, the best fitting shapes for each region of the scene would be retained. The performance of the Bees Algorithm on partial shapes needs also to be evaluated, in order to assess its suitability to cluttered environments.

Finally, the current implementation of the Bees Algorithm is also extendable to other kinds of shapes, as long as a measure of the distance of the points from the candidate shape can be expressed, and the concordance of the normals can be evaluated.

Chapter 4

Bees Algorithm: A Theoretical Analysis

This chapter is dedicated to a formal analysis of the Bees Algorithm. In Section 4.1 the Bees Algorithm is formalised using a rigorous mathematical description, beyond the qualitative biological metaphor. Section 4.2 analyses the properties of the local search performed by a single site, providing a mathematical definition and a discussion on the site reach and expected progress. In Section 4.3 the effect of the site abandonment procedure is discussed, and the stalling probability is computed under mild assumptions and in presence and absence of the neighbourhood shrinking procedure. In the same section, the implications of choosing a large stagnation limit *stlim* or a high *nr* are discussed. Finally, Section 4.4 focuses on the neighbourhood shape, and the implications of using an isotropic neighbourhood instead of the standard anisotropic hypercube, has in high dimensional space.

4.1 Formal Definition of the Bees Algorithm

The Bees Algorithm iteratively looks for better solutions to a specified optimisation problem. The algorithm is terminated when a given stopping criterion is met (e.g. a pre-set number of optimisation cycles has elapsed, a solution of satisfactory quality is found). Despite minor differences, the notation concerning the main parameters and operators of the Bees Algorithm is consistent in the literature. With some minor changes, it is also used in this chapter:

- **ns** number of scout bees used *only* in the global search;
- **nb** number of sites where local search is performed;
- **nr** number of recruited forager bees for each of the nb sites;
- **stlim** number of cycles of local stagnation before a site is abandoned;
- **ngh** initial neighbourhood size of the nb sites;
- α neighbourhood shrinking parameter ($0 < \alpha < 1$);

In the standard Bees Algorithm, the parameter ns describes the total number of scouts used for random exploration (here ns) plus the number of scouts (nb) marking the neighbourhoods (sites) selected for local search. That is, $ns^{standard} = ns + nb$. Also, it is customary to allocate a larger number of foragers (nre) to the very best $ne < nb$ (*elite*) sites, and less ($nrb < nre$) to the remaining $nb - ne$ best sites. This distinction is not necessary for the analysis proposed in this chapter, and for the sake of compactness is dropped. Henceforth, the parameter nr will refer likewise to nre or nrb.

In this study only continuous optimisation is considered, and each solution is represented by an N-dimensional vector of real-valued decision variables $s^g = \{s^g[1], ..., s^g[N]\} \in \mathbb{R}^n$. The solutions are evaluated by a fitness function $\mathscr{F}$ specific to the problem domain, which the algorithm aims to maximise. The analysis of this chapter is equally valid for a minimisation problem ($\min\{\mathscr{F}(\cdot)\} \equiv \max\{-\mathscr{F}(\cdot)\}$).

In this chapter, each of the $s \in \{s^{(1)}, \ldots, s^{(nb)}\}$ nb sites selected for local search is denoted by a centre s^g and two additional variables: the *time-to-live* integer variable s^{ttl}, and the local search *edge* s^e. The time-to-live variable s^{ttl} is a counter that indicates the number of remaining cycles of stagnation before the site is abandoned. The edge s^e defines the current spatial extent (henceforth called *search scope*) of the local search.

For the sake of simplicity, unless otherwise stated, all the decision variables will be henceforth defined in the same interval. Accordingly, s^e and ngh are scalars, and the search scope at a given site s is delimited by a *hypercube* $\mathcal{C}$ of edge s^e centred in the solution s^g. Hereafter, this region will be indicated as $\mathcal{C}(s^g, s^e)$. Local search is performed uniformly sampling nr solutions inside $\mathcal{C}(s^g, s^e)$. In the general case that the interval of definition is not equal for all parameters, s^e and ngh will be defined as vectors of size N. In this case, local search is performed inside a *box* (i.e. an N-*orthotope*) of

edges $s^e = \{s^e[1], \ldots, s^e[N]\}$ centred in s^g. When relevant, the consequences of using box sampling rather than cubic sampling will be discussed.

The algorithm steps are described in box 4.1. Except for minor changes (i.e. no *elite* sites), the procedure described in box 4.1 can be regarded as the Standard BA (Pham and Castellani, 2009). The initialisation and site abandonment procedures are designed to keep constant at each generation the sampling rate of the solution space[1].

Local search aims to find the fitness optimum within a neighbourhood centred on a promising solution. Because the centre of the neighbourhood is updated as better solutions are found (step 3), the scope of local search dynamically changes, and eventually includes the local attractor point in the search space (i.e. a local optimum). It should be noted that, like any stochastic optimisation procedure, local search is not guaranteed to stop at the local optimum. In particular, local search may be prematurely abandoned when *(i)* it stagnates for *stlim* iterations (e.g. stops on a flat surface) or *(ii)* global search finds more promising regions (fitter solutions) elsewhere in the search space (step 2).

Global random search aims to find previously unexplored regions of high fitness in the search space. Global search can also be used to increase adaptation to changes in dynamic fitness landscapes. The solutions found via local (i.e. the centres of the nb neighbourhoods) and global search are ranked at the end of every optimisation cycle, and the fittest nb solutions are kept as seeds (centres) for the next optimisation cycle. As the local exploitation of one given site progresses, the probability that this site is abandoned because random search found a fitter solution decreases. For this reason, some authors do not use global search (Pham and Castellani, 2009), or give randomly generated solutions (*young bees*) time to 'grow up' (Castellani et al., 2012).

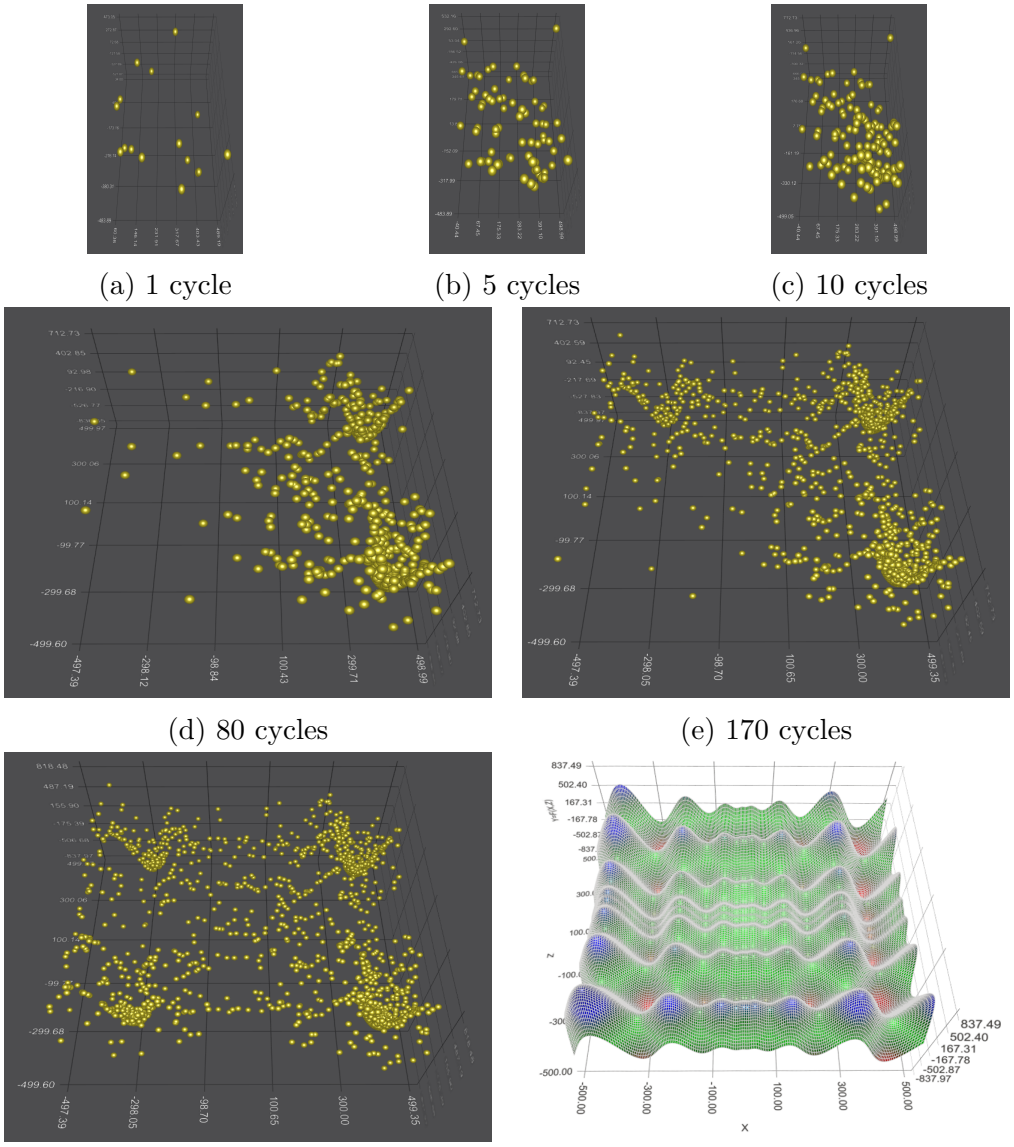
[1] This is particularly useful when the BA performances are compared with other techniques.

[2] If the interval of definition of the parameters is not the same for all variables, the i-th component of the vector s^e is initialised as $s_i^e = ngh \cdot (M_i - m_i)$ where M_i and m_i are the maximum and minimum value of the interval of definition of the i-th parameter.

Bees Algorithm: Main Steps

1. (**Initialisation**) The initial population P is created sampling $ns + nr \cdot nb$ solutions at random across the N-dimensional solution space. Each solution s^g marks the centre of a neighbourhood (*site*) s of edge s^e (i.e. a *search scope* $\mathcal{C}(s^g, s^e)$). The two variables s^{ttl} and s^e are initialised for each site as follows: $s^{ttl} = stlim$ and $s^e = ngh \cdot (M - m)$, where M and m are respectively the upper and lower limit of the interval of definition of the variables;
2. (**Selection**) The best nb sites $s^{(1)}, \ldots, s^{(nb)}$ centred on the solutions of highest fitness are selected from P for local search (**Waggle Dance**), the others are removed from the population;
3. (**Local Search**) For each of the nb sites $s \in \{s^{(1)}, \ldots, s^{(nb)}\}$ in P selected for local search, the following steps are performed:
   (a) If $s^{ttl} = 0$ the site is abandoned (**Site Abandonment**), and nr new solutions are randomly sampled across the search space. The best v of these nr solutions is used as the centre for a new site s, which is initialiased with $s^{ttl} = stlim$ and $s^e = ngh \cdot (M - m)$;
   (b) **(Foraging)** If $s^{ttl} > 0$, nr solutions $v_1, \ldots, v_{nr}$ are randomly sampled with uniform distribution within $\mathcal{C}(s^g, s^e)$. The solution v of highest fitness is selected, whilst the other solutions are discarded:
      i. if $\mathcal{F}(v) > \mathcal{F}(s^g)$, v replaces s^g as the site centre ($s^g = v$). The time to live is set to $s^{ttl} = stlim$ and the edge is kept unchanged (s^e);
      ii. If $\mathcal{F}(v) \leq \mathcal{F}(s^g)$, the local search is said to stagnate. The edge is reduced of a fixed factor[1] $s^e = \alpha s^e$ (**Neighbourhood Shrinking**), and the time to live is reduced to $s^{ttl} = s^{ttl} - 1$;
4. (**Global Search**) ns new (*scout*) solutions $v_1, \ldots, v_{ns}$ are uniformly sampled in the search space. They become the centres of ns new sites, each initialised with $s^{ttl} = stlim$ and $s^e = ngh \cdot (M - m)$.
5. (**Population Update**) The nb scouts marking the centres of the sites evolved via local search (3) and the ns scouts created by global search (4) make up the new population P;
6. (**Stopping Criterion**) if the stopping criterion is not met go back to step 2, otherwise return the solution of highest fitness found;

Algorithm 4.1: The Bees Algorithm



(a) 1 cycle (b) 5 cycles (c) 10 cycles

(d) 80 cycles (e) 170 cycles

(f) 200 cycles (g) $\mathscr{F}$ function

Figure 4.1: Example of space explored by the BA using only one site, displayed at different cycle numbers. As it is possible to observe, in the first cycles (Figures 4.1a to 4.1c) the site explore only the right-bottom part of the function. When the site stall, another region of the function with a strong optimum is explored (Figure 4.1d). Overtime, even with just one site, the BA is able to explore the most important (Figures 4.1e and 4.1f) local optima regions.

4.2 Analysis of Local Search Properties

This section clarifies the properties of the local search procedure of the Standard BA, such as its average step size and speed of convergence. These properties are estimated from an *a-posteriori* analysis on the 'lifetime' of a generic site s, from its discovery by a scout to abandonment when local search *stalls* (i.e. it fails to progress for *stlim* iterations). The case that the site is replaced by a more promising site found via global search is not included. If needed, the results of the below analysis are applicable to describe the behaviour of local search from any point in time, not necessarily the discovery of the site, until abandonment. In this context the site is analysed both in terms of local search and the effect of the sequence of local searches performed during its lifetime. Importantly, in contrast with a single local search, bounded by design, the sequence of local searches performed by a single site can cover the whole search space. This is illustrated with the example in Figure 4.1, where the explored space (i.e. all the sampled solutions) is shown in the case of a BA with a single site, using the following parameters: $ns = 0, ne = 0, nb = 1, nrb = 15, stlim = 10$ and $ngh = 1$; applied to minimise the Schwefel (Schwefel, 1981) fitness function. Moreover, it is worth noting that the final solution may not be the local optimum, that is, the sequence of local searches may only provide an approximation of the local optimum.

4.2.1 Local Search: Introduction and Definitions

The following nomenclature will be used:

- s is the site, described at any iteration (*cycle*) t by a triple $s_t = \{s_t^g, s_t^e, s_t^{ttl}\}$: the centre s_t^g, the edge s_t^e and the time to live s_t^{ttl};
- n is the number of local search cycles from discovery to abandonment of the site;
- s_1^g is the starting point (site centre) of the local search procedure;
- s_t^g with $1 < t < n$ is the site centre after t local search cycles;
- s_n^g is the final result of the local search, namely, the neighbourhood centre at the last local search cycle, before the site is abandoned (i.e. $s_n^{ttl} = 0$);

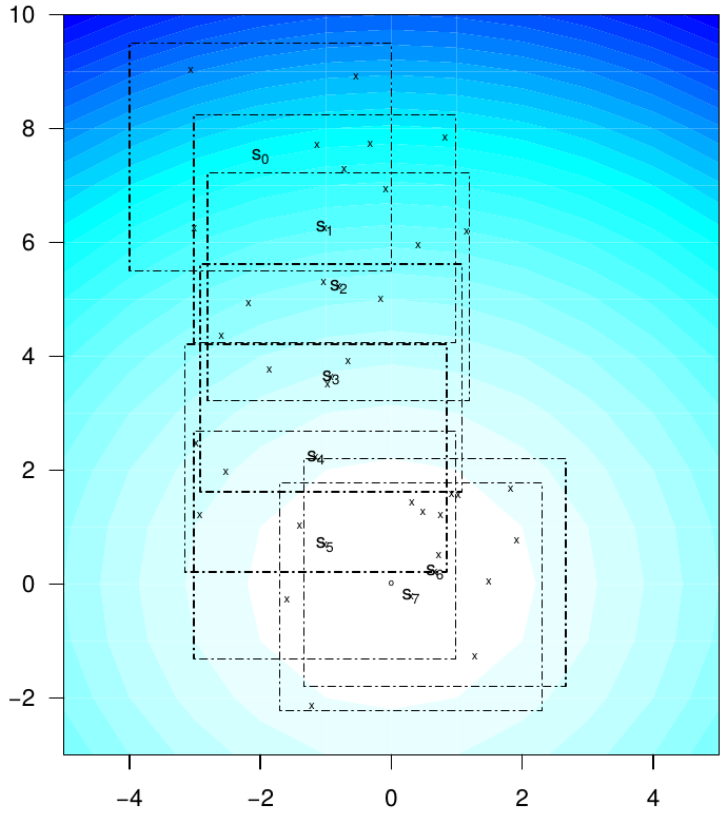


Figure 4.2: Example of sequence of solutions S found by a site. The squares represent the local search boundaries and the marks represent the forager bees sampled in the local searches.

- S is the series of solutions found by local search at site s:

$$S = \{s_1^g, \ldots, s_n^g\} \tag{4.1}$$

The solution found in the t^{th} local search cycle $L_{nr}(s_t^g)$ can be formalised as the result of the following endomorphic function (maximisation problem):

$$L_{nr}(s_t^g) = \arg\max_v \{\mathscr{F}(v) \mid v \in \{s_t^g, v_1, \ldots, v_{nr}\}, v_i \sim \mathcal{C}(s_t^g, s_t^e)\} \tag{4.2}$$

where $v_i \sim \mathcal{C}(s_t^g, s_t^e)$ is a uniform sampling of a solution v_i in the hypercube centred in s_t^g with edge s_t^e. This sampling will be further discussed in Section 4.4. Since this is an *a posteriori* analysis, it will be assumed that every set of candidate solutions sampled at each cycle is known.

A local maximum L_{opt} is defined as:

$$L_{opt} \text{ is a local maximum } \Leftrightarrow \exists \epsilon > 0 \mid \mathscr{F}(L_{opt}) \geq \mathscr{F}(v) \quad \forall v \in \mathcal{C}(L_{opt}, \epsilon) \tag{4.3}$$

If L_{opt} is the optimum of the subregion $\mathcal{C}(s_t^g, s_t^e)$, the operator L_{nr} provides a stochastic approximation of the local optimum within $\mathcal{C}(s_t^g, s_t^e)$. The expected quality of this approximation increases monotonically with the number nr of candidate solutions sampled.

$$L_{opt} = \lim_{nr \to \infty} L_{nr}(s) = \arg\max_{v} \{\mathscr{F}(v) \mid v \in \mathcal{C}(s_t^g, s_t^e)\} \tag{4.4}$$

The series of solutions S defined in Equation (4.1) and visible in the example in Figure 4.2 shares the same convergence properties of the LJ Search proved in (Gopalakrishnan Nair, 1979). Namely, without site abandonment, a number of steps n exists such that the series of solutions S will eventually converge to a local optimum.

Lemma 1 ($\mathscr{F}(s_i)$ series)**.** *The series of fitness scores $\mathscr{F}(s_i)$ for $i = 1, \ldots, n$ is monotonic increasing:*

$$\mathscr{F}(s_i) \geq \mathscr{F}(s_j) \Leftrightarrow i \geq j \qquad \forall i, j \in \{1, \ldots, n\} \tag{4.5}$$

with

$$\mathscr{F}(s_i) = \mathscr{F}(s_j) \Leftrightarrow s_i = s_j \qquad \forall i, j \in \{1, \ldots, n\} \tag{4.6}$$

and ordering the solutions by their fitness score is equivalent to ordering them by discovery, namely:

$$\nexists s_k \mid \mathscr{F}(s_i) \leq \mathscr{F}(s_k) \leq \mathscr{F}(s_{i+1}) \qquad \forall i \in \{1, \ldots, n-1\} \quad i, i+1 \neq k \tag{4.7}$$

Proof. Equations (4.5) and (4.6) can be proved directly by the definition in Equation (4.2). Let's assume that a solution s_k from Equation (4.7) exists, then we have either $k < i$ with $\mathscr{F}(s_i) \leq \mathscr{F}(s_k)$ or $k > i + 1$ with $\mathscr{F}(s_k) \leq \mathscr{F}(s_{i+1})$. Both cases are equally impossible due to the monotonously of the $\mathscr{F}$ series. □

Due to the monotonically increasing nature of the series $\mathscr{F}(s_1^g), \ldots, \mathscr{F}(s_n^g)$, s_n^g is the best solution found in the n iterations of local search at site s.

It is important to notice that whilst the higher an index $i \in \{1, \ldots, n\}$ is, the closer the fitness score $\mathscr{F}(s_i)$ is to the final score $\mathscr{F}(s_n)$, the same can't

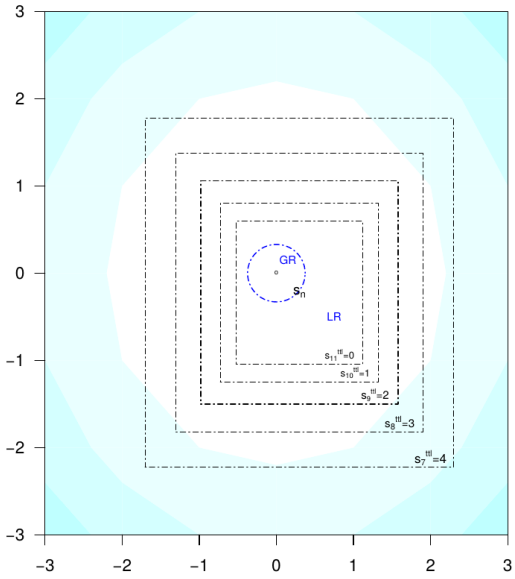


Figure 4.3: Example of neighborhood shrinking and how it affects the local search boundaries in a site. In this example the visible local part of the fitness function is an hypersphere.

be said about the distance to the final solution $d(s_i, s_n)$. That is, even in a concave function like $\mathscr{F}(x) = -x[0]^2 - x[1]$ with optimum in $s_n^g = [0, 0]$, we can have a situation with $s_{n-1}^g = [1, 5]$ and $s_{n-2}^g = [3, 1]$ where $\mathscr{F}(s_{n-1}) = -6 > -10 = \mathscr{F}(s_{n-2})$ and with $d(s_{n-1}, s_n) \approx 5.1 > 3.16 \approx d(s_{n-2}, s_n)$.

The standard neighbourhood shrinking heuristic can be formally defined for a hypercube as follows ($0 < \alpha < 1$):

$$s_{t+1}^e = \begin{cases} s_t^e & L_{nr}(s_t^g) \neq s_t^g \\ \alpha s_t^e & L_{nr}(s_t^g) = s_t^g \end{cases} \tag{4.8}$$

Neighbourhood shrinking (illustraded in Figure 4.3) can be similarly defined in the more general case of *box* sampling, where the interval of definition is not the same for all the N variables. In this latter case, s_t^e is the N-dimensional vector of search scope edges, and each edge is reduced of the same fixed factor α. It also holds for isotropic (hyperspherical) sampling. The standard site abandonment heuristic is applied when:

$$s_n^g = s_{n-1}^g = \cdots = s_{n-stlim}^g \tag{4.9}$$

In other terms, the local search is terminated when *stlim* consecutive *fix points*[2] of L_{nr} are found. It is also true that:

$$s_t^g = s_{t+k}^g \;\Leftrightarrow\; s_t^g = s_{t+1}^g = \cdots = s_{t+k}^g \tag{4.10}$$

for any positive index $k \in \mathbb{N}$.

The above definitions and properties are shared by most BA techniques, or can be easily modified to include other BA variants.

Proposition 1 (Search scope Intersection: Successive Solutions)**.** *Given two consecutive solutions s_t^g and s_{t+1}^g, s_t^g is included in the search scope of s_{t+1}^g*

$$s_t^g \in \mathcal{C}(s_{t+1}^g, s_{t+1}^e) \tag{4.11}$$

Proof. The proof is trivial: if local search stagnates at cycle t, $s_{t+1}^g = s_t^g$ and $s_{t+1}^e < s_t^e$ (neighbourhood shrinking). Then $s_t^g \in \mathcal{C}(s_{t+1}^g, s_{t+1}^e) = \mathcal{C}(s_t^g, s_{t+1}^e)$. If local search progresses at cycle t, $s_{t+1}^g \neq s_t^g$ and $s_{t+1}^e = s_t^e$ (no neighbourhood shrinking). Remembering that $s_{t+1}^g \in \mathcal{C}(s_t^g, s_t^e)$, it follows that $s_t^g \in \mathcal{C}(s_{t+1}^g, s_t^e) = \mathcal{C}(s_{t+1}^g, s_{t+1}^e)$. □

This property also holds in case of box and hyperspherical sampling is used.

4.2.2 Bounds on Reach

Hereafter, the distance in the solution space that local search is able to cover at a given site in a given number of cycles will be indicated as the *reach* of local search. That is, the reach is the distance between the starting point of local search (s_1) and the best approximation of the local optimum after n cycles (s_n), namely $d(s_1, s_n)$. The upper and lower boundaries of the reach are defined as follows:

Proposition 2 (Reach)**.** *The reach of local search in n learning cycles at a given site s centred on solution s_1^g is bounded within the $\left[0, n\frac{s_1^e\sqrt{N}}{2}\right]$ interval, where s_1^e is the site edge at the start of the search.*

Proof. Minimum reach occurs when local search stalls since the very beginning, namely $s_t^g = s_{t+1}^g \; \forall t \in \{1, \ldots, n = stlim\}$, and thus $s_1^g = s_n^g$.

[2] A fix point x of a function f is a point such as $f(x) = x$

At cycle t, local search is bounded within the hypercube $\mathcal{C}$ centred in s_t^g, where the farthest solutions lie at the four vertexes of $\mathcal{C}$. To attain maximum reach, local search must progress at each cycle, so as the initial site edge s_1^e is not reduced (i.e. no neighbourhood shrinking). The maximum step size per cycle is bounded by the distance between the centre and a vertex of the N-dimensional hypercube $\mathcal{C}$, that is $d_v = s_t^e\sqrt{N}/2$. The upper bound of the reach at a given site is therefore n times d_v. □

Proposition 2 gives the boundaries of 'how far' local search can travel in n learning cycles. The maximum step size is achievable only when the segment that joins s_1^g to s_n^g is parallel to the diagonal of the hypercube $\mathcal{C}$, and every pair of subsequent solutions s_t^g and s_{t+1}^g $(1 \leq t < n)$ are distant $d(s_t^g, s_{t+1}^g) = s_t^e\sqrt{N}/2$. For example, this would be the case of a fitness landscape consisting of a sloped hyperplane aligned with the diagonal of the hypercube $\mathcal{C}$, or a hypersphere of centre c lying in the direction of one of the diagonals of the hypercube centred in s_1^g.

The reach is related to the convergence speed (i.e. the number of iterations used to reach the local attractor) of local search at a given site. If the distance between the centre of the site s_1 and the optimum L_{opt} is $d(s_1^g, L_{opt})$, according to Proposition 2 the minimum number of iterations n_{min} required to reach L_{opt} are:

$$n_{min}\frac{s_1^e\sqrt{N}}{2} = d(s_1^g, L_{opt}) \Rightarrow n_{min} = \frac{2d(s_1^g, L_{opt})}{s_1^e\sqrt{N}} \tag{4.12}$$

This is the upper bound on the convergence speed, and can be used to evaluate the efficiency of local search on different fitness landscapes.

In the more general case of asymmetric boundaries (i.e. box sampling), the maximum reach can be computed as follows:

$$max_{reach} = \frac{n}{2}\sqrt{\sum_{i=0}^{N-1} s_1^e[i]^2} \tag{4.13}$$

where $s_1^e[i]$ is the i-th component of vector s_1^e. Finally, it is worth mentioning that most - if not all - BA variants use a hypercube to define the scope of local search. Consequently, the foragers are sampled inside an anisotropic region, and the maximum reach depends on the orientation of the segment that joins s_1^g to L_{opt} respect to the diagonal of the hypercube.

4.2.3 Expected Progress

To understand the behaviour of local search, it is useful to calculate the expected step size of one iteration of the procedure under certain assumptions on the fitness surface.

Proposition 3 (Expected Step Size)**.** *Given a strict monotonic increasing one-dimensional fitness landscape in* $[0, \ell] \in \mathbb{R}$ *(e.g. a straight line), and a site centred in* $s_t^g = \ell/2$ *with edge* $s_t^e = \ell$*, the expected step size of one local search iteration (i.e. the average distance between* s_t^g *and* s_{t+1}^g*) is:*

$$d(s_t^g, s_{t+1}^g) = \ell \frac{0.5^{nr+1} + nr}{nr + 1} - \frac{\ell}{2} \tag{4.14}$$

Proof. The goal is to calculate the expected output of the stochastic local search operator $s_{t+1}^g = L_{nr}(s_t^g)$ defined in Equation (4.2), with the search scope within $[0, \ell]$. This output can be expressed as the following continuous random variable:

$$\begin{aligned} X &= \arg\max_{x_i} \{\mathscr{F}(s_t^g), \mathscr{F}(x_1), \ldots, \mathscr{F}(x_{nr})\} = \\ &= \arg\max_{x_i} \{\phi(x_1), \ldots, \phi(x_{nr})\} \end{aligned} \tag{4.15}$$

where:

$$x_i \sim U(0, \ell) \quad \text{and} \quad \phi(x_i) = \max\{\mathscr{F}(s_t^g), \mathscr{F}(x_i)\} \tag{4.16}$$

The expected value E of a continuous random variable X defined in the interval $[a, b]$ is computed (Ross, 2014) as:

$$E[X] = \int_a^b x \cdot PDF_X(x) dx \tag{4.17}$$

where $PDF_X(x)$ is the probability density function of X. The probability density function of a random variable is equal to the derivative of the cumulative distribution function $CDF_X(x)$. In this case:

$$CDF_X(x) = \prod_{i=1}^{nr} P(\phi(x_i) \leq x) = CDF_{\phi(x)}(x)^{nr} \tag{4.18}$$

where $P(\phi(x_i) \leq x)$ is the probability that one random sample x_i of the search scope is less or equal to x. Differentiating $CDF_X(x)$ and plugging the derivative into Equation (4.17):

$$E[X] = \int_0^l \phi(x) \cdot nr \cdot CDF_{\phi(x)}(x)^{nr-1} PDF_{\phi(x)}(x) dx \tag{4.19}$$

Using the *Law of the Unconscious Statistician* (Ross, 2014) is possible to make the following substitution:

$$\begin{aligned} E[X] &= \int_0^{\ell} \phi(x) \cdot nr \cdot CDF_{\phi(x)}(x)^{nr-1} PDF_{\phi(x)}(x) dx \\ &= \int_0^{\ell} \phi(x) \cdot nr \cdot CDF_{x}(x)^{nr-1} PDF_{x}(x) dx \end{aligned} \tag{4.20}$$

The cumulative distribution function and the probability density function of a random variable X sampled with uniform probability in $U(0, \ell)$ are:

$$CDF_U(x) = \begin{cases} 0 & x < 0 \\ \frac{x}{\ell} & 0 \leq x < \ell \\ 1 & y \geq \ell \end{cases} \qquad PDF_U(x) = \begin{cases} \frac{1}{\ell} & 0 \leq x \leq \ell \\ 0 & \text{elsewhere} \end{cases} \tag{4.21}$$

From Equation (4.15) it is known that $x \sim U(0, \ell)$, also $\mathscr{F}$ is assumed to be monotonic[3], therefore:

$$\begin{aligned} E[X] &= \int_0^{\ell} \max\{x, s_t^g\} \cdot nr \cdot CDF_U(x)^{nr-1} PDF_U(x) dx = \\ &= \int_0^{\ell/2} \ell/2 \cdot nr \cdot \left(\frac{x}{\ell}\right)^{nr-1} \frac{1}{\ell} dx + \int_{\ell/2}^{\ell} x \cdot nr \cdot \left(\frac{x}{\ell}\right)^{nr-1} \frac{1}{\ell} dx = \\ &= \frac{nr \cdot x^{nr}}{2nr(\ell^{nr-1})}\Big|_0^{\ell/2} + \frac{nr \cdot x^{nr+1}}{\ell^{nr}(nr+1)}\Big|_{\ell/2}^{l} = \\ &= 0.5^{nr+1}\ell + \frac{nr \cdot \ell(1 - 0.5^{nr+1})}{nr+1} = \\ &= \frac{nr0.5^{nr+1}\ell + 0.5^{nr+1}\ell + nr \cdot \ell - nr0.5^{nr+1}\ell}{nr+1} = \ell\frac{0.5^{nr+1} + nr}{nr+1} \end{aligned} \tag{4.22}$$

Therefore the average step size is:

$$d(s_t^g, s_{t+1}^g) = |s_{t+1}^g - s_t^g| = \ell\frac{0.5^{nr+1} + nr}{nr+1} - \frac{\ell}{2} \tag{4.23}$$

□

[3]In the proof the case of monotonic increasing fitness is considered. This is not a loss of generality since only the expected step size is considered, not the direction.

The result of Proposition 3 is valid for any locally monotonic fitness slope, as long as $\mathcal{C}$ is fully in the monotonic region. Its validity extends also to multi-dimensional surfaces such as regions of hyperplanes or hyperspheres. For this to happen, the search scope must be isotropic (i.e. a hypersphere), the fitness landscape must be strictly monotonic along the straight line joining the centre of $\mathcal{C}$ to the local fitness maximum, and the fitness landscape inside the search scope must be symmetric respect to said straight line. If these conditions are verified, the expected step size will be given by Equation (4.14) for the direction where the slope is monotonic, and zero (no bias) in the other directions. The above conditions apply in the common case where local search is climbing one side of a fairly regular hill or slope, but $\mathcal{C}$ does not include the fitness maximum yet.

Expected Step Size: Experimental Verification

The theoretical result of Proposition 3 was verified numerically (figure 4.4) on three 2D cases: on an inclined plane (leftmost column), near the top of a spherical hill (the hill top is at the border of the search scope, middle column), and far from the top of the spherical hill (rightmost column). The neighbourhood is a circle of radius 0.5 centred in $\{0.5, 0.5\}$. In all cases, the fitness surface is monotonically increasing along the horizontal diameter line, and symmetric with respect to that line. The number of foragers was varied ($nr = 1, 10, 20$). The plots show that the average (red triangle) of 10^3 independent local search trials is always in good agreement with the theoretical expectation (at the bottom of each sub-figure) along the horizontal diameter line (where the fitness landscape is monotonic), and aligned to the centre of the search scope in the vertical direction (i.e. no bias in the vertical direction).

Far from the peak, where the curvature of the sphere is small, the spread of the solutions on the fitness landscape is large, and indistinguishable from the spread on the planar surface. Near the hill top, where the curvature is large, the solutions are tightly clustered near the fitness maximum. This behaviour suggests that local search becomes increasingly focused and exploitative as it approaches the local fitness maximum.

Figure 4.4 also shows little difference between the spread of the solutions obtained using 10 and 20 foragers. Indeed, the expected step size grows in sublinear fashion with the number of foragers (eq: 4.14). Figure 4.5 shows how the average step size of 10^4 independent local search trials varies with the

number of foragers (nr). Also in this case, the search trials were performed in a circle of radius 0.5 centred in $\{0.5, 0.5\}$, and the plot shows the result of local search (s_{t+1}) along the direction of the slope of an inclined plane. The numerical averages (blue dot) are in good agreement with the theoretical expectations of Equation (4.14) (red line). The plot highlights how the result of local search quickly reaches the borders of the neighbourhood, that is the asymptotic value of 1. In general, it can be said that the size of the neighbourhood determines more than the number of foragers the ability of local search to quickly climb (descend) a fitness slope.

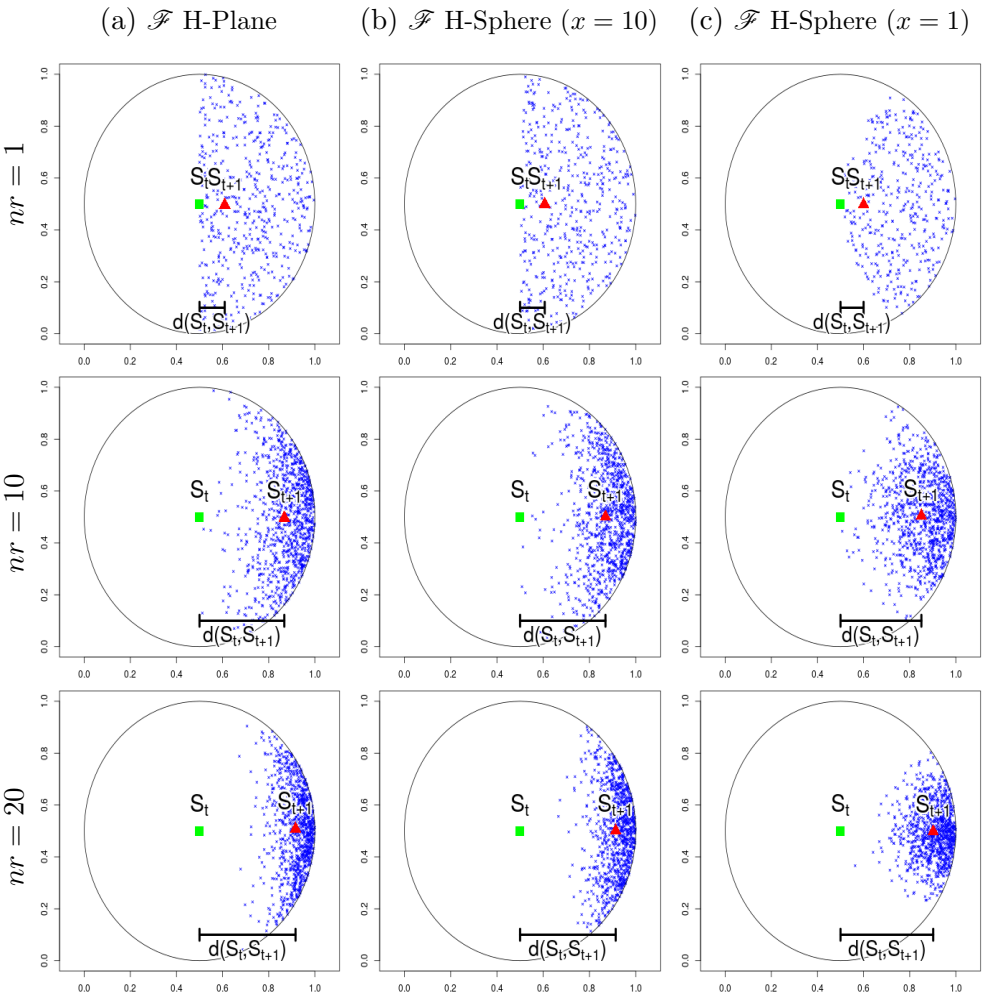


Figure 4.4: Search results (s_{t+1}) of 10^3 independent local search trials in three 2D fitness landscapes: a plane sloped in the horizontal direction (left column), a hypersphere with centre in $x = 10, y = 0.5$ (middle column), and a hypershpere centred in $x = 1, y = 0.5$ (right column). An isotropic circular search scope of centre $s_t^g = [0.5, 0.5]$ (green square) and radius $s_t^r = 0.5$ was used. The number of foragers nr was set to 1 (top row), 10 (middle row), and 20 (bottom row). The blue dots represent the solutions found in the local search trials, and their arithmetic average is marked by the red triangle. The maximum is always on the border of the search scope, at the right-end extreme of the horizontal diameter line. At the bottom of each panel, the expected step size (4.14) in the direction of the maximum is shown.

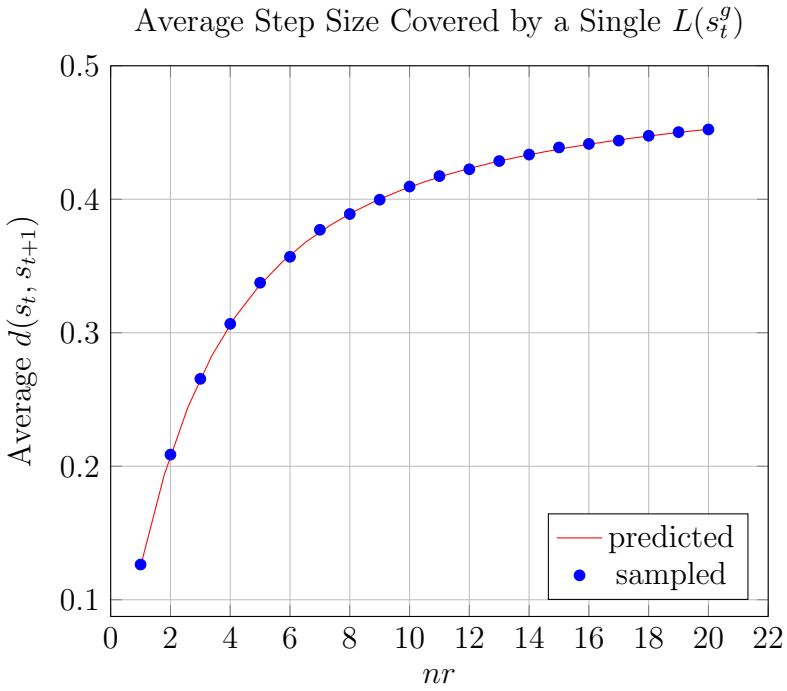


Figure 4.5: Result of local search using an isotropic search scope of radius $s_t^r = 0.5$ and centred in $s_t^g = \{0.5, 0.5\}$ on a sloped planar fitness surface. The predicted value (red line) along the direction of the slope was calcuated from Equation (4.14), and closely matches the average values of 10^4 independent local search runs (blue dots).

4.3 Site Abandonment: Local Search Stalling Probability

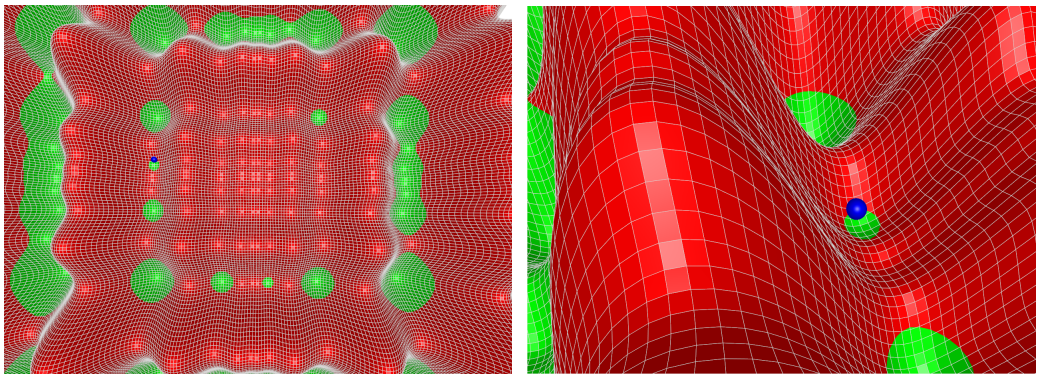


Figure 4.6: Example of $\mathcal{GR}_{s_t}$ (in green) and $\mathcal{LR}_{s_t}$ (in red) regions, as described on Equation (4.24), of a solution s_t (the blue dot) assuming $s_t^e = (M - m)$ and minimisation problem on the Schwefel (Schwefel, 1981) function.

This section analyses the probability that a site may be abandoned due to lack of progress of local search.

4.3.1 Site Abandonment: Definitions and Properties

Let s_t^g be the centre of site s at the t^{th} local search cycle and $\mathcal{C}(s_t^g, s_t^e)$ the search scope. Let $\mathcal{LR}_{s_t}$ and $\mathcal{GR}_{s_t}$ be the subregions of $\mathcal{C}$ including solutions of respectively lower or equal, and higher fitness. Explicitly:

$$\begin{aligned} \mathcal{LR}_{s_t} \subseteq \mathcal{C}(s_t^g, s_t^e) \quad & v \in \mathcal{LR}_{s_t} \Leftrightarrow \mathscr{F}(v) \leq \mathscr{F}(s_t^g) \\ \mathcal{GR}_{s_t} \subset \mathcal{C}(s_t^g, s_t^e) \quad & v \in \mathcal{GR}_{s_t} \Leftrightarrow \mathscr{F}(v) > \mathscr{F}(s_t^g) \end{aligned} \tag{4.24}$$

where

$$\mathcal{LR}_{s_t} \cup \mathcal{GR}_{s_t} = \mathcal{C}(s_t^g, s_t^e) \tag{4.25}$$

According to the above definitions, it can be said that local search progresses if the output of the endomorphism L_{nr} belongs to $\mathcal{GR}_{s_t}$:

$$\left[L_{nr}(s_t^g) = s_{t+1}^g \neq s_t^g\right] \Leftrightarrow s_{t+1}^g \in \mathcal{GR}_{s_t} \tag{4.26}$$

In general, $\mathcal{LR}_{s_t}$ and $\mathcal{GR}_{s_t}$ may include non-contiguous subregions, since the region covered by $\mathcal{C}$ may contain several local optima.

Hereafter, the volume of an N-dimensional region A will be indicated as $\mathcal{V}(A)$. To analyse the likelihood that local search stalls and the site is abandoned, it is useful to define the following two ratios:

$$|\mathcal{LR}_{s_t}| = \frac{\mathcal{V}(\mathcal{LR}_{s_t})}{\mathcal{V}\left(\mathcal{C}(s_t^g, s_t^e)\right)} \qquad |\mathcal{GR}_{s_t}| = \frac{\mathcal{V}(\mathcal{GR}_{s_t})}{\mathcal{V}\left(\mathcal{C}(s_t^g, s_t^e)\right)} \tag{4.27}$$

within the local search scope $\mathcal{C}(s_t^g, s_t^e)$, $|\mathcal{LR}_{s_t}|$ and $|\mathcal{GR}_{s_t}|$ represent the fraction of space where solutions of respectively lower and higher fitness lie. That is, they represent the *relative coverage* of $\mathcal{C}$ of the two regions $\mathcal{LR}_{s_t}$ and $\mathcal{GR}_{s_t}$. In particular, $|\mathcal{GR}_{s_t}|$ represents the probability that one random sample of the search scope yields a solution of higher fitness than s. From Equation (4.27), the following properties hold:

$$0 < |\mathcal{LR}_{s_t}| \leq 1 \quad 0 \leq |\mathcal{GR}_{s_t}| < 1 \qquad |\mathcal{LR}_{s_t}| + |\mathcal{GR}_{s_t}| = 1 \tag{4.28}$$

Also, from Equation (4.27) it follows that a solution s_t^g is the global optimum of the subregion $\mathcal{C}(s_t^g, s_t^e)$ if and only if:

$$\mathscr{F}(s_t^g) \geq \mathscr{F}(v) \;\; \forall v \in \mathcal{C}(s_t^g, s_t^e) \quad \Leftrightarrow \quad |\mathcal{LR}_{s_t}| = 1 \wedge |\mathcal{GR}_{s_t}| = 0$$

The local exploitative search of the BA aims to locate L_{opt}, inside the search scope $\mathcal{C}(s_t^g, s_t^e)$. If the $\mathcal{GR}_{s_t}$ region is significantly smaller than the search scope, the probability of finding a better solution than s^g is small, and progress may be slow or stop. The neighbourhood shrinking procedure may mitigate this problem, progressively reducing the search scope and increasing the probability that a forager is generated inside $\mathcal{GR}_{s_t}$. For this to happen, neighbourhood shrinking needs to keep $\mathcal{GR}_{s_t}$ inside the search scope. Unless it is a local optimum, it can be shown that the site centre s^g is at least contiguous to $\mathcal{GR}_{s_t}$. That is:

Proposition 4. *A solution s^g is either a local optimum of the fitness function $\mathscr{F}$, or lies on the border $\mathcal{GR}_{s_t}^-$ of $\mathcal{GR}_{s_t}$.*

Proof. This can be proven by contradiction:

$$s^g \notin \mathcal{GR}_{s_t}^- \Leftrightarrow \exists \epsilon > 0 \mid v \in \mathcal{LR}_{s_t} \quad \forall v \in B(s^g, \epsilon) \tag{4.29}$$

where $B(s^g, \epsilon)$ is an N-dimensional ball of radius ϵ and centred in s^g. However:

$$v \in \mathcal{LR}_{s_t} \Leftrightarrow \mathscr{F}(v) \leq \mathscr{F}(s^g) \tag{4.30}$$

so s^g is either a local optimum or is inside $\mathcal{GR}_{s_t}^-$. □

This property holds for virtually any BA variant. An important direct consequence is that, if s^g is not already a local optimum, there's no neighbourhood reduction that completely excludes $\mathcal{GR}_{s_t}$ from the search scope. Accordingly, neighbourhood shrinking does not affect the ability of local search to converge to a local optimum. However, it affects the upper bound of the convergence speed towards the optimum (eq. 4.12), and the likelihood that the search may remain trapped into a secondary peak. Thus, it can be said that neighbourhood shrinking is most beneficial in the latest iterations of the local search at a site, when a good local optimum - or the global optimum - has been approximately located and the search focus is shifted from convergence speed to the accuracy of the solution. In many cases, it can be argued that local search is indeed more likely to stall (and hence neighbourhood shrinking to be performed) once local search approaches the local peak, and $\mathcal{GR}_{s_t}$ becomes increasingly small.

4.3.2 Stalling Probability Without Neighbourhood Shrinking

Let us consider a site s centred on s_t^g at cycle t. The probability that local search without neighbourhood shrinking stalls at s will be henceforth indicated as $P(s_t^g = s^g_{t+s_t^{ttl}})$. It is computed as follows:

Proposition 5 (Stalling Probability Without Shrinking)**.** *Given site s centred on s_t^g at cycle t, the probability that local search without neighbourhood shrinking stalls is:*

$$P(s_t^g = s^g_{t+s_t^{ttl}}) = |\mathcal{LR}_{s_t}|^{nr \cdot s_t^{ttl}} \tag{4.31}$$

Proof. The site stalls if the search stagnates for the next s_t^{ttl} cycles, that is, if all the nr candidate solutions generated during s_t^{ttl} local search cycles lie in $\mathcal{LR}_{s_k}$. When local search stagnates, the centre of the site is unchanged, and if the search scope is not changed (no neighbourhood shrinking), $|\mathcal{LR}_{s_t}|$ is constant:

$$|\mathcal{LR}_{s_t}| = |\mathcal{LR}_{s_k}| \qquad \forall k \in \mathbb{N} \mid t \leq k < t + s_t^{ttl} \tag{4.32}$$

The joint probability that all the solutions sampled during one given cycle k of local search belong to $\mathcal{LR}_{s_k}$ is indicated as:

$$P(v \in \mathcal{LR}_{s_k}) \qquad \forall v \in \{v_1, \ldots, v_{nr}\} \quad v_i \sim \mathcal{C}(s_k^g, s_k^e) \tag{4.33}$$

Due to the uniform sampling, the probability that one solution is picked from $\mathcal{LR}_{s_k}$ corresponds to $|\mathcal{LR}_{s_k}|$. Remembering Equation (4.32), it follows that:

$$P(v \in \mathcal{LR}_{s_k}) = |\mathcal{LR}_{s_k}|^{nr} = |\mathcal{LR}_{s_t}|^{nr} \tag{4.34}$$

The stalling probability can then be computed as the joint probability of s_t^{ttl} consecutive stagnations of local search cycles:

$$P(s_t = s_{t+s_t^{ttl}}) = \prod_{k=t}^{t+s_t^{ttl}-1} |\mathcal{LR}_{s_k}|^{nr} = |\mathcal{LR}_{s_t}|^{nr \cdot s_t^{ttl}} \tag{4.35}$$

□

One important aspect of the stalling probability is that, since local search is random, it is not affected by the slope of the fitness surface. Proposition 5 is valid regardless whether $s_t^{ttl} = stlim$, that is, local search has not begun to stagnate yet, or $s_t^{ttl} < stlim$ and local search has already begun to stagnate.

Variants that use a dynamic assignment of foragers, like (Packianather et al., 2009; Pham and Darwish, 2010; Pham and Darwish, 2008), yield a more complex behaviour that leads to a different stalling probability formulation. Some ideas on how to deal with these variants will be discussed later in this section. If neighbourhood shrinking is used, the progressive reduction of the search scope needs to be taken into account. In this case, it is possible that if local search is trapped in a secondary peak, the $\mathcal{GR}_{s_t}$ region may be lost as the search scope is reduced.

4.3.3 Stalling Probability With Neighbourhood Shrinking

Let us consider the case where after t cycles, local search stagnates for k cycles at s_t^g inside the basin of attraction of a local optimum L_{opt} ($s_t^g \neq L_{opt}$ and $\mathcal{GR}_{s_t}$ is one unique region). In this case, Proposition 4 stipulates that s_t^g lies on the border of $\mathcal{GR}_{s_t}$. The most likely cause for repeated stalling of local search is that $\mathcal{GR}_{s_t}$ is small compared to $\mathcal{C}(s_t^g, s_t^e)$. However, if $\mathcal{GR}_{s_t}$ is small in comparison with $\mathcal{C}(s_t^g, s_t^e)$, and s_t^g is on the border of $\mathcal{GR}_{s_t}$, the distance d between s_t^g and L_{opt} is likely to be small compared to the search edge ($d(s_t^g, L_{opt}) \ll s_t^e$). That is, $\mathcal{GR}_{s_t}$ is likely to be far from the border of the search scope, and is not going to be changed by the neighbourhood shrinking procedure.

If the $\mathcal{GR}_{s_t}$ region at time t is unchanged after k successive applications of the neighbourhood shrinking procedure, it is possible to compute the relative coverages (eq. 4.27) of $\mathcal{LR}_{s_{t+k}}$ and $\mathcal{GR}_{s_{t+k}}$ as follows:

Lemma 2 (Coverage Reduction with constant $\mathcal{GR}_{s_t}$)**.** *Let s_t^g be the centre of site s at cycle t in the N-dimensional solution space. If local search stagnates for k cycles ($s_t^g = s_{t+k}^g$), and the region $\mathcal{GR}_{s_t} \neq \emptyset$ is not changed by neighbourhood shrinking, the relative coverages of $\mathcal{LR}_{s_{t+k}}$ and $\mathcal{GR}_{s_{t+k}}$ become:*

$$|\mathcal{LR}_{s_{t+k}}| = \frac{1}{\alpha^{kN}}(|\mathcal{LR}_{s_t}| - 1) + 1 = 1 - \frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}| \tag{4.36}$$

$$|\mathcal{GR}_{s_{t+k}}| = \frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}| \tag{4.37}$$

Proof. Since $\mathcal{GR}_{s_t}$ is not changed, $\mathcal{GR}_{s_t} = \mathcal{GR}_{s_{t+j}}$ and $\mathcal{V}(\mathcal{GR}_{s_t}) = \mathcal{V}(\mathcal{GR}_{s_{t+j}})$ $\forall j = 1, \ldots, k$. Also, remembering Equation (4.25):

$$\mathcal{LR}_{s_t} \cup \mathcal{GR}_{s_t} = \mathcal{C}(s_t^g, s_t^e) \Rightarrow \mathcal{V}(\mathcal{LR}_{s_t}) + \mathcal{V}(\mathcal{GR}_{s_t}) = \mathcal{V}(\mathcal{C}(s_t^g, s_t^e)) \tag{4.38}$$

However, neighbourhood shrinking reduces the local search edge of a factor α (Section 4.1 and eq. 4.8). That is, $s_{t+1}^e = \alpha s_t^e$ and after k successive repetitions of neighbourhood shrinking $s_{t+k}^e = \alpha^k s_t^e$. The volume of the search scope is reduced accordingly:

$$\mathcal{V}(\mathcal{C}(s_{t+k}^g, s_{t+k}^e)) = \mathcal{V}(\mathcal{C}(s_t^g, \alpha^k s_t^e)) = \alpha^{kN}\mathcal{V}(\mathcal{C}(s_t^g, s_t^e)) \tag{4.39}$$

Since $\mathcal{GR}_{s_t}$ is not reduced, the reduction $\mathcal{LR}_{s_t}$ will be equal to the reduction of $\mathcal{C}(s_t^g, s_t^e))$. That is:

$$\mathcal{V}(\mathcal{LR}_{s_{t+k}}) = \mathcal{V}(\mathcal{LR}_{s_t}) - \left(\mathcal{V}(\mathcal{C}(s_t^g, s_t^e)) - \mathcal{V}(\mathcal{C}(s_{t+k}^g, s_{t+k}^e))\right) \tag{4.40}$$

From the definition of the relative coverage (Equations (4.27) and (4.39)):

$$|\mathcal{LR}_{s_{t+k}}| = \frac{\mathcal{V}(\mathcal{LR}_{s_{t+k}})}{\mathcal{V}(\mathcal{C}(s_{t+k}^g, s_{t+k}^e))} = \frac{\mathcal{V}(\mathcal{LR}_{s_{t+k}})}{\alpha^{kN}\mathcal{V}(\mathcal{C}(s_t^g, s_t^e))} \tag{4.41}$$

And from Equation (4.40):

$$\begin{aligned}\frac{\mathcal{V}(\mathcal{LR}_{s_{t+k}})}{\alpha^{kN}\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))} &= \frac{1}{\alpha^{kN}}\cdot\frac{\mathcal{V}(\mathcal{LR}_{s_t})-(\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))-\mathcal{V}(\mathcal{C}(s_{t+k}^g,s_{t+k}^e)))}{\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))} = \\ &= \frac{1}{\alpha^{kN}}\cdot\frac{\mathcal{V}(\mathcal{LR}_{s_t})-(\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))-\alpha^{kN}\mathcal{V}(\mathcal{C}(s_t^g,s_t^e)))}{\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))} = \\ &= \frac{1}{\alpha^{kN}}\cdot\frac{\mathcal{V}(\mathcal{LR}_{s_t})}{\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))}-\frac{(1-\alpha^{kN})}{\alpha^{kN}}\cdot\frac{\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))}{\mathcal{V}(\mathcal{C}(s_t^g,s_t^e))} = \\ &= \frac{1}{\alpha^{kN}}\cdot|\mathcal{LR}_{s_t}|-\frac{(1-\alpha^{kN})}{\alpha^{kN}}\end{aligned} \tag{4.42}$$

Equation (4.36) is obtained rearranging the final line of Equation (4.42). Remembering Equation (4.28), it is also straightforward to show that:

$$\begin{aligned}|\mathcal{LR}_{s_{t+k}}| &= \frac{1}{\alpha^{kN}}(|\mathcal{LR}_{s_t}|-1)+1 = \frac{1}{\alpha^{kN}}(1-|\mathcal{GR}_{s_t}|-1)+1 = \\ &= 1-\frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}|\end{aligned} \tag{4.43}$$

Finally, Equation (4.37) is obtained from Equation (4.36) and Equation (4.28)

$$|\mathcal{GR}_{s_{t+k}}| = 1-|\mathcal{LR}_{s_{t+k}}| = 1-\left(1-\frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}|\right) = \frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}| \tag{4.44}$$

□

In the above analysis it is important to remember that $\frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}| < 1$, otherwise $\mathcal{GR}_{s_t}$ would be larger than $\mathcal{C}(s_t^g, s_t^e)$, which is impossible by definition.

Lemma 2 is of quite general validity, as long as $\mathcal{GR}_{s_t}$ is small respect to $\mathcal{C}(s_t^g, s_t^e)$, and located relatively far from the borders of $\mathcal{C}(s_t^g, s_t^e)$. Even when a portion of $\mathcal{GR}_{s_t}$ is close to the border of $\mathcal{C}(s_t^g, s_t^e)$, neighbourhood shrinking reduces mostly the largest area ($\mathcal{LR}_{s_t}$), and $\mathcal{GR}_{s_{t+1}} \simeq \mathcal{GR}_{s_t}$. Equation (4.37) shows that the relative coverage of $\mathcal{GR}_{s_t}$, and hence the likelihood of sampling a fitter solution than s_t^g, grows ($1/\alpha > 1$) exponentially. Neighbourhood shrinking is therefore a powerful heuristics to foster progress in the the local search procedure. Neighbourhood shrinking introduces also a trade-off between reducing the reach of local search, and hence slowing down the convergence to the local optimum (see eq. 4.12), and making local search progress more likely, thus avoiding several cycles of stalling. The probability of a complete stalling of local search (i.e. site abandonment) can be calculated from Equation (4.37) as follows:

Proposition 6 (Stalling Probability With Neighbourhood Shrinking and Constant $\mathcal{GR}_{s_t}$)**.** *Let s_t^g be the centre of site s at cycle t in the N-dimensional solution space. The probability that local search stagnates for k cycles if $\mathcal{GR}_{s_t}$ is not changed by neighbourhood shrinking is:*

$$P(s_t^g = s_{t+k}^g) = \prod_{h=1}^{k} \left(1 - \frac{1}{\alpha^{hN}} |\mathcal{GR}_{s_t}|\right)^{nr} \tag{4.45}$$

Proof. After h cycles of stalling, the probability $P_{nr=1}(s_h^g = s_{h+1}^g)$ of not sampling a single solution fitter than s_t^g in $\mathcal{C}(s_t^g, s_{t+h}^e)$ is determined by the relative coverage of $\mathcal{LR}_{s_{t+h}}$, which is defined in Equation (4.36):

$$P_{nr=1}(s_h^g = s_{h+1}^g) = |\mathcal{LR}_{s_{t+h}}| = 1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}| \tag{4.46}$$

The probability of stalling at any cycle h is equal to the probability of not picking a fitter solution than s_t^g in nr independent samples of $\mathcal{C}(s_t^g, s_{t+h}^e)$:

$$P(s_h^g = s_{h+1}^g) = P_{nr=1}(s_h^g = s_{h+1}^g)^{nr} \tag{4.47}$$

The probability of k consecutive cycles of stalling is calculated from Equations (4.46) and (4.47):

$$P(s_t^g = s_{t+k}^g) = \prod_{h=1}^{k-1} P(s_h^g = s_{h+1}^g) = \prod_{h=1}^{k-1} \left(1 - \frac{1}{\alpha^{hN}} |\mathcal{GR}_{s_t}|\right)^{nr} \tag{4.48}$$

□

This result is valid as long as the number of recruited bees is constant for the k cycles monitored. If the number of bees changes at every iteration, for example as in Packianather et al. (2009), nr in Equation (4.45) should be replaced by a variable number nr_k.

The stalling probability can never be 0, since $\mathcal{LR}_{s_t} \neq \emptyset$ for any s_t. It should also be noted that the results of Propositions 5 and 6 are independent of the neighbourhood shape. The implications of using hyperspherical instead of hypercubic neighbourhoods will be discussed in Section 4.4.

4.3.4 A large *stlim* or *nr*?

Proposition 6 is important to understand the behaviour of the Bees Algorithm when neighbourhood shrinking does not change $\mathcal{GR}_{s_t}$, or at least does not change it significantly. As discussed in Section 4.3.3, this occurrence is most likely when neighbourhood search is near the local optimum, that is, $\mathcal{GR}_{s_t}$ is small and near the centre of $\mathcal{C}$. In this case, the probability that local search stagnates is large ($|\mathcal{GR}_{s_t}|$ is small), and the site may be abandoned after *stlim* cycles of stalling before the local optimum is found (local search stalls).

The probability that local search stalls depends on the number *nr* of solutions that are sampled in one local search cycle, the stalling limit *stlim*, and the size of the search scope. The larger *nr* and *stlim* are, the more likely is to pick at least one solution within $\mathcal{GR}_{s_t}$, and thus the smaller is the likelihood that local search stalls. However, the effect of *nr* and *stlim* on the stalling probability is not the same, due to the nonlinear reduction of the search scope by neighbourhood shrinking. Given a fixed number of sampling opportunities (equal to $nr \cdot stlim$), the question is whether it is more beneficial to sample thoroughly $\mathcal{C}$ for lesser iterations (large *nr*), or sample less intensely $\mathcal{C}$ for longer times (large *stlim*).

In this section, it is assumed that *nr* and *stlim* can be increased by an integer factor $q > 1$, and the local search stalling probability will be indicated as $P_{nr}(s_t = s_{t+stlim})$, where the index *nr* accounts for the number of candidate solutions sampled in $\mathcal{C}$ in one local search cycle.

Lemma 3. *Let s_t^g be the centre of site s at cycle t in the N-dimensional solution space. Assuming that $\mathcal{GR}_{s_t}$ is not changed by neighbourhood shrinking, an increase in the stalling limit by an integer factor $q > 1$ modifies the stalling probability of local search as follows:*

$$P_{nr}(s_t^g = s_{t+q\cdot stlim}^g) = P_{nr}(s_t^g = s_{t+stlim}^g) \prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{nr} \quad (4.49)$$

Proof. From Proposition 6:

$$\begin{aligned} P_{nr}(s_t^g = s_{t+q\cdot stlim}^g) &= \prod_{k=1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}|\right)^{nr} = \\ &= \prod_{k=1}^{stlim} \left(1 - \frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}|\right)^{nr} \cdot \prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}|\right)^{nr} = \\ &= P_{nr}(s_t = s_{t+stlim}) \prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}}|\mathcal{GR}_{s_t}|\right)^{nr} \end{aligned} \tag{4.50}$$

□

Lemma 4. *Let s_t be the centre of site s at cycle t in the N-dimensional solution space. Assuming that $\mathcal{GR}_{s_t}$ is not changed by neighbourhood shrinking, an increase in the number of foragers by an integer factor $q > 1$ modifies the stalling probability of local search as follows:*

$$P_{q\cdot nr}(s_t^g = s_{t+stlim}^g) = P_{nr}(s_t^g = s_{t+stlim}^g)^q \tag{4.51}$$

Proof. The proof is straightforward:

$$\begin{aligned} P_{q\cdot nr}(s_t^g = s_{t+stlim}^g) &= \prod_{k=1}^{stlim} \left(1 - \beta^k|\mathcal{GR}_{s_t}|\right)^{q\cdot nr} = \\ &= \left(\prod_{k=1}^{stlim} \left(1 - \beta^k|\mathcal{GR}_{s_t}|\right)^{nr}\right)^q = \\ &= P_{nr}(s_t^g = s_{t+stlim}^g)^q \end{aligned} \tag{4.52}$$

□

The next remark will prove that if $\mathcal{GR}_{s_t}$ is not changed by neighbourhood shrinking, increasing the stalling limit of an integer factor $q > 1$ has more effect on decreasing the stalling probability than increasing the number of foragers by the same factor.

Proposition 7 (*stlim* vs. *nr*)**.** *Let s_t be the centre of site s at cycle t in the N-dimensional solution space. Assuming that $\mathcal{GR}_{s_t}$ is not changed by neighbourhood shrinking, an increase in the stalling limit by an integer factor*

$q > 1$ reduces the stalling probability more than an equal increase in the number of foragers.

$$P_{nr}(s_t^g = s_{t+q\cdot stlim}^g) < P_{q\cdot nr}(s_t^g = s_{t+stlim}^g) \tag{4.53}$$

Proof. Remembering Lemmas 3 and 4, Equation (4.53) can be re-written as:

$$P_{nr}(s_t^g = s_{t+stlim}^g) \prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{nr} < P_{nr}(s_t^g = s_{t+stlim}^g)^q \tag{4.54}$$

with $P_{nr}(s_t^g = s_{t+stlim}^g)$ a non-null probability, and hence a positive real number. Equation (4.54) can thus be rewritten as:

$$\prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{nr} < P_{nr}(s_t^g = s_{t+stlim}^g)^{q-1} \tag{4.55}$$

Remembering Proposition 6:

$$P_{nr}(s_t^g = s_{t+stlim}^g)^{q-1} = \left(\prod_{k=1}^{stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{nr}\right)^{q-1} \tag{4.56}$$

Equation (4.55) becomes:

$$\prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{nr} < \prod_{k=1}^{stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{(q-1)\cdot nr} \tag{4.57}$$

The two terms inside the brackets on the right and left hand sides of 4.57 express the relative coverage of $\mathcal{LR}_{s_t}$ at time k. That is, they represent the probability of picking a solution of lower fitness than s_t inside $\mathcal{C}$ at time k. They become smaller as k increases ($\alpha < 1$), and thus:

$$1 - \frac{1}{\alpha^{stlim\cdot N}} |\mathcal{GR}_{s_t}| \leq 1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}| \qquad \forall k \in \{1, stlim\} \tag{4.58}$$

Likewise:

$$1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}| \leq 1 - \frac{1}{\alpha^{stlim\cdot N}} |\mathcal{GR}_{s_t}| \qquad \forall k \in \{stlim + 1, q \cdot stlim\} \tag{4.59}$$

Accordingly:

$$X = \prod_{k=1}^{stlim} \left(1 - \frac{1}{\alpha^{stlim \cdot N}} |\mathcal{GR}_{s_t}|\right)^{(q-1)\cdot nr} < \prod_{k=1}^{stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{(q-1)\cdot nr} = Y \tag{4.60}$$

and

$$W = \prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{nr} < \prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{stlim \cdot N}} |\mathcal{GR}_{s_t}|\right)^{nr} = Z \tag{4.61}$$

Equation (4.57) ($W < Y$) is certainly true if $W < Z \leq X < Y$, that is, if:

$$\prod_{k=stlim+1}^{q\cdot stlim} \left(1 - \frac{1}{\alpha^{stlim \cdot N}} |\mathcal{GR}_{s_t}|\right)^{nr} \leq \prod_{k=1}^{stlim} \left(1 - \frac{1}{\alpha^{stlim \cdot N}} |\mathcal{GR}_{s_t}|\right)^{(q-1)\cdot nr} \tag{4.62}$$

Setting $A = \left(1 - \frac{1}{\alpha^{stlim \cdot N}} |\mathcal{GR}_{s_t}|\right)$, Equation (4.62) can be rewritten as:

$$\prod_{k=stlim+1}^{q\cdot stlim} (A)^{nr} \leq \prod_{k=1}^{stlim} (A)^{(q-1)\cdot nr} \tag{4.63}$$

Developing the sequence of products:

$$(A)^{nr \cdot q \cdot stlim} \leq (A)^{(q-1)\cdot nr \cdot stlim} \tag{4.64}$$

Given that A is a stalling probability, and hence $0 < A \leq 1$, the inequality Equation (4.64) is true because the left hand side is raised to a higher power than the right hand side of the inequality. □

Proposition 7 can also be proven considering a fixed number of available sampling opportunities $T = (q \cdot nr) \cdot stlim = nr \cdot (q \cdot stlim)$ of the search scope. If the choice is to increase the number of foragers, $\mathcal{C}$ will be sampled $q \cdot nr$ times for at most $stlim$ cycles of stalling before being abandoned. If $\mathcal{GR}_{s_t}$ is unchanged by neighbourhood shrinking, all candidate solutions will be sampled inside $\mathcal{LR}_{s_t}, \ldots, \mathcal{LR}_{s_{t+stlim}}$ with a stalling probability $\pi_{nr} \geq A = \left(1 - \frac{1}{\alpha^{stlim \cdot N}} |\mathcal{GR}_{s_t}|\right)$ (eq. 4.58). If instead the choice is to increase the stalling limit, $\mathcal{C}$ will be sampled nr times for $q \cdot stlim$ cycles, and $(q \cdot stlim - stlim) \cdot nr$ of these samples will have a stalling probability $\pi_{stlim} \leq A = \left(1 - \frac{1}{\alpha^{stlim \cdot N}} |\mathcal{GR}_{s_t}|\right)$ (eq. 4.59).

As long as $\mathcal{GR}_{s_t}$ is not disjoint multimodal, Proposition 7 gives the practitioner a useful guideline to parameterize the Bees Algorithm. This case is not uncommon as the scope of the local search has narrowed down on the attraction basin of one peak of performance. If $\mathcal{GR}_{s_t}$ contain several peaks, there is the risk that repeated applications of the neighbourhood shrinking procedure may cut the main peak out of $\mathcal{GR}_{s_t}$. In this latter case, a high nr ensures that many sampling attempts are made before the main peak is lost. Unfortunately, the actual fitness landscape is not known, and trial-and-error is usually needed to address the nr vs. $stlim$ trade-off. However, several empirical studies (Pham and Castellani, 2009; Pham and Castellani, 2014; Pham and Castellani, 2015) obtained the best performances over a large set of varied benchmarks using large $stlim$ values, suggesting a wide applicability of Proposition 7.

4.4 Local Search Scope Shape

Among the numerous variants of the BA, the shape of the search scope is one of the least researched features in the literature. In the standard formulation of the Bees Algorithm (Section 4.1), the search scope $\mathcal{C}(s_t^g, s_t^e)$ of a site s at the cycle t, is defined as a hypercube of side s_t^e centred in s_t^g. A new candidate solution $v \in \mathcal{C}(s_t^g, s_t^e)$ is generated uniformly sampling the hypercube $\mathcal{C}(s_t^g, s_t^e)$. The main limitation of this *hypercubic sampling* is the anisotropic character of the search, which has the shortest extent in the direction of the coordinate axes, and the longest aligned with the diagonals of the $\mathcal{C}(s_t^g, s_t^e)$ hypercube. This anisotropy introduces a bias in the local search.

Moreover, as the dimensionality of the solution space increases, the volume of the $\mathcal{C}(s_t^g, s_t^e)$ hypercube exponentially increases, making the sampling more sparse (*curse of dimensionality*, Bellman, 2015).

4.4.1 Isotropic Local Search

An *isotropic* search scope can be implemented using a *hypersphere* (*ball*) $\mathcal{B}$ centred in s_t^g of radius s_t^r. The reader is referred to (Cook, 1957) and (Muller, 1959) for the algorithmic details of how to achieve a uniformly distributed spherical sampling.

Cubic sampling can be replaced by spherical sampling keeping the rest of

the Bees Algorithm unchanged. Neighbourhood shrinking in this case shrinks the hypersphere radius instead of the hypercube edge.

However, replacing cubic with spherical sampling does change the properties of the search. For instance, the maximum reach of local search (Proposition 2) in one cycle changes from the diagonal of the hypercube $\frac{s_t^e\sqrt{N}}{2}$ to the radius of the hypersphere s_t^r, and does not scale any more with the dimensionality of the search space. Moreover, if cubic sampling is used, the volume of the search scope grows with the number of dimensions:

$$\mathcal{V}(\mathcal{C}(s_t^g, s_t^e)) = (s_t^e)^N \tag{4.65}$$

whilst the volume of the hypersphere initially grows and then decreases with the number of dimensions (Stibor et al., 2006):

$$\mathcal{V}(\mathcal{C}(s_t^g, s_t^r)) = V_N \cdot (s_t^r)^N = \frac{\pi^{N/2}}{\Gamma\left(N/2+1\right)}(s_t^r)^N \tag{4.66}$$

where Γ is gamma function. More precisely, keeping the radius s_t^r fixed, the volume increases for the first N^* dimensions, where

$$N^* = \{N \mid D_{N-1} < s_t^r \leq D_N\} \qquad D_N = \frac{\Gamma\left(N/2+3/2\right)}{\sqrt{\pi}\Gamma\left(N/2+1\right)} \tag{4.67}$$

and sharply decreases afterwards, approaching zero for large N values. As mentioned in Section 4.3, replacing cubic with spherical sampling does not alter the validity of Propositions 5 and 6.

Proposition 8 (Scope Variation Invariance)**.** *Let $\mathcal{C}(s_t^g, s_t^e)$ and $\mathcal{B}(s_t^g, s^r)$ be the local search scope using respectively cubic and spherical sampling, and $|\mathcal{GR}_{s_t}|^C$ and $|\mathcal{GR}_{s_t}|^S$ be the relative coverage of the $\mathcal{GR}_{s_t}$ region using respectively cubic and spherical sampling. If neighbourhood shrinking does not change the $\mathcal{GR}_{s_t}$ region, shrinking the edge/radius of the search scope of a factor α leads to the same change in the respective coverages:*

$$\begin{aligned} \mathcal{C}(s_t^g, \alpha s_t^e) &\Rightarrow \frac{1}{\alpha^N}|\mathcal{GR}_{s_t}|^C \\ \mathcal{B}(s_t^g, \alpha s_t^r) &\Rightarrow \frac{1}{\alpha^N}|\mathcal{GR}_{s_t}|^S \end{aligned} \tag{4.68}$$

Proof. This can be directly proven as follows:

$$\frac{\mathcal{V}(\mathcal{GR}_{s_t})}{\mathcal{V}(\mathcal{C}(s_t^g, \alpha s_t^e))} = \frac{\mathcal{V}(\mathcal{GR}_{s_t})}{(\alpha s_t^e)^N} = \frac{1}{\alpha^N} \cdot \frac{\mathcal{V}(\mathcal{GR}_{s_t})}{(s_t^e)^N} = \frac{1}{\alpha^N} \cdot \frac{\mathcal{V}(\mathcal{GR}_{s_t})}{\mathcal{V}(\mathcal{C}(s_t^g, s_t^e))} \tag{4.69}$$

Parameter	Value
N	4
nr	15
s_t^e	10
s_t^{ttl}	8
s_t^g	$[1, 0, 0, 0]$
$\mathcal{GR}_{s_t}$ center	$[0, 0, 0, 0]$
$\mathcal{GR}_{s_t}$ radius	1

Table 4.1: Parameters setting used in the tests.

$$\frac{\mathcal{V}(\mathcal{GR}_{s_t})}{\mathcal{V}(\mathcal{B}(s_t^g, \alpha s_t^r))} = \frac{\mathcal{V}(\mathcal{GR}_{s_t})}{V_N \cdot (\alpha s_t^r)^N} = \frac{1}{\alpha^N} \cdot \frac{\mathcal{V}(\mathcal{GR}_{s_t})}{V_N \cdot (s_t^r)^N} = \frac{1}{\alpha^N} \cdot \frac{\mathcal{V}(\mathcal{GR}_{s_t})}{\mathcal{V}(\mathcal{B}(s_t^g, s_t^r))} \quad (4.70)$$

□

The consequence of Proposition 8 is that the stagnation probability is computed in the same way (Proposition 6) regardless of the kind of sampling used. However, the different behaviour of the search scope volume in the two cases has important implications for high dimensional spaces.

A possible enhancement of the current algorithm would be to switch the shape of the search scope opportunistically to foster the exploratory (cubic sampling) or exploitative (spherical sampling) goal of local search.

4.4.2 Stalling Probability: Experimental Verification

To verify and visualise the theoretical predictions of Section 4.3, and how the stalling probability varies with the shape of the neighbourhood, the following experimental tests were carried out. The situation where local search has converged inside a large basin of attraction was mimicked. A four-dimensional fitness landscape of hyperspherical shape was considered. The hyperspherical basin had unitary radius, and was centred in the origin of the Cartesian space. In this landscape, local search was performed using a hypecubic neighbourhood of edge $s_t^e = 10$, time to live $s_t^{ttl} = 8$, and initially centred in $s_t^g = [1, 0, 0, 0]$. Fifteen forager bees were used to search the neighbourhood. The parameters of the example are summarised in Table 4.1.

In this case, the $\mathcal{GR}_{s_t} \ll \mathcal{C}(s_t^g, s_t^e)$ region is a hypersphere centred in the origin with unitary radius[4]. As per Proposition 4, s_t^g lies on the (open)

[4]The actual radius of $\mathcal{GR}_{s_t}$ is 1^- since $s_t^g \notin \mathcal{GR}_{s_t}$.

surface of the hypersphere $\mathcal{GR}_{s_t}$. It can be shown that the following variables take the values:

$$\mathcal{V}(\mathcal{C}(s_t^g, s_t^e)) = 10^4 \quad \mathcal{V}(\mathcal{GR}_{s_t}) \approx 4.9348 \quad |\mathcal{GR}_{s_t}| \approx 4.9348 \cdot 10^{-4} \tag{4.71}$$

and, by complementarity:

$$\mathcal{V}(\mathcal{LR}_{s_t}) \approx 9995.0652 \qquad |\mathcal{LR}_{s_t}| \approx 0.9995 \tag{4.72}$$

When no neighbourhood shrinking is used (case 1), the stalling probability is given by Proposition 5:

$$P(s_t^g = s^g_{t+s^{ttl}}) = 0.9995^{15 \cdot 8} \approx 0.9425 \tag{4.73}$$

When neighbourhood shrinking is used (case 2, shrinking factor $\alpha = 0.9$), the $\mathcal{GR}_{s_t}$ region is unchanged, therefore its volume is the same. Local search stalls after $s_t^{ttl} = 8$ consecutive cycles of stagnation. For each of these cycles of stagnation, neighbourhood shrinking is applied. The volumes of the initial and final search scope are:

$$\mathcal{V}(\mathcal{C}(s_t^g, s_t^e)) = 10^4 \quad \mathcal{V}(\mathcal{C}(s^g_{t+s_t^{ttl}}, s^e_{t+s_t^{ttl}})) = 343.3684 \tag{4.74}$$

where $s^e_{t+s_t^{ttl}} \approx 4.3047$. The initial and final relative coverage of the $\mathcal{GR}$ regions are:

$$|\mathcal{GR}_{s_t}| \approx 4.9348 \cdot 10^{-4} \qquad |\mathcal{GR}_{s_{t+s_t^{ttl}}}| \approx 1.4372 \cdot 10^{-2} \tag{4.75}$$

According to Proposition 6 the stalling probability is now equal to:

$$\begin{aligned} P(s_t^g = s^g_{t+s^{ttl}}) &= \prod_{k=1}^{s_t^{ttl}} \left(1 - \frac{1}{\alpha^{kN}} |\mathcal{GR}_{s_t}|\right)^{nr} = \\ &= \prod_{k=1}^{8} \left(1 - \frac{1}{0.9^{k \cdot 4}} (4.9348 \cdot 10^{-4})\right)^{15} \approx 0.67144 \end{aligned} \tag{4.76}$$

When a hyperspherical (isotropic) neighbourhood of radius $s^r = 5$ (equivalent to a cubic sampling with edge $s_t^e = 10$) is used (case 3), the search scope volume $\mathcal{V}(\mathcal{B}(s_t^g, s_t^r))$ and the relative coverage of $\mathcal{GR}_s$ are:

$$\mathcal{V}(\mathcal{B}(s_t^g, s_t^r)) \approx 3084.2514 \quad |\mathcal{GR}_{s_t}| \approx 1.6 \cdot 10^{-3} \quad |\mathcal{LR}_{s_t}| \approx 0.9984 \tag{4.77}$$

	Sampling Type			
	Cubic		Spheric	
	Predicted	Experimental	Predicted	Experimental
Without NS	0.9425	0.9429	0.8252	0.8249
With NS	0.6714	0.67137	0.2725	0.2716

Table 4.2: Predicted stalling probability and experimental frequency of stalling events for the four cases described in Section 4.4.2.

If neighbourhood shrinking is not used, the predicted stalling probability is (Proposition 5):

$$P(s_t^g = s^g_{t+s_t^{ttl}}) = 0.9984^{15 \cdot 8} \approx 0.8252 \tag{4.78}$$

If neighbourhood shrinking is performed (case 4), the volumes of the initial and final (after $s_t^{ttl} = 8$ consecutive cycles of stagnation) search scope are:

$$\mathcal{V}(\mathcal{B}(s_t^g, s_t^r)) \approx 3084.2514 \quad \mathcal{V}(\mathcal{B}(s^g_{t+s_t^{ttl}}, s^r_{t+s_t^{ttl}})) \approx 105.9034 \tag{4.79}$$

and the initial and final $\mathcal{GR}_s$ relative coverages are:

$$|\mathcal{GR}_{s_t}| \approx 1.6 \cdot 10^{-3} \quad |\mathcal{GR}_{s_{t+s^{ttl}}}| \approx 4.6597 \cdot 10^{-2} \tag{4.80}$$

The predicted stalling probability is (Proposition 6):

$$P(s_t = s_{t+s^{ttl}}) = \prod_{k=1}^{8} \left(1 - \frac{1}{0.9^{k \cdot 4}}(1.6 \cdot 10^{-3})\right)^{15} \approx 0.2725 \tag{4.81}$$

The theoretical predictions were numerically tested, performing 10^6 independent optimisation runs for each of the above four cases. The results of the tests are summarised in Table 4.2

The empirical results prove the validity of the theoretical predictions. In particular, it is apparent that neighbourhood shrinking increases the probability of progress in local search, thus reducing the stalling probability. At the same time, the empirical examples show the significance of the consequences associated to the choice of neighbourhood shape. In general, the analysis of this section points out that the standard practice of using cubic

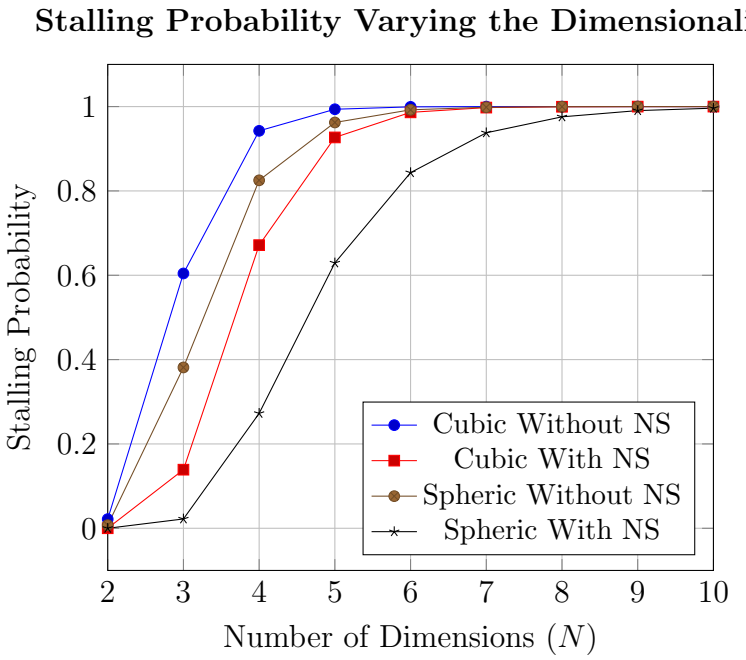


Figure 4.7: Stalling probability using different sampling methods, with and without the neighbourhood shrinking. All the parameters are kept fixed except the number of dimensions of the problem.

sampling should be reevaluated in term of the search bias introduced, and the evolution of the stalling probabilities with repeated iterations of neighbourhood shrinking. In detail, a hypersphere of diameter $2s_t^r$ has a smaller hypervolume than a hypercube of edge $s_t^e = 2s_t^r$, and determines a more exploitative search with higher probability of finding solutions in $\mathcal{GR}_{s_t}$.

The tests for the four cases were also repeated varying the dimensionality of the fitness landscape from 2 to 10. The experimental results are shown in Figure 4.7 and confirm that neighbourhood shrinking and hyperspherical sampling are effective policies against premature stalling.

4.5 Discussion

There is a marked imbalance in the Swarm Intelligence literature, with a prevalence of experimental over theoretical studies. Despite the large success in applications, mathematical analysis of the algorithms is still limited, leav-

ing several open questions on the behaviour, parameterization, complexity, and nature of the various algorithms. Explanation of the metaheuristics is often limited to the biological metaphor, which may prevent the reader from gaining a full understanding of the search mechanisms (Camacho-Villalón et al., 2018). The scarcity of analytical foundations in Swarm Intelligence research has been pointed out by several authors (Yang, 2012; Swan et al., 2015; Piotrowski, 2018).

In this thesis, the properties and main features of the Bees Algorithm were formalised and analysed. Despite a number of experimental studies (Pham and Castellani, 2009; Pham and Castellani, 2014; Pham and Castellani, 2015; Hussein et al., 2017) benchmarked the capabilities of the Bees Algorithm, an analytical investigation of its behaviour and operators had never been carried out. The results of the proposed study clarify and support the previous experimental findings, as well as reveal so far overlooked properties. The main findings are summarised below.

The similarities and differences between the Bees Algorithm and standard optimisation methods were discussed. In particular, the Bees Algorithm can be regarded as a parallel version of the LJ Search and VNS methods, where the sampling of the neighbourhood is adaptively allocated (*waggle dance*) according to the fitness of the seed solution. In terms of local search, the main difference with the two aforementioned methods is in the way the neighbourhood is varied: the Bees Algorithm uses neighbourhood shrinking, whilst VNS tries a number of randomly generated shapes, and standard LJ shrinks the neighbourhood regardless of the progress of local search. Also, the Bees Algorithm terminates the local search after *stlim* stagnation cycles, whilst LJ Search customarily terminates the search after a fixed number of iterations regardless of the progress. In terms of overall metaheuristic, the Bees Algorithm performs several local searches in parallel, adaptively shifting the sampling effort at each generation according to the progress of the search. Neighbourhoods can be abandoned due to lack of progress, or replaced with more promising ones found via global search. For a comparison between the Bees Algorithm and akin swarm optimisation techniques (Kennedy and Eberhart, 1995; Karaboga and Basturk, 2007) the reader is referred to (Pham and Castellani, 2009).

The theoretical analysis of the properties of local search showed that the expected step size quickly approaches the maximum value as the number of forager bees is increased. If local search is desired to quickly climb (descend) the fitness slope, a large neighbourhood size is more beneficial than a large

number of foragers. Analysis of the stalling probability also found limited benefits in increasing the number of local foragers. That is, neighbourhood shrinking and a large stagnation limit are the most effective policies against premature stagnation of local search. This latter result is in good agreement with the indications of several experimental studies (Pham and Castellani, 2009; Pham and Castellani, 2014; Pham and Castellani, 2015), where the best performances were obtained using the largest allowed value for the stagnation limit *stlim*.

One of the main contributions of this theoretical analysis regards the shape of the local neighbourhood. For ease of implementation, nearly all versions of the Bees Algorithm used hypercubic local neighbourhoods. As demonstrated, hypercubic sampling biases the search along the directions of the diagonal, and has poor exploitation capabilities in high dimensional spaces due to the curse of dimensionality. That is, the volume of hypercubic neighbourhoods is a power function of the search scope edge s_e. As suggested in Section 4.4, the neighbourhood shape might be varied during the search to switch from explorative (cubic sampling) to exploitative (spherical sampling) search strategies.

4.6 Conclusions

The Bees Algorithm is a popular optimisation method inspired by the foraging behaviour of honey bees. Despite several experimental investigations, the properties of the Bees Algorithm has never been analysed formally. This chapter covers this gap, focusing particularly on the properties of local search.

The main indications are that the local search capabilities of the Bees Algorithm are mainly determined by the size and shape of the neighbourhood, and the number of allowed stagnation cycles. A large neighbourhood enables a quicker progress on the fitness landscape. Conversely, reducing the neighbourhood size helps avoiding premature stagnation of local search. The effect of increasing forager recruitment on the expected search step size (Section 4.2) and stagnation probability (Section 4.3) grows sublinearly with the number of bees.

The shape of the neighbourhood function has been so far largely overlooked in the Bees Algorithm literature. However, it was shown in Section 4.4 that the customary choice of hypercubic sampling creates large neighbourhoods in high-dimensional spaces due to the curse of dimensionality. On the

other hand, hyperspherical sampling creates neighbourhoods of sizes that vary according to the gamma function, and tend to become small in high-dimensional spaces (zero for infinitely high-dimensional spaces). Thus, the exploitation capability of local search is highly influenced by the choice of neighbourhood shape.

Overall, the Bees Algorithm can be seen as a parallel adaptive version of the LJ Search and VNS algorithms (Section 2.3), in which the modification of the neighbourhood size and allocation of sampling opportunities are dynamically adjusted according to the fitness of the neighbourhood centres and the local progress of the search. Differently from LJ Search and VNS, the Bees Algorithm also keeps on searching the fitness landscape for new promising neighbourhoods via the global search procedure.

Throughout the chapter, the Bees Algorithm was presented in a rigorously mathematical and algorithmic format, beyond the customary qualitative description based on the biological metaphor. It is hoped that this new formalism improves the understanding of the Bees Algorithm, and spurs new analytical studies on its properties, and its similarities to and differences with other Swarm Intelligence metaheuristics.

Chapter 5

Local Optimum Region Radius Estimator for Multi-Shape Primitive Fitting

In this chapter the Local Optimum Region Radius Estimator (LORRE) algorithm is described in detail, and its performance is demonstrated through a number of experimental tests.

The results of the tests presented in Chapter 3 demonstrated the ability of the Bees Algorithm to solve the primitive PF problem in PC scenes including one shape only. However, many vision problems entail the recognition of multiple instances of shapes in the same scene. As discussed in Section 5.2, this case entails the location of multiple local optima corresponding to the various shapes present in the scene. Furthermore, as shown later in the chapter, a number of additional local optima are created by the combination of groups of shapes. These additional optima do not correspond to a valid solution, but create large spurious basins of attraction.

LORRE was created to prevent the BA from falling multiple times in the same basin of attraction. That is, to force the algorithm to explore new areas of the search space. The algorithm was inspired by the results of the analysis of the BA dynamics presented in Chapter 4.

The approach used in LORRE is to couple the BA search with real-time modification of the fitness topology, in order to discourage further searches in already explored local optima regions. When local search is terminated at a site (i.e. the site abandonment procedure is triggered), LORRE memorises the local optimum and estimates the radius of a convex spherical region

around it. This region could correspond to the entire basin of attraction of the local optimum, or its largest spherical sub-region in case of a convex or oblongated basin. The estimation of the radius is done using information from the history of local search steps that brought to the optimum. Once the radius is estimated, the region is 'filled' to prevent new searches from falling again inside. At the end of the search, the set of memorised local optima is retrieved, pruned of the weakest solutions (poor local optima or solutions found due to premature convergence), and returned. If only one optimum is needed, only the best solution is returned.

The multi-shape problem is discussed in Section 5.2. In Section 5.3 the key components and steps of LORRE are illustrated. Section 5.4 discusses possible strategies for solution pruning. A brief discussion on the algorithm behaviour and parameterisation is provided in Section 5.5, whilst Sections 5.6 and 5.7 report the results of two sets of experimental tests. In the first set of experiments (Section 5.6), the algorithm is evaluated against benchmark functions selected for their specific topological properties. In the second set (Section 5.7), the algorithm is applied to an example of multi-shape PF problem.

5.1 Introduction

In contrast with *single-solution* optimisation problems, where the location of the global optimum (or a solution of comparable fitness) is sought, *multi-solution* problems require the location of a set of n best (i.e. fittest) local optima. In the first case, the search algorithm is required to converge consistently to the global optimum. If randomly re-initialised, it should converge again to the same solution, that is, to the optimum. In the second case, the algorithm is required to converge to different local optima. If re-started, it should converge to a previously not visited local optimum.

Many of the most popular optimisation metaheuristics are designed for single-solution problems. Particularly in presence of one (a few) large attraction basin(s) in the search space, they tend to converge to the same (few) solutions. This consistency is certainly a hindrance when looking for multiple solutions, and may even be detrimental to single-solution problems if the global fitness optimum has a narrow basin of attraction.

The BA (Pham and Castellani, 2009) is a biologically inspired search metaheuristics originally designed for single-solution optimisation problems.

The BA performs in parallel independent locally bounded searches on the fitness landscape. Every time one of these local searches converges to a peak, the site abandonment procedure re-starts it to a new random solution. These features increase the exploration capability of the BA, and make it a good candidate for multi-solution optimisation problems. Indeed, Pham et al. (2007) used the BA for finding multiple solutions in a preliminary design optimisation problem. However, the standard Bees Algorithm still lacks a mechanism to avoid falling repeatedly into large basins of attraction.

In this work, a novel version of the BA was created to address the multi-solution problem, and in particular to find as many different fitness optima as possible. The proposed algorithm dynamically modifies the search space once local search is exhausted at a site, 'plugging up' the basin of attraction of the found optimum with a *filler*, in case of minimisation problems, or 'levelling' the local peak, in case of maximisation problems.

The two key aspects of LORRE are the *smoothness* of the region covered by the filler, which can be tuned by a parameter, and the *dynamic* estimation of the *radius* of the filled region. The first aspect is achieved by adding *derating functions* to the original cost function. The second aspect is obtained by analysing the geometric distribution of the worst solutions found in the various iterations of the local search.

If the basin of attraction is not spherical, LORRE will plug up only a sub-region of it. However, for reasonably smooth functions, this sub-region will include the solutions of highest fitness, and the remaining region of the basin of attraction won't be attractive anymore to scout bees. In extreme cases (e.g. a narrow valley), it may take more than one derating function (i.e. more than one local searches) to fill the basin of attraction.

The idea of using a derating function to modify the search space and fill up visited basins of attraction is taken from the SN technique (Beasley et al., 1993). The main shortcoming of SN is that the radius of the derating function is fixed for all basins of attraction and needs to be tuned manually by the user. In the proposed algorithm the radius of the derating function is automatically calculated from the distribution of the worst solutions, and individually fitted to the estimated radius of each visited basin of attraction.

The idea of avoiding already explored sub-regions has been already considered in the BA literature in the context of single-solution problems. Shatnawi et al. (2013) modified the basic BA adding local and global memories of the visited regions. Whilst the local memory is used by the individual bees to avoid landing on the same solution twice, the global memory is used

by the local search to avoid already explored regions. Another BA variant (Imanguliyev, 2013) uses *shift* operators to identify planar sub-regions in the search space, and includes them in a tabu list. The algorithm identifies a planar sub-region when all the solutions sampled inside have a fitness score within a fixed range. Unfortunately, Imanguliyev defined the shift operators only for one-dimensional problems, and it is not clear how to apply the algorithm to most of practical problems which are multi-dimensional.

To address multi-solution optimisation problems, the proposed algorithm uses information gathered from foragers that landed on solutions of low fitness. As it will be shown in the following sections of this chapter, these solutions provide useful information about the search space topology. This information is discarded in the common BA practice, where in a local search step all the solutions but the fittest are removed. To the best of the author's knowledge, the hybrid BA developed by Abdullah and Alzaqebah (2013) is the only instance of BA where information from suboptimal solutions is used, in that case to increase the exploratory ability of the algorithm.

5.2 Multiple Shapes in the Scene

As shown in Chapter 3 the Bees Algorithm is able to recognise primitive shapes in a PC with accuracy comparable to the state-of-the-art. The tests were carried out in scenes containing only one primitive shape (single-solution problem). However, many practical applications involve the presence of multiple shapes in the same scene, of which the parameters need to be estimated. In this case the BA, being a single-solution algorithm, is ill-suited to solve a multi-solution problem. Pham et al. (2007) approached the multi-solution problem keeping all the local optima found by the BA (i.e. all the centres of the abandoned sites) as potential solutions. This approach will be discussed more in detail in Section 5.7 and used as a baseline to benchmark the proposed LORRE algorithm. For the sake of simplicity, in this study only scenes containing multiple shapes of the same type will be considered. Shapes of different types would give basins of attraction of different form, but wouldn't change the nature of the multiple shape problem.

Before presenting the LORRE, it is worthwhile to check if the fitness function used for the single-shape problems in Chapter 3 is suitable to the multi-shape case. Preliminary tests indicated that this was not the case.

To visualise the problem, a PC containing 2 spheres of unitary radius,

respectively centred in $(-2, 0, 0)$ and $(2, 0, 0)$ was used (Figure 5.1a). A certain number of co-planar candidate solutions (primitive spheres) of fixed radius were generated in a grid, keeping the z-component of the centre fixed at 0 and varying in steps the x and y coordinates on the $z = 0$ plane. The fitness of each candidate solution was assessed using Equation (3.1). Figure 5.1 shows how the fitness $\mathscr{F}$ topology varies in two cases: a population of spheres of radius $r = 1$, and a population of spheres of radius of $r = 2$.

In Figure 5.1b, the maximum fitness score (best fitting primitives) is correctly located near the two centres $[-2, 0]$ and $[2, 0]$. However, in Figure 5.1c, where solutions of $r = 2$ were tested, the global optimum is close to the origin $[0, 0]$. Although in the case shown in Figure 5.1c no solution is correct (the candidate solutions have radius $r = 2$ whilst the two primitives have radius $r = 1$), the best solutions have a higher fitness score (0.8) than the two perfectly matching solutions (of fitness 0.56) in Figure 5.1b.

This problem is probably due to the fact that the fitness function $\mathscr{F}$ is invariant to the candidate shape size. This was not an issue related to the single-shape problems because the normalising δ_{max} term in Equation (3.1) depends on the size of the PC, and can be considered an approximation of the optimal shape size. In multi-shape problems, each single shape covers only a fraction of the size of the whole PC. Consequently, the δ_{max} term fails to effectively normalise the fitness score of a candidate primitive. That is, large candidate primitives (as in Figure 5.1c) yield a higher fitness score than smaller primitives as in Figure 5.1b. To remediate this problem, a new fitness function $\mathscr{F}_{MS}$ that includes the size of the shape was devised:

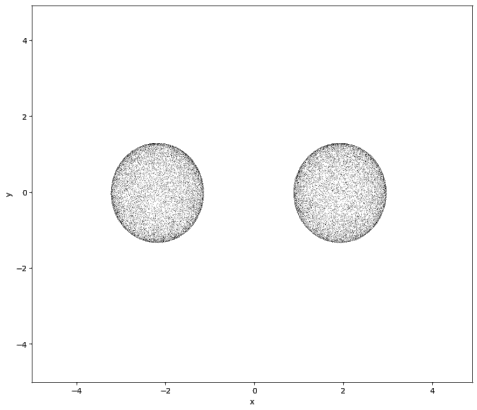
$$\mathscr{F}_{MS}(I, \mathcal{PC}) = \sum_{i=1}^{N} NC(p_i, I) \left(1 - \min\left(1, \frac{|\delta(p_i, I)|}{\lambda}\right)\right) \tag{5.1}$$

where λ is a shape-specific reduction factor defined as follows:

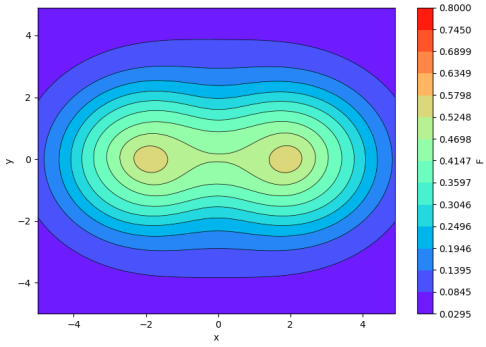
$$\lambda = \begin{cases} r & \text{sphere} \\ \sqrt{h^2 + w^2 + d^2} & \text{box} \\ \sqrt{h^2 + r^2} & \text{cylinder} \end{cases} \tag{5.2}$$

where r, h, w and d are the radius, height, width and depth, respectively.

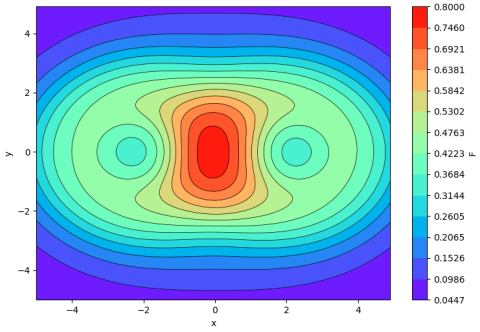
As shown in Figure 5.2, the new fitness function correctly identifies the centres of the two spheres as the regions of maximum fitness in both Figure 5.2a and Figure 5.2b. It is thus suitable to the multi-shape problem.



(a) Point Cloud containing two co-planar identical spheres, viewed from a direction perpendicular to the X-Y plane

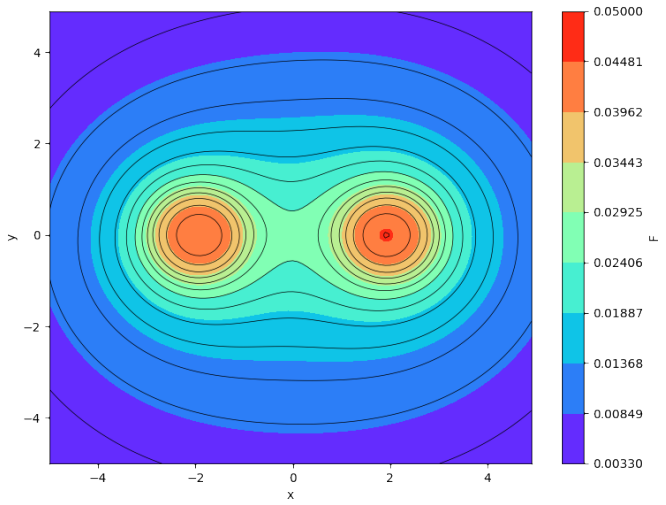


(b) $\mathscr{F}$ Landscape with $r = 1$

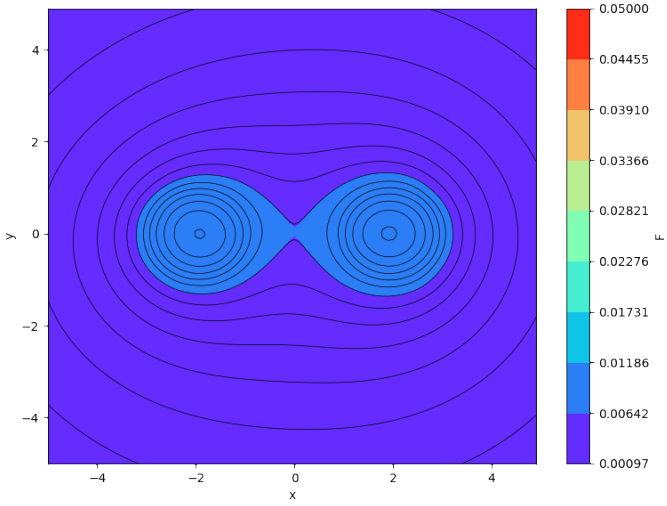


(c) $\mathscr{F}$ Landscape with $r = 2$

Figure 5.1: Point cloud containing two co-planar identical spheres of $r = 1$ and centre of $(-2, 0, 0)$ and $(2, 0, 0)$, respectively. The solutions are tested with the fitness function $\mathscr{F}$ described in Equation (3.1). The fitness is mapped where a candidate solution is centred.



(a) $\mathscr{F}_{MS}$ Landscape with $r = 1$



(b) $\mathscr{F}_{MS}$ Landscape with $r = 2$

Figure 5.2: Point cloud containing 2 identical spheres of $r = 1$ and centred at $(-2, 0, 0)$ and $(2, 0, 0)$, respectively. Candidate solutions are tested with the fitness function $\mathscr{F}_{MS}$ described in Equation (5.1). The fitness is mapped where a candidate solution is centred.

5.3 Description

The LORRE algorithm can be broken down into a number of basic parts. The *exploratory* part aims to find the global optimum of a modified version of the fitness function (Equation (5.8) in Algorithm 5.1). For this part the standard version of the BA is used, with the only differences that a) every cycle of the algorithm, the worst solution sampled during local search at each site is memorised, and b) the solutions are evaluated on a dynamically modified fitness function $\mathscr{M}$ instead of the original fitness function $\mathscr{F}$ (Equation (5.8)).

The *fitness modification* part is executed when the abandonment procedure is triggered at a site. When this happens, the best found solution is stored in the *solution set*, and the fitness function changed to avoid searching again in the same region. The radius of the region is estimated analysing the set of worst solutions sampled at each step of the local search at the site (Section 5.3.1).

Finally the *pruning part* takes place when the main loop of the algorithm is terminated (i.e. the stopping criterion has been met). In the pruning part, the solution set is pruned of the least fit elements (Section 5.4), and the remaining solutions are returned as the set of local optima found by the algorithm.

The power of the LORRE algorithm comes from its ability to estimate the extent of the heights in the fitness landscape. That is, given a found optimum o, LORRE finds the largest sub-region of the solution space where o is included, and where the $\mathscr{F}$ landscape is convex in case of minimisation problems or concave for maximisation problems. This region may coincide with the entire basin of attraction of o, or a subregion of it where (some of) the solutions of highest fitness lie. Without loss of generality, unless otherwise indicated, maximisation problems will be considered in this chapter.

Initially, the algorithm operates exactly as the standard BA, where independent local searches are performed in parallel at multiple sites in the solution space. When at a step t the search at a site has stagnated for *stlim* consecutive cycles, it is assumed that local optimum has been found. The centre o and radius o^r of a region around this optimum is estimated as described in Section 5.3.1. The optimum o is added to the set of found local optima O and the fitness function is modified using a derating function that lowers (or increases, in case of minimisation problems) the score of the sub-region around o. The modified fitness function is then used by all sites generated after step t. This process is repeated until the stopping criterion

for the BA is met. When the algorithm is terminated, the set O contains the list of all potential local optima of the fitness surface. This list will be pruned (Section 5.4) of duplicate and weak solutions.

By dynamically estimating the radius of regions surrounding the local optima, and discouraging further search in these regions via derating functions, LORRE effectively addresses the multi-solution problem. Moreover, since each local optimum radius is estimated individually, this method can handle the presence of heterogeneous attraction basins (i.e. attraction basins of different shapes and radius) on the fitness surface.

5.3.1 Local Optimum Region Centre and Radius Estimation

In the BA the local optimum at a site is approached through a series of local search steps (see Equation (4.1)). At each step, nr foragers sample the search space in a neighbourhood centred on the best-so-far, and if one of the forages lands on a better solution than the current best, it becomes the new scout (i.e. the new centre of the neighbourhood). The remaining $nr - 1$ scouts are discarded, together with the information they gathered on the neighbourhood. It should be noted that the evaluation of the fitness function $\mathscr{F}$ is usually the single most expensive operation, in computational terms, in an optimisation technique (Castellani, 2018). Therefore, retrieving and making use also of the information from the discarded foragers would greatly improve the efficiency of the algorithm.

The key assumption of the LORRE is that at a late stage in a site lifetime, the centre is likely to be close to the local optimum o. Once the optimum is approximately located, local search may still yield a few iterations of minimal progress (where the optimum is approximated with increasing precision), and then stalls, shrinking the neighbourhood for *stlim* iterations until the site abandonment procedure is triggered.

As discussed in Chapter 4, the solution s_n marked by the scout in the final *stlim* iterations is the best approximation of a local optimum $s_n \approx o$, and thus will be used as the centre of the region that is excluded from future searches (henceforth called the *tabu region*). The fitness landscape in the tabu region is assumed to be convex and isotropic (symmetric) about the optimum (e.g. a hypersphere or a paraboloid of revolution). The radius of the tabu region needs then to be estimated.

When the basin of attraction is not symmetric around the optimum, LORRE is designed to define the tabu region in its largest convex and symmetric sub-region. This is done using the specific radius selection criteria described in Section 5.3.1.

As local search progresses at a site, and the centre is located with increasing precision, the shrinking procedure progressively reduces the edge ngh of the neighbourhood. At a certain step k, the local search neighbourhood will approximate the largest convex region around the optimum. That is, it approximates the region that will be set as tabu when the local search at the site is exhausted. Starting from step k, there will be a direct relationship between the proximity of a solution to the centre of the neighbourhood, and its fitness. Consequently, the *worst* found solutions w_t at each subsequent step $t = k, \ldots, n$ will be near the border of the neighbourhood, and *far* from o. Figure 5.3 shows this property for a one-dimensional fitness maximisation problem.

Given a site s, the radius $r \approx o^r$ of the tabu region can be approximated as follows:

1. collect all the worst solutions $S^w = \{w_1, \ldots, w_n\}$ sampled in all the n local search steps performed in s.

2. select a border solution $w^* \in S^w$ according a given criterion;

3. compute the radius approximation r according the position of the border solution w^*;

Figure 5.4 shows in a minimisation problem how the contour of a local optimum region can be inferred from the set of worst solutions S^w (in red), and how this region can be plugged up using a derating function to discourage further local searches. Some criteria for selecting w^* are proposed in the next subsection.

Radius Selection Criteria

The simplest criterion to select the border solution $w^* \in S^w$ would be to take the solution w_m of *median* distance to o. This idea exploits the fact that many of the solutions $w_t \in S^w$ often lie at the boundary of the tabu region (see Figure 5.3), and thus the element of median distance to the centre is likely to be close to this border. With this approach the radius approximation r

can be computed as the distance between w^* and the best solution found in site s, namely:

$$r = d(w^*, s_n) \tag{5.3}$$

Although simple, this criterion was empirically proven inaccurate when the tabu region is small, relatively to the neighbourhood reduction stepsize (defined by the *stlim* and α parameters). In this case the solution w^* tends to be located outside the tabu region border. Conversely, when the tabu region is large, relatively to the neighbourhood reduction stepsize, the solution w^* tends to be located inside the tabu region border. Both cases result in a poor estimation of the radius r.

An alternative selection criterion that better exploits the topological information gathered by the site can be defined using the set of best solutions S and the set of worst solutions S^w found.

This criterion looks for the topologically closest best solution s^* to s_n such as there exist at least one worst solution w^* topologically closer to s_n with worst fitness score than s^*. If such solution s^* exists, this is likely situated beyond the tabu region border. Both s^* and w^* can be formally defined as follows:

$$s^* = \arg\min_{s \in S} \{d(s, s_n) | \hat{W}_{s,s_n} \neq \emptyset\} \tag{5.4}$$

where $\hat{W}_{s,s_n}$ is the subset of worst solutions S^w defined as follows:

$$w \in \hat{W}_{s,s_n} \Leftrightarrow w \in S^w \wedge d(w, s_n) < d(s, s_n) \wedge \mathscr{F}(w) < \mathscr{F}(s) \tag{5.5}$$

If such s^* exists, the radius can be estimated as:

$$r = d(w^*, s_n) \qquad w^* = \arg\min_{w \in \hat{W}} \{\mathscr{F}(w)\} \tag{5.6}$$

If such s^* doesn't exists, all the S solutions are likely being sampled inside the same tabu region, so the radius can be estimated as:

$$r = d(s_1, s_n) \tag{5.7}$$

5.3.2 The LORRE Algorithm Step by Step

The LORRE algorithm is based on the BA, therefore it is possible to describe its steps as a modification of the BA described in Algorithm 4.1. The

most notable differences are the inclusion of a *level* variable s^l associated to every site, that is $s = \{s^g, s^e, s^{ttl}, s^l\}$, the adaptive fitness function $\mathscr{M}$ that is initialised as $\mathscr{F}$ and is progressively modified by the addition of *derating functions*, and the estimation of the local optimum centre and radius, computed every time a site stalls and stored in an ordered set $\mathscr{O}$.

Each time a new scout is generated, it is evaluated on the latest modified fitness function. If the scout finds a site which is chosen for local search, this fitness function will be used to evaluate all the candidate solutions found within the site by the foragers. In other words, the fitness function used to evaluate the scout becomes associated to the site, and will be used until the site is abandoned. The purpose of the level variable s^l is to identify which version of the modified fitness is used at the site. In practical terms, s^l is a counter that keeps track of how many sites had been abandoned at the time local search was started at the site, and thus records how many derating functions are used in the modified fitness function $\mathscr{M}$. These modifications to the BA are summarised in Algorithm 5.1.

The action of the derating functions on the fitness landscape is shown in Figure 5.5 for one sample run of the LORRE algorithm.

The G function present in Equation (5.8) in Algorithm, 5.1 is a *derating function*, which locally diminish the fitness score. Some examples of derating functions are:

- *proportional function*:

$$G_p(s^g, o) = \begin{cases} \left(\frac{d(s^g,o)}{o^r}\right)^\alpha & d(s^g, o) < o^r \\ 1 & d(s^g, o) \geq o^r \end{cases} \tag{5.9}$$

  which is $G_p(s^g, o) = 0$ when $s^g = o$ and increases progressively with the distance $d(s^g, o)$ until $d(s^g, o) = o^r$.

- *exponential function*:

$$G_e(s^g, o) = \begin{cases} e^{\ln m \cdot \frac{o^r - d(s^g,o)}{o^r}} & d(s^g, o) < o^r \\ 1 & d(s^g, o) \geq o^r \end{cases} \tag{5.10}$$

  where $\alpha > 0$ is a parameter, and $m > 0$ is the derating minimal value that the function G_e takes when $d(s, o) = 0$ (i.e. $G_e(o, o) = m$).

The above are standard derating functions used in the literature (Beasley et al., 1993). Alternatively to the derating functions, a *flat* derating factor can be used:

$$\mathscr{M}_{flat}(s) = \begin{cases} o^l & d(s,o) < o^r \\ \mathscr{F}(s) & d(s,o) \geq o^r \end{cases} \quad o = \arg\min_{o \in L_{opt}} \{d(s,o)\} \tag{5.11}$$

This derating factor can't be used with techniques such as SN, since they don't use the topological information of the fitness landscape. The main differences between using Equation (5.11) instead of Equation (5.8) are that the penalising factor (i.e. $o^l = \mathscr{F}(w^*)$) is:

- not pre-set but derived from the local topology of $\mathscr{F}$;
- it covers the tabu region with a flat surface instead of a 'hole' (maximisation problem) or a 'bump' (minimisation problem);

The latter point is particularly important because it avoids the creation of artificial basins of attraction such as valleys created by neighbouring derating functions (see Figure 5.4b). For this reason, the modified fitness function of Equation (5.11) will be used henceforth.

When the search part of the algorithm is terminated, a post processing part is performed to select from the set O the best optima. This part is described in the following section.

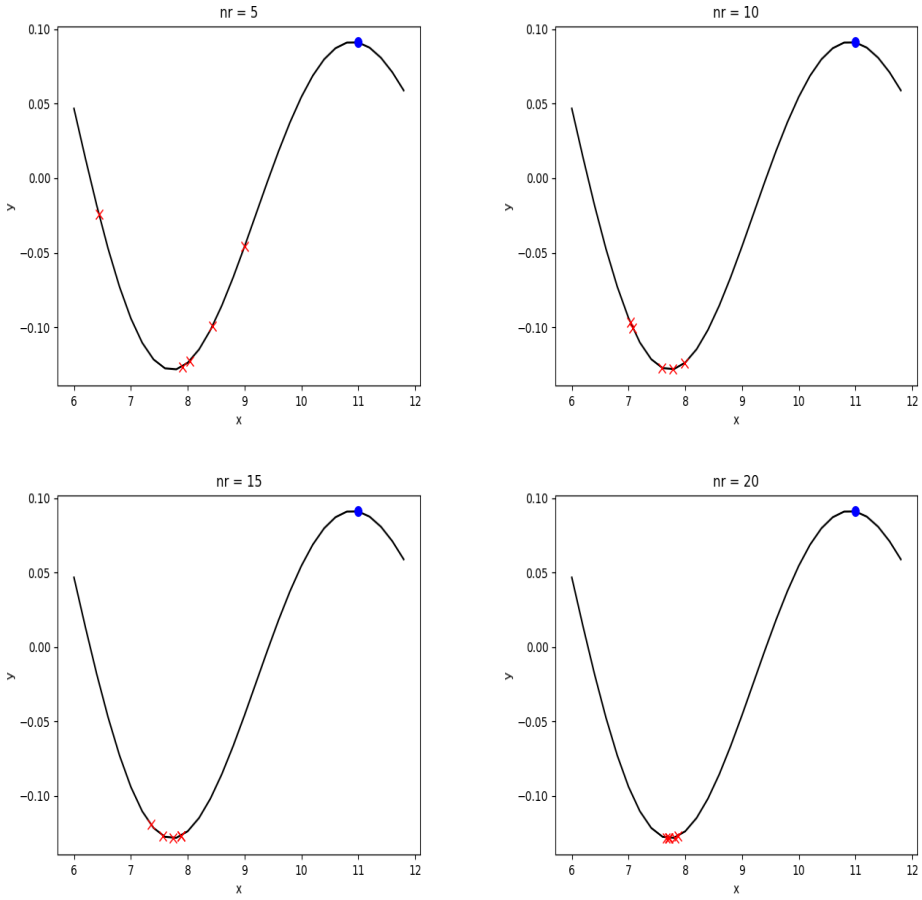
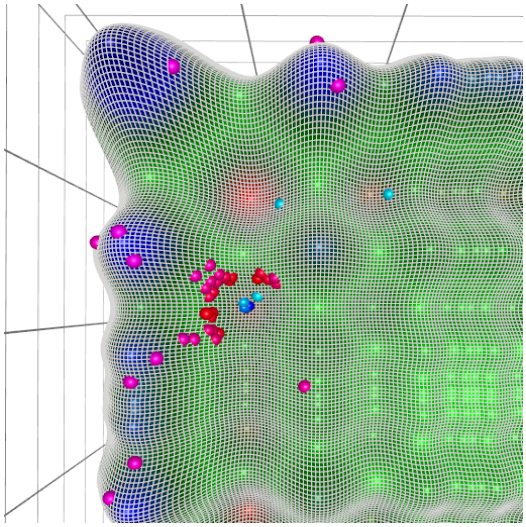
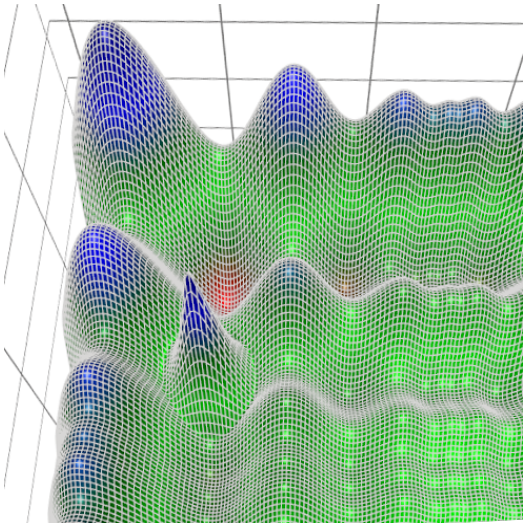


Figure 5.3: The example shows and instance of BA local search on the one-dimensional fitness function $\mathscr{F}(s^g) = -^{\sin(s^g)}/_{s^g}$ in the interval $[6, 12]$. The neighbourhood is centred in s_n (in blue) and the search is performed using $stlim = 5$ and different nr. The worst solutions (in red) generated at each step of the last $stlim$ steps lie around the local optimum region boundary.



(a) Before



(b) After

Figure 5.4: Minimisation problem on the multimodal fitness landscape of the Schwefel function (Laguna and Marti, 2005). The tabu region is delimited using the best (blue) and worst (red) solutions found during the local search (a), and a derating function is applied to 'plug' the hole and discourage further searches in the region (b).

LORRE Algorithm: Main Steps

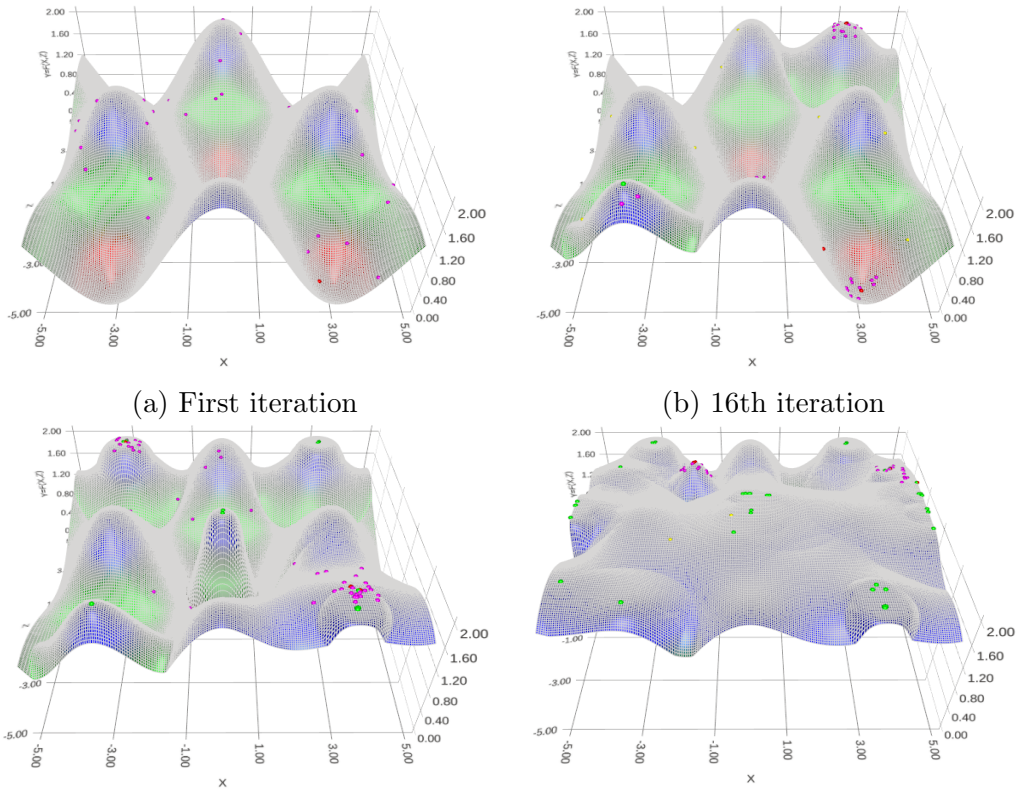
1. (**Initialisation**) Every initial solution is generated with a 0 level, namely $s^l = 0$, and the local optima set is initialised as $\mathscr{O} = \emptyset$;
2. (**Modified Fitness**) Every solution s is evaluated by the following modified fitness function $\mathscr{M}$:

$$\mathscr{M}(s^g) = \mathscr{F}(s^g) \prod_{i=0}^{s^l} G(s^g, o_i) \tag{5.8}$$

   where o_i is the i^{th} found optimum in $\mathscr{O}$ and $G(s^g, o_i)$ is a derating function.

3. (**Level Adjustment**) Every time a new candidate solution s is generated, its level s^l is set as follows:
   - if s is a scout bee used in the global search, then $s^l = |\mathscr{O}|$;
   - if s is a forager bee of the site v, then $s^l = v^l$;
4. (**Optimum Estimation**) When a site stalls:
   (a) its best local solution s_n becomes a new optimum o with radius o^r found as following the steps suggested in Section 5.3.1 and derating level $o^l = \mathscr{M}(w^*)$;
   (b) the new local optimum is added to the solutions set $\mathscr{O} = \mathscr{O} \cup o$;
5. (**Search Termination**) Terminate the search process when a stop criteria is met;
6. (**Solutions Pruning**) Prune the local optima set $\mathscr{O}$ from all the solutions that are unlikely to be distinct local optima of $\mathscr{F}$, following the steps explained in Section 5.4;
7. (**Algorithm Termination**) The algorithm terminates returning the pruned local optima set as a list of local optima of $\mathscr{F}$;

Algorithm 5.1: The LORRE Algorithm



(a) First iteration

(b) 16th iteration

(c) 70th iteration

(d) 200th iteration

Figure 5.5: Modified search space on the Griewank (Molga and Smutnicki, 2005) function (minimisation problem). Proportional derating functions with dynamically estimated radius are progressively added every time a local minimum is found.

5.4 Strategies for Solutions Pruning

Once the exploratory part of the LORRE algorithm is completed, the set of candidate optima $\mathscr{O}$ will contain several solutions. These solutions may be:

(i) valid optima, that are a good approximation of the local optima of $\mathscr{F}$;

(ii) duplicate solutions, or other solutions from the same basin of attraction but of lower fitness;

(iii) spurious optima, solutions away from optima in $\mathscr{F}$, either due to premature convergence of the local search at a site, or artifacts introduced by the application of the derating functions in $\mathscr{M}$;

Pruning the solution set from duplicate and spurious optima is not a trivial task, and is a common problem in multi-solution optimisation. The most reliable way to select the best optima is to use domain knowledge to weed out spurious, duplicate, and low-fitness solutions. When this knowledge is not present, some heuristics can be used to perform the selection.

Spurious optima (iii) are mainly created in the last iterations of the algorithm, when most or all the optima in $\mathscr{F}$ have already been found and the landscape created by $\mathscr{M}$ is mainly determined by the derating functions. A fitness-based ranking procedure is therefore able to eliminate most of this kind of optima.

Duplicate optima (ii) are possible since at any time several (at most nb) sites use the same modified fitness function. This happens at the start of the LORRE algorithm when no site has yet stalled, and the original fitness function $\mathscr{F}$ is used at all sites. After the fitness function has been modified, any two or more sites initialised at the same step (at most all nb) will share the same fitness function. In this situation, it is possible that some or all the sites sharing the same fitness function converge at the same optimum. These duplicate solutions can be detected by their similarity.

Since often only optima of very high fitness are sought, another possible pruning strategy is to rank all the solutions $o_i \in \mathscr{O}$ by their fitness score, and use a finite difference operator D to assess where the fitness presents a sharp decrease. The D operator approximates the first derivative for the fitness score in the discrete space of the ranked solutions, and can be defined as follows:

$$D(t) = \mathscr{F}(o_i) - \mathscr{F}(o_{o-1}) \qquad i > 1 \tag{5.12}$$

In conclusion, since the problem of selecting the local optima is specific to the topology of $\mathscr{F}$ and the task, an optimal pruning criterion doesn't exist. Instead, different pruning filters can be implemented, and some examples based on the above remarks are summarised as follows:

domain-guided the motivation behind the use of a search metaheuristics is that often very little is known about the possible solutions. However, depending on the domain, some properties of the desired solutions may be known, and these properties can be used to guide the pruning process. For instance, in certain cases a minimum fitness score is given for acceptable solutions. This information can be used to rule out suboptimal solutions from $\mathscr{O}$ (*absolute cut-off*);

percentile cut-off the solutions in $\mathscr{O}$ can be sorted according to their score, and only solutions within a certain percentile are retained.

derivative cut-off the above absolute and percentile cut-off criteria rely on domain knowledge, which is not always available. Using the function defined in Equation (5.12) it is possible to establish a cut-off point at a given solution $o_t \in \mathscr{O}$ based on the $D(t)$ signal. Sorting the elements of $\mathscr{O}$ in descending order, $D(t)$ is always non-positive. A cut-off point can then be defined where the absolute difference $D(t) - D(t+1) > c$ or the relative difference ${}^{D(t+1)}/_{D(t)} > c$.

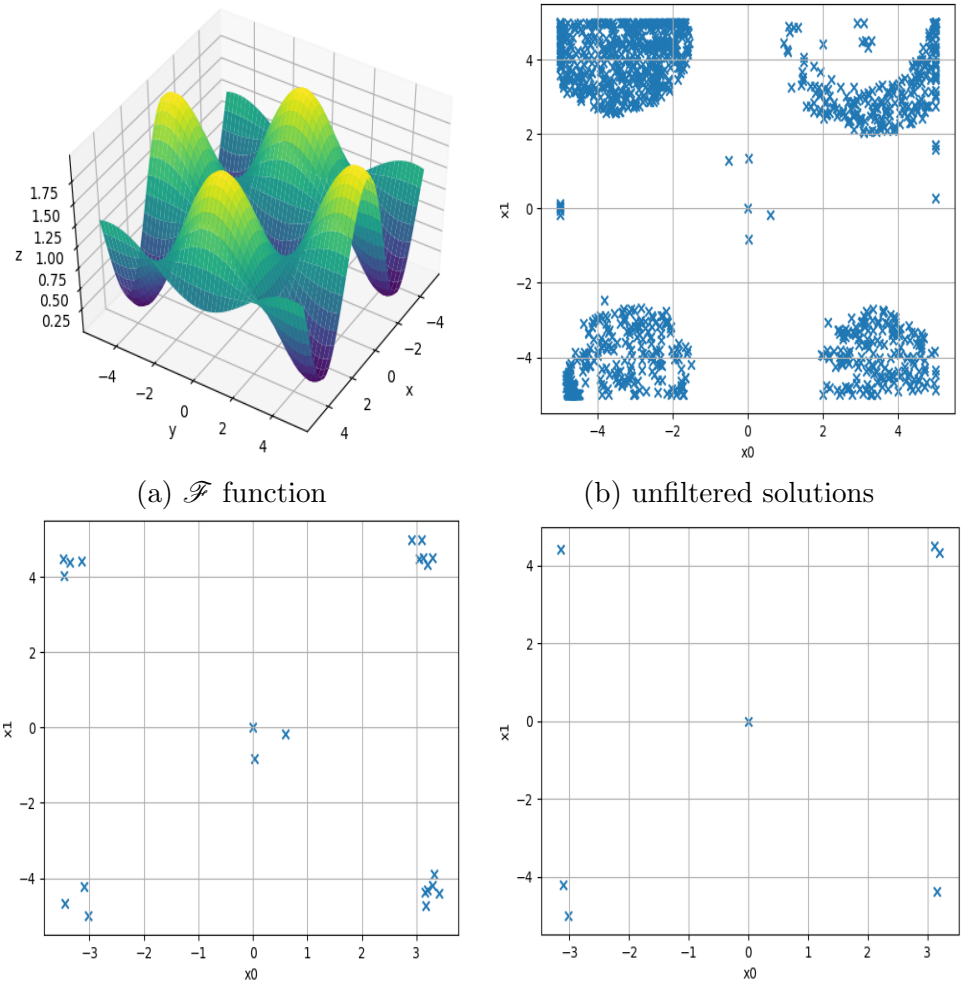
similarity-based two or more sites may find the same local optimum. Duplicate solutions can be detected by their similarity, and the time when the local search was initialised. Only the fittest element of two or more duplicate solutions is kept.

level-based sites generated at the same time use the same modified fitness function (Section 5.3.2). The level variable s^l identifies which modified version of the fitness function is used at a site. Thus, two topologically close solutions (having overlapping derating functions) found by sites using the same fitness function (same level), are likely to come from the same basin of attraction.

Figure 5.6 shows an example of pruning techniques applied on the solutions found by a run of the LORRE algorithm used to find the minima of the

Griewank function in $[-5, 5]$. In this example, the first sites generated derating functions of peak values of about[1] -1 with a radius that covers most of the local optima region. However, the points outside those regions have a lower score than -1 so the next solutions still tend to stall in the same regions becoming duplicated optima and, as it's possible to see in the top-rightmost part of Figure 5.6b, deceptive optima. These optima are removed applying different filters. A simple absolute cutoff (at -0.3) is able to remove most of the unwanted solutions (see Figure 5.6c) and a further level-based filter removes the duplicates almost completely (see Figure 5.6d).

[1] since this is a minimisation problem, the values of the Griewank function have reversed sign in $\mathscr{F}$.



(a) $\mathscr{F}$ function

(b) unfiltered solutions

(c) absolute cutoff

(d) absolute cutoff and level-based

Figure 5.6: Example of solutions found on the Griewank function (minimisation problem) and the effect of different solutions pruning filters applied

5.5 Some Considerations

When a site is abandoned by BA at iteration t, a derating function is applied to $\mathscr{F}$ at the location of the optimum o, to create a modified fitness function $\mathscr{M}$. This new function $\mathscr{M}$ will be used by all the scout bees (and any forager they may recruit) generated from iteration $t+1$ onwards, until a new site abandonment triggers another modification to $\mathscr{M}$.

Given that exhausted sites are abandoned asynchronously, to be replaced by new sites using a modified fitness function, it can be said that LORRE performs in parallel different local searches on different fitness landscapes (unless search in two or more sites is initialised at the same time). This introduces the problem of comparing the fitness of the scouts (i.e. the fittest-so-far solutions) of different sites, since the fitness evaluation criterion is not the same for all sites. This problem exists when the solutions created using global search are compared to the solutions of local search, to select the next nb sites for local search. Consequently, global search may be best avoided in the LORRE procedure, with exception for:

- The initialisation (step 1 of the algorithm), where only scout bees are used. In this case, all the scouts evaluated on the same fitness function $\mathscr{F}$;

- The BA iteration after a site is abandoned, nrb scout bees may be sent out to perform global search, and the best solution found becomes the centre of a new local search site . These scout bees are evaluated on the same fitness function (i.e. $\mathscr{M}$ with level $l = |\mathscr{O}|$ maximum) and are not compared with other existing sites;

For the same reason, partitioning the best nb sites into elite and best, with a different allocation of foragers for the two groups, should be avoided in the LORRE procedure.

5.6 Tests on Benchmark Functions

To assess the performance of the LORRE algorithm, 3 tests cases have been selected, and the performance of the proposed algorithm is evaluated against a standard BA using Sequential Niching (SN) to prevent further searches near already discovered optima. Since SN and LORRE use the BA for search,

the differences found in the results will be determined by the way the two algorithms modify the fitness landscape.

Selecting the correct derating radius in the SN algorithm is of crucial importance. This task is often very difficult, since the radius of the optimum region (the equivalent of the tabu region for LORRE) is only guessed without using any information from the search. To simplify the task, in this study the dimensionality (number of independent variables) of the benchmark functions is reduced to two.

Two popular benchmark functions were selected from the literature for the tests because of their topological properties, together with a third custom-made function. These benchmark functions have been designed as minimisation problems, therefore the algorithms will be set in finding the global minimum. For consistency reasons, the third custom-made function is also designed to model a minimisation problem. The first benchmark is the Griewank function (Molga and Smutnicki, 2005), considered for $-5 \leq x_i \leq 5$ and defined as follows:

$$Griewank_{lowered}(x, y) = 1 + \frac{1}{4000}(x^2 + y^2) - \cos\left(\frac{x \cdot y}{\sqrt{2}}\right) - 2 \qquad (5.13)$$

This definition, here called *lowered Griewank*, differs from the original one only for the -2 term, a modification made necessary only for technical reasons. The implementations for the SN and LORRE algorithms used in these tests always maximise $\mathscr{F}$, so in order to minimise a target function $g(x)$, the fitness function is set as the opposite of the target function $\mathscr{F}(x) = -g(x)$. The values of the Griewank function, as it is defined in literature, are always positive. As a result the values of the fitness function $\mathscr{F}$ used to minimise the Griewank function are always negative. This create problems with the derating functions Equations (5.9) and (5.10) defined in the SN, which work as intended only when the fitness function is positive. Adding the -2 term in the Griewank definition makes $\mathscr{F}$ always positive, solving this issue. Notably, the derating procedure described in Equation (5.11), unique to the LORRE algorithm, doesn't suffer of the same problem, working as intended even when $\mathscr{F}$ is negative.

The Griewank function presents clearly discernible optima of similar depth situated in symmetric positions. All the attraction basins of the optima are large and have the same radius. This function therefore presents the ideal conditions for the application of the SN algorithm which uses derating functions of fixed pre-set radius.

Another function that presents multiple symmetric optima was made specifically for this study and, considered for $-20 \leq x_i \leq 20$, is defined as follows:

$$H\&H(x, y) = -\sum_{i=1}^{n} \mathcal{N}_{PDF}(x, y, \mu_i, C_i) * v_i \tag{5.14}$$

where $\mathcal{N}_{PDF}$ is the probability density function of the multivariate normal distribution of average μ_i and covariance matrix

$$C_i = \begin{pmatrix} c_i & 0 \\ 0 & c_i \end{pmatrix}$$

The last coefficient v_i is used to lower/rise the local minima. The actual values of the coefficients are defined in Table 5.1. This function will be

i	μ_i	c_i	v_i
1	$[0, 0]$	10.5	2.0
2	$[20, 0]$	14.0	2.5
3	$[0, 20]$	16.0	2.7
4	$[-20, 0]$	12.0	2.5
5	$[0, -20]$	9.0	2.3
6	$[10, 10]$	0.1	0.05
7	$[-10, -10]$	0.2	0.3
8	$[-10, 10]$	0.25	0.24
9	$[10, -10]$	0.17	0.23

Table 5.1: Coefficients values for the Hills & Holes 5.14 function.

henceforth called *Hills & Holes*. Differently from the Griewank function, Hills & Holes presents basins of attraction of very different radius. Furthermore, the basins of deepest holes are the narrowest. This function is expected to be more difficult for the SN algorithm, since a an optimal radius of fixed size doesn't exists for the derating functions. In the experiments, two SN algorithms using derating functions of different radii are evaluated.

The last benchmark is the 2D Michalewicz (Molga and Smutnicki, 2005) function, considered for $0 \leq x_i \leq \pi$, which is defined as follows:

$$Michalewicz(x, y) = -\sin(x)\sin^{20}\left(\frac{x^2}{\pi}\right) - \sin(y)\sin^{20}\left(2\frac{y^2}{\pi}\right) \tag{5.15}$$

The Michalewicz function is characterised by two relatively close main minima[2] at the crossing of valleys and surrounded by plateaux. This function should be the most difficult for SN amongst those considered, since it is very easy to exclude one or both the main minima with a derating function misplaced in one of the valleys. The three functions are shown in Figure 5.7.

The domain where the three benchmarks are defined, together with the parameters used for the algorithms are given in Table 5.2. The algorithms were parameterised by trial and error. For two benchmarks, the radius of the derating functions for the SN algorithm was optimised to fit as closely as possible the basin of attraction of the main (Michalewicz) or all (Griewank) the local minima. This setting is the most favourable for SN. In the case of Hills & Holes function, there is no one radius that fits all basins of attraction, so the algorithm was tested in two parameterisations: one where the derating functions use the radius that fits the narrowest basins, and one where it fits the widest basins.

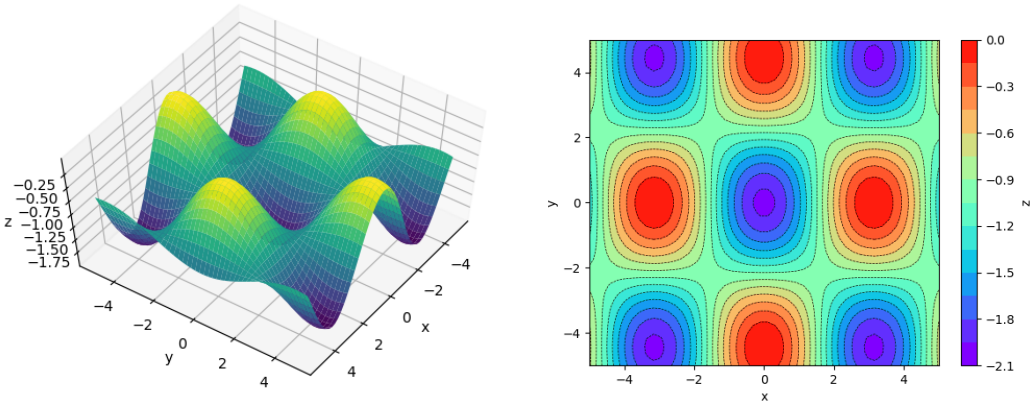
In all cases, the parameters ns, ne and nre in the BA used by the SN and LORRE are set to 0 (Section 5.5). SN changes the fitness landscape (Section 2.4) every 15 iterations of the BA. Given the simplicity of the benchmarks, this interval is adequate to locate with good accuracy one of the minima.

The total number of derating functions added (i.e. stops and restarts with a modified fitness function) is equal to the number of optima (no) in the benchmark function. This setting too creates the optimal conditions for SN to find all the optima in the fitness landscape, without generating any further spurious solutions once all the basins of attraction have been filled by the derating functions. The LORRE algorithm terminates after $15 \cdot no$ iterations, performing thus the same number of fitness evaluations in one run as SN (the two algorithms are parameterised to perform the same number of fitness evaluations per iteration).

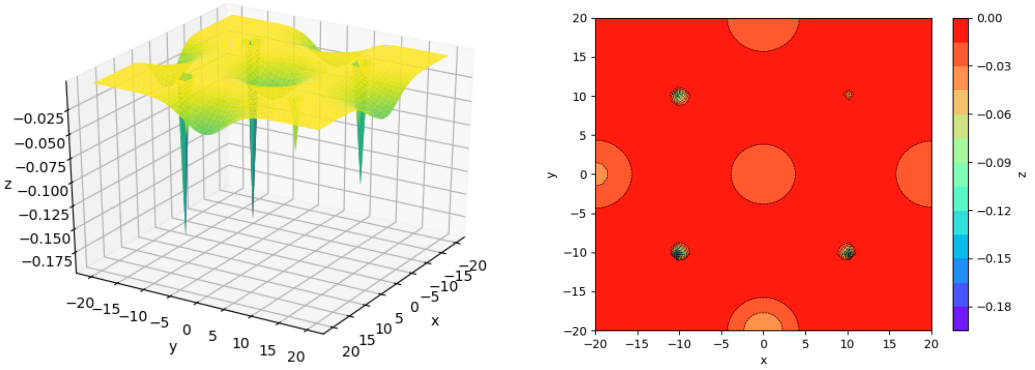
5.6.1 Evaluation Criteria

The LORRE and the SN algorithms were tested on the three becnhmark functions described in the previous section. Their performance is evaluated as follows:

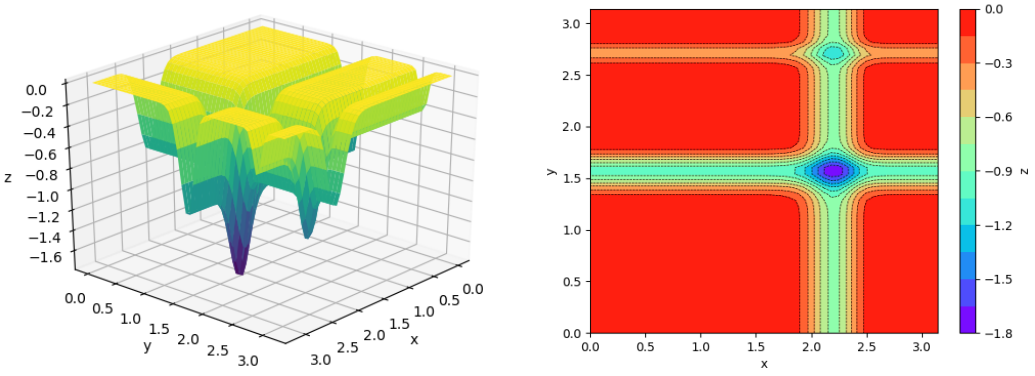
[2]In the experiments the *strict* variant of optimum o definition is used: $\exists \epsilon > 0 \mid \mathscr{F}(x) < \mathscr{F}(o) \forall x \in \mathcal{B}(o, \epsilon)$



(a) (lowered) Griewank



(b) Hills and Holes



(c) Michalewicz

Figure 5.7: Functions used in the experiments.

Function	Algorithm	*nb*	*nrb*	*stlim*	Iterations	Derating Radius
Griewank	SN	5	8	5	60	2
	LORRE	5	8	5	60	-
Hills & Holes	SN	1	4	3	120	$[2, 5.5]$
	LORRE	1	4	3	120	-
Michalewicz	SN	5	10	6	30	0.25
	LORRE	5	10	6	30	-

Table 5.2: Algorithm parameters used for the different functions.

1. Each algorithm (LORRE or SN) is run until the pre-set maximum number of iterations is reached;
2. Two different instances of SN using different derating radius are evaluated.
3. All the found solutions $\mathscr{O}$ are extracted and, in case of the LORRE, filtered using absolute cut-off and level-based pruning filters;
4. For each test, an error value is generated as follows:

$$Error(\mathscr{O}) = \frac{1}{no} \sum_{i=1}^{no} \min_{o \in \mathscr{O}} \{d(o, ro_i)\} \tag{5.16}$$

with $ro_1, \ldots, ro_{no}$ optima of the function $\mathscr{F}$.

Equation (5.16) takes every peak in the fitness landscape (ro_i), and finds its distance from the closest found solution (o). In the ideal case that the algorithm under evaluation has located all the local optima with good accuracy, there will be a closely matching solution for each optimum ($d \approx 0$, and the total error value will be small or zero. If one or more fitness optima were missed or poorly approximated (e.g. local search stalled prematurely), the distance of the closest solution from the optimum will be significant, and the error value large. Thus, the measure in Equation (5.16) captures the ability of an algorithm to locate all optima as well as its precision in locating them.

5.6.2 Experimental Results

For each function, both algorithms were run 100 times, and the five number summary of the error values is presented in Table 5.3. The table shows that

with exception of the Griewank function, LORRE always outperforms SN. In the case of the Griewank function, it should be remembered that the experiment is strongly biased in favour of SN, since the algorithm is given the exact radius of the minima whilst LORRE must find it. Despite being at disadvantage, the results obtained by LORRE are very close to those obtained using SN. The differences in performance are particularly noticeable in the Hills & Holes function, where there is no one radius that fits all holes.

Function	Algorithm	min	1^{th} q.	median	3^{th} q.	max
Griewank	SN	3.157×10^{-2}	6.560×10^{-2}	8.417×10^{-2}	9.740×10^{-2}	1.400×10^{-1}
	LORRE	1.582×10^{-2}	8.456×10^{-2}	1.231×10^{-1}	1.664×10^{-1}	2.684×10^{-1}
H&H ($r = 2$)	SN	1.994	3.547	4.734	5.397	8.017
	LORRE	1.387	2.426	2.955	3.481	5.242
H&H ($r = 5.5$)	SN	3.489×10^{-1}	3.171	3.552	4.617	6.112
	LORRE	1.387	2.426	2.955	3.481	5.242
Michalewicz	SN	4.438×10^{-3}	5.704×10^{-1}	5.748×10^{-1}	5.825×10^{-1}	6.128×10^{-1}
	LORRE	1.873×10^{-3}	3.059×10^{-2}	5.502×10^{-2}	5.732×10^{-1}	6.917×10^{-1}

Table 5.3: Five Values Summary of the error for the different test cases.

In conclusion, the above results show that even using the most favourable settings for SN, LORRE is highly competitive on multi-solution problems.

5.7 LORRE Applied to Multi-Shape Problem

The results of the tests reported in Section 3.3 suggest that the BA is a good technique to tackle the single-shape PF problem, whilst the results in Section 5.6 suggest that the LORRE algorithm is a good technique to tackle multi-solutions problems. In order to assess the most innovative aspect of the LORRE algorithm, namely its ability to exclude already explored regions, it is here tested against the regular BA. Since the base search mechanism is identical, if the tests will show any difference, this difference will be due to the effect of the derating functions applied to the local optima regions in the LORRE algorithm. In the case of the BA all the found solutions will be evaluated, in contrast with the LORRE algorithm, where the found solutions are pruned before the evaluation step. For each type of shape, 20 different instances of PC were created, each instance containing multiple instances of the shape were performed. Each individual instances featured a PC including

8 equidistant shapes placed on a symmetrical grid at points (x, y, z) where $x, y, z \in \{0, 3\}$. Each shape in a PC is composed of 100 points, has a random size of extremes as in Table 5.4 and, when applicable, a random rotation.

Size Parameter	Sphere	Box	Cylinder
radius	$(0.3, 1.0)$	-	$(0.3, 1.0)$
height	-	$(0.3, 1.0)$	$(0.3, 1.0)$
width	-	$(0.3, 1.0)$	-
depth	-	$(0.3, 1.0)$	-

Table 5.4: Parameter ranges for the random shapes generation.

The BA and LORRE algorithms were differently parameterised for each shape type. The parameterisation of the two algorithms is given in Table 5.5, and has been optimised via trial and error. Using the parameters in Table 5.5, the two algorithms exactly sample the same number of solutions in the solution space.

The range of the solution space was set to $[-5, 5]$ for each centre coordinate, and $[0.1, 10]$ for the size parameters for both the algorithms. On each PC, the BA and LORRE algorithms were run for 200 iterations.

The final solution $s_n = o$ found at each stalled site, as well as all the current centre of the sites that had not yet stalled when the algorithm was terminated, were included in a set $\mathscr{O}$. The error of an algorithm, with a solutions set $\mathscr{O}$, on a PC composed by the set of 8 primitives P is computed using the formula in Equation (5.16) ($no = 8$):

$$E(S, P) = \frac{1}{8} \sum_{p \in P} \min_{o \in \mathscr{O}} \{d(p, o)\} \tag{5.17}$$

Shape	Algorithm	ns	nb	ne	nrb	nre	$stlim$
Sphere	BA	0	2	1	4	8	5
	LORRE	-	4	-	3	-	2
Box	BA	0	5	1	10	20	5
	LORRE	-	5	-	12	-	4
Cylinder	BA	0	5	1	10	12	5
	LORRE	-	4	-	13	-	3

Table 5.5: Algorithms Parameters for the different shape types.

where $d(p, o)$ is the distance between each of the 8 primitives p and a solution o. This distance is computed as the Euclidean distance of the vectors composed by the shape centre and size parameters. This error metric has been chosen for its simplicity.

For each test, 10 independent runs were performed for each algorithm, and the five number summary of the results is shown for each type of shape in Table 5.6 and fig. 5.8.

The Mann-Withey test was used to assess the significance of the differences between the results obtained by the two algorithms. For each shape, the Mann-Withey test compared the error results attained on the 200 independent tests performed (10 independent runs for each of the 20 test PC). The p-values obtained in the Mann-Whitney tests are smaller than $4 \cdot 10^{-55}$, indicating that the results on each shape are significantly different.

Shape	Algorithm	min	q1	median	q3	max
Sphere	BA	2.927	3.366	3.429	3.472	3.607
	LORRE	1.092	1.574	2.015	2.415	3.533
Box	BA	4.617	5.909	6.611	7.236	8.985
	LORRE	2.443	3.704	4.329	5.002	6.462
Cylinder	BA	3.412	4.707	5.407	6.154	7.918
	LORRE	2.035	2.925	3.426	4.057	5.546

Table 5.6: Error values of the multi-shape tests of the BA and LORRE algorithms as five values summary.

5.7.1 Discussion

It is useful to notice that the terms of the comparison are biased towards the BA for the following reasons. First, once the local search stalls at (near) a fitness peak, LORRE is prevented from returning to the local optimum by a derating function. Contrarily, in the the BA other instances of local search (i.e. scouts) may re-discover the same peak and further improve the accuracy with which the peak is located. Second, the error function defined in Equation (5.17) takes into account, for each shape, only the *best* approximation amongst all the solutions found by the BA. In the case of LORRE, Equation (5.17) uses only the solutions that have been selected after the pruning process. If one of the peaks hasn't been found with high accuracy, it may

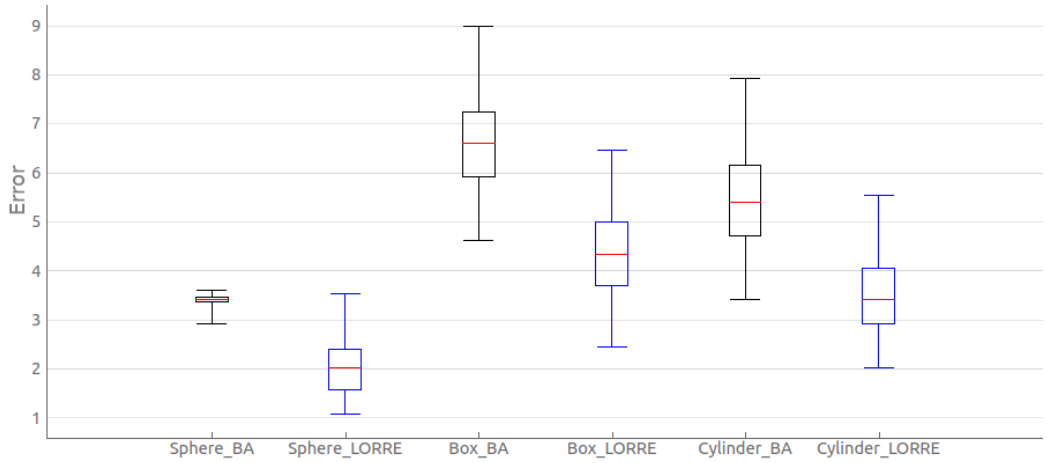


Figure 5.8: Graphical representation of the multi-shape test results for spheres, boxes and cylinders. The LORRE results are in blue.

get filtered out from the set $\mathscr{O}$. In summary, the risk mistakenly removing a poorly approximated peak compounds the problem that LORRE has only one attempt to locate with precision a peak. Removing the pruning step to evaluate the two algorithms on a more equal footing would not be correct, since pruning is an integral part of multi-solution problems.

Despite the biases discussed above, the experimental results reported in Table 5.6 show that LORRE attains a smaller error than the standard BA in the identification of all shapes.

5.8 Conclusion

In this chapter a new algorithm for the multi-solution problem has been presented. Topological information on the fitness landscape given by the worst solutions produced by local search is discarded in standard BA practice. Conversely, LORRE exploits this information to create tabu regions that exclude already explored regions from future local search attempts. LORRE stands out from similar state-of-the-art multi-solution techniques for its ability to dynamically estimate the radius of already visited regions in the search space as well as the capability of using different radii for different regions.

In this chapter several pruning criteria for the found solutions were also provided. It is worth noticing that the pruning problem is often overlooked

in the discussion of similar multi-solution techniques like the EA variant with fitness sharing and the SN.

The performance of LORRE was assessed on different benchmark functions as well as on the real-world problem of multi-shape recognition PF. In the latter case, the LORRE performances compared to the BA showed how the heuristics introduced are beneficial in solving a practical multi-solution problem. In the former case, the algorithm outperforms the SN, a state-of-the-art technique in the multi-solution problems. These results are particularly interesting, considering the tests on the benchmark functions with the derating radius, in the SN, is set to optimal values for the different functions and the tests on the multi-shape problem are similarly biased in favour of the BA. The promising results of these preliminary tests suggest that LORRE is suited to tackle multi-solution problems, and can be a valuable tool in many practical applications.

Chapter 6

Conclusions

In this work, PF has been approached as a numerical optimisation problem. Two different solutions have been proposed, respectively variants of EA and BA, and the results achieved suggested the BA as the most fit solution for the problem, even in presence of noisy models. This approach yields the competitive advantage of not requiring ad-hoc assumptions on the object scanned.

The work done on the PF problem motivated a deep analysis of the BA, with a focus on the statistical properties of the sequence of local searches performed by a site. The study started from a mathematical formalisation of the BA, and performed an *a-posteriori* analysis on a site lifetime, from its inclusion into the nb best sites, to its abandonment. The stalling probability was evaluated in different cases, with and without the neighbourhood shrinking heuristics. The effect of different parameterisations of the BA was statistically analysed. The results of the analysis yielded useful information on trade-offs such as the stagnation limit $stlim$ versus the number of foragers nr at a site. The results of the analysis confirmed experimental findings from the literature, and provide useful guidelines for the parameterisation of the Bees Algorithm. Finally, for the first time the effect of the choice of neighbourhood shape on the algorithm performance has been discussed, and the results are particularly relevant for high dimensional problems.

Similarities and differences between BA variants and similar algorithms (e.g. LJ Search and VNS) were discussed where appropriate. This work of analysis of the BA is particularly significant for the Swarm Intelligence field, where theoretical studies are usually scarce and limited in scope, and the literature is overwhelmingly composed of empirical studies.

The study of the BA local search drew the attention on the possibility of inferring the local topology of the fitness landscape from the results of the sampling within one neighbourhood. This led to a novel approach to discourage further searches in already visited regions of the solution space. This approach was implemented in the new LORRE algorithm, which was conceived with the aim of solving multi-solution optimisation problems. The motivation for the LORRE algorithm was to find multiple primitives in one PC. The LORRE algorithm was proven competitive with the well-known SN algorithm on a number of benchmark functions. Tested on multi-shape models, LORRE confirmed its competitiveness and accuracy. The results of the tests presented in this thesis indicate that the LORRE algorithm is a promising innovative approach for finding multiple local optima in multi-modal functions.

6.1 Future Work

This thesis included a thorough statistical analysis of the BA, provided a general approach to the PF problem, and devised an innovative technique for the multi-solution optimisation problem. Perhaps most importantly, this work provided the ground for interesting future research directions.

Chapter 3 presented a new method to fit geometric primitives to PC models. Three different types of primitives were considered: spheres, boxes and cylinders. However, the proposed approach is general enough to be used on other primitive shapes. Furthermore, the technique can be used to fit non-primitive shapes, provided a point-to-surface distance and normal concordance can be defined (see Equation (3.1)). Further work should extend the application of the proposed approach to other kinds of primitives. Combinations of primitives could be used to create more complex shapes.

The experimental tests proved the robustness of the BA to deal with sampling error, which simulates sensor imprecision. Further work should investigate the ability of the proposed algorithm to fit partially occluded shapes. This occurrence is common in many real-world applications, particularly in cluttered and unstructured (e.g. outdoor scenes) environments.

The BA analysis in Chapter 4 shed light on some dynamics of the algorithm. This however is not an exhaustive study. The dynamics of multiple local searches, and their relations with the global search carried out by the random scouts, have been only mentioned in this work. These dynamics

deserve a deeper analysis.

Section 4.4 analysed for the first time the implications of the choice of neighbourhood shape for the local search, especially in the context of exploitative search in high dimensional problems. This preliminary study should be deepened and expanded, and the results could be utilised to optimise the exploitative search of the algorithm.

The experiments performed in Chapter 5 suggest that the LORRE algorithm is a valid approach for multi-solution optimisation problems. One of the main advantages of LORRE respect to similar multi-solution techniques, is that it doesn't employ a pre-fixed derating radius. At present, its main limitation is that it works optimally on isotropic basins of attraction; that is, on attraction basins that are symmetric about the optimum. A more sophisticated approach would be to use anisotropic derating functions, for example hyper-ellipsoids instead of hyper-spheres. The same considerations made in Section 5.3.1 to draw the borders of isotropic regions, can be made to delimit more complex shapes in the fitness surface.

The preliminary experiments on the multi-shape problem described in Section 5.7 showed promising results. However, the complexity of this problem demands a deeper validation of the fitness function proposed in Equation (5.1) on more complex cases, such as partly occluded shapes, heavy background noise, etc. The presence of shapes of different type in the same scene must also be considered. Finally, the evaluation of the proposed method on real scenes is needed to provide the final validation for use in real-world applications.

6.2 Contributions of this Thesis

The work undertaken by the author in the PhD program and detailed in this thesis brought the following scientific contributions:

- it provided two numerical optimisation solutions to the PF problem, one based on an EA and another based on the BA. Experimental evaluation of the two methods was carried out;
- it formalised for the first time the BA in mathematical terms, as a necessary step towards a formal analysis of the algorithm;
- it performed a statistical analysis of key parts of the BA, with a focus

on the dynamics of the sequence of consecutive local searches at a site, including:

  - the analysis of the progress of local search at a site, including the determination of the a) expected reach and b) upper bound on the speed of convergence to a local optimum;
  - the analysis of the stalling probability at a site, and its impact on the accuracy of the local optimum approximation;
  - the analysis of the trade-off in the allocation of sampling resources between stagnation limit (when search at a site is deemed exhausted)) and number of foragers per BA cycle.
  - the analysis of the implications of the choice of shape for the neighbourhood in local search, and the potential impact of this choice on the local search progress in high dimensional optimisation problems;

- designed and implemented an innovative algorithm, called LORRE, suited to finding the best local optima in multimodal functions;
- tested the LORRE algorithm on a selected number of optimisation benchmarks, and successfully applied it to the multi-shape variant of the PF problem;

6.3 Publications Arising from this Thesis

The work undertaken in the PhD program and described in this thesis resulted in the following publications:

- Pham, D.T., Baronti, L., Zhang, B. and Castellani, M., 2018. "Optimisation of Engineering Systems With the Bees Algorithm." International Journal of Artificial Life Research (IJALR), 8(1), pp.1-15.
- Baronti L., Alston M., Mavrakis N., Ghalamzan A. M. and Castellani M. "Primitive Shape Fitting in Point Clouds Using the Bees Algorithm." *Under second revision for* Applied Sciences.
- Baronti L., Castellani, M. and Pham, D.T. "An Analysis of the Search Mechanisms of the Bees Algorithm" *Under first revision for* Swarm and Evolutionary Computation (SWEVO).

Part of this work has also been presented in the following conferences:

- Baronti L. and Castellani, M. 2018 "Application of LORRE, a Novel Algorithm for Multi-Modal Optimisation to Shape Recognition for Robotic Manipulation in Disassembly Operations." III International Workshop on Autonomous Remanufacturing (IWAR)
- Castellani, M. and Baronti L. 2018, Primitive fitting based on the Bees Algorithm for robotic disassembly in remanufacturing. First International Workshop on Application of the Bees Algorithm in sustainable development (IWABA 2018), Hanoi, Vietnam

Bibliography

Abdullah, S. and M. Alzaqebah (2013). "A hybrid self-adaptive bees algorithm for examination timetabling problems". In: *Applied Soft Computing* 13.8, pp. 3608–3620.

Ahmad, S. (2012). "A study of search neighbourhood in the bees algorithm". PhD thesis. Cardiff University.

Alston, M. (2019). "Evolutionary Algorithm vs. RANSAC for primitive shape fitting." MA thesis. Birmingham, UK: University of Birmingham.

Auger, A. and B. Doerr (2011). *Theory of randomized search heuristics: Foundations and recent developments*. Vol. 1. World Scientific.

Ayala, F. J. and J. A. Kiger (1984). *Modern genetics*. Menlo Park Benjamin and Cummings.

Ballard, D. H. (1981). "Generalizing the Hough transform to detect arbitrary shapes". In: *Pattern recognition* 13.2, pp. 111–122.

Beasley, D., D. R. Bull, and R. R. Martin (1993). "A sequential niche technique for multimodal function optimization". In: *Evolutionary computation* 1.2, pp. 101–125.

Bellman, R. E. (2015). *Adaptive control processes: a guided tour*. Princeton university press.

Bjorkman, M., Y. Bekiroglu, V. Hogman, and D. Kragic (2013). "Enhancing visual perception of shape through tactile glances". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 3180–3186.

Bolles, R. C. and M. A. Fischler (1981). "A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data." In: *IJCAI*. Vol. 1981, pp. 637–643.

Boulch, A. and R. Marlet (2012). "Fast and robust normal estimation for point clouds with sharp features". In: *Computer graphics forum*. Vol. 31. 5. Wiley Online Library, pp. 1765–1774.

Bräysy, O. (2001). *Local search and variable neighborhood search algorithms for the vehicle routing problem with time windows*. Vaasan yliopisto.

Camacho-Villalón, C. L., M. Dorigo, and T. Stützle (2018). "Why the Intelligent Water Drops Cannot Be Considered as a Novel Algorithm". In: *International Conference on Swarm Intelligence*. Springer, pp. 302–314.

Castellani, M. (2018). "Competitive co-evolution of multi-layer perceptron classifiers". In: *Soft Computing* 22.10, pp. 3417–3432.

Castellani, M., Q. T. Pham, and D. T. Pham (2012). "Dynamic optimisation by a modified bees algorithm". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 226.7, pp. 956–971.

Chum, O. and J. Matas (2008). "Optimal Randomized RANSAC". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.8, pp. 1472–1482. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.70787.

Cook, J. (1957). "Rational formulae for the production of a spherically symmetric probability distribution". In: *Mathematics of Computation* 11.58, pp. 81–82.

Dalitz, C., T. Schramke, and M. Jeltsch (2017). "Iterative hough transform for line detection in 3D point clouds". In: *Image Processing On Line* 7, pp. 184–196.

De Jong, K. A. (1975). "An analysis of the behavior of a class of genetic adaptive systems." PhD thesis.

Djenić, A., N. Radojičić, M. Marić, and M. Mladenović (2016). "Parallel VNS for bus terminal location problem". In: *Applied Soft Computing* 42, pp. 448–458.

Dorigo, M. and C. Blum (2005). "Ant colony optimization theory: A survey". In: *Theoretical computer science* 344.2-3, pp. 243–278.

Eberhart, R. and J. Kennedy (1995). "Particle swarm optimization". In: *Proceedings of the IEEE international conference on neural networks*. Vol. 4. Citeseer, pp. 1942–1948.

Fogel, D. B. (2006). *Evolutionary computation: toward a new philosophy of machine intelligence*. Vol. 1. John Wiley & Sons.

Frisch, K. (1968). "The role of dances in recruiting bees to familiar sites". In: *Animal Behaviour* 16.4, pp. 531–533.

Garcia-López, F., B. Melián-Batista, J. A. Moreno-Pérez, and J. M. Moreno-Vega (2002). "The parallel variable neighborhood search for the p-median problem". In: *Journal of Heuristics* 8.3, pp. 375–388.

Ghanbarzadeh, A. (2007). "Bees Algorithm: a Novel Optimisation Tool". PhD thesis.

Glover, F. (1977). "Heuristics for integer programming using surrogate constraints". In: *Decision sciences* 8.1, pp. 156–166.

Glover, F. and M. Laguna (1998). "Tabu search". In: *Handbook of combinatorial optimization*. Springer, pp. 2093–2229.

Gopalakrishnan Nair, G. (1979). "On the convergence of the LJ search method". In: *Journal of Optimization Theory and Applications* 28.3, pp. 429–434.

Gotardo, P. F., O. R. Bellon, and L. Silva (2003). "Range image segmentation by surface extraction using an improved robust estimator". In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 2. IEEE, pp. II–33.

Hansen, P. and N. Mladenović (2003). "Variable neighborhood search". In: *Handbook of metaheuristics*. Springer, pp. 145–184.

Hast, A., J. Nysjö, and A. Marchetti (2013). "Optimal ransac-towards a repeatable algorithm for finding the optimal set". In:

Hussein, W. A., S. Sahran, and S. N. H. S. Abdullah (2017). "The variants of the Bees Algorithm (BA): a survey". In: *Artificial Intelligence Review* 47.1, pp. 67–121.

Imanguliyev, A. (2013). "Enhancements for the Bees Algorithm". PhD thesis. Cardiff University.

John, H. (1992). *Holland, Adaptation in natural and artificial systems*.

Karaboga, D. and B. Basturk (2007). "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm". In: *Journal of global optimization* 39.3, pp. 459–471.

Kennedy, J. and R. Eberhart (1995). "Particle swarm optimization". In: *Proceedings of the 1995 IEEE International Conference on Neural networks*. IEEE, pp. 1942–1948.

Khoshelham, K. (2007). "Extending generalized hough transform to detect 3d objects in laser range data". In: *ISPRS Workshop on Laser Scanning and SilviLaser 2007, 12-14 September 2007, Espoo, Finland*. Citeseer.

Laguna, M. and R. Marti (2005). "Experimental testing of advanced scatter search designs for global optimization of multimodal functions". In: *Journal of Global Optimization* 33.2, pp. 235–255.

Levine, M. D. and M. D. Levine (1985). *Vision in man and machine*. Vol. 574. McGraw-Hill New York.

Liu, J. and Z.-k. Wu (2014). "An adaptive approach for primitive shape extraction from point clouds". In: *Optik - International Journal for Light*

and Electron Optics 125.9, pp. 2000–2008. ISSN: 0030-4026. DOI: https://doi.org/10.1016/j.ijleo.2013.03.176.

Lutton, E. and P. Martinez (1994). "A genetic algorithm for the detection of 2D geometric primitives in images". In: *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*. Vol. 1. IEEE, pp. 526–528.

Luus, R. (1974a). "A practical approach to time-optimal control of nonlinear systems". In: *Industrial & Engineering Chemistry Process Design and Development* 13.4, pp. 405–408.

Luus, R. (1974b). "Optimal control by direct search on feedback gain matrix". In: *Chemical Engineering Science* 29.4, pp. 1013–1017.

Luus, R. and T. Jaakola (1973). "Optimization by direct search and systematic reduction of the size of search region". In: *AIChE Journal* 19.4, pp. 760–766.

Mavrakis, N., R. Stolkin, L. Baronti, M. Kopicki, M. Castellani, et al. (2016). "Analysis of the inertia and dynamics of grasped objects, for choosing optimal grasps to enable torque-efficient post-grasp manipulations". In: *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, pp. 171–178.

Mitra, N. J. and A. Nguyen (2003). "Estimating surface normals in noisy point cloud data". In: *Proceedings of the nineteenth annual symposium on Computational geometry*. ACM, pp. 322–328.

Mladenović, N., J. Petrović, V. Kovačević-Vujčić, and M. Čangalović (2003). "Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search". In: *European Journal of Operational Research* 151.2, pp. 389–399.

Mladenović, N. and P. Hansen (1997). "Variable neighborhood search". In: *Computers & operations research* 24.11, pp. 1097–1100.

Molga, M. and C. Smutnicki (2005). "Test functions for optimization needs". In: *Test functions for optimization needs* 101.

Mukhopadhyay, P. and B. B. Chaudhuri (2015). "A survey of Hough Transform". In: *Pattern Recognition* 48.3, pp. 993–1010.

Muller, M. E. (1959). "A note on a method for generating points uniformly on n-dimensional spheres". In: *Communications of the ACM* 2.4, pp. 19–20.

Nair, G. G. (1979). "Suboptimal control of nonlinear time-delay systems". In: *Journal of Optimization Theory and Applications* 29.1, pp. 87–99.

Oh, S. H. and R. Luus (1976). "Optimal feedback control of time-delay systems". In: *AIChE Journal* 22.1, pp. 140–147.

Packianather, M., M. Landy, and D. T. Pham (2009). "Enhancing the speed of the Bees Algorithm using Pheromone-based Recruitment". In: *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*. IEEE, pp. 789–794.

Pham, D., M. Castellani, and A. Ghanbarzadeh (2007). "Preliminary design using the bees algorithm". In: *Proceedings of eighth international conference on laser metrology, CMM and machine tool performance, LAMDAMAP, Euspen, Cardiff, UK*, pp. 420–429.

Pham, D. T., L. Baronti, B. Zhang, and M. Castellani (2018). "Optimisation of Engineering Systems With the Bees Algorithm". In: *International Journal of Artificial Life Research (IJALR)* 8.1, pp. 1–15.

Pham, D. T. and M. Castellani (2009). "The Bees Algorithm: Modelling Foraging Behaviour to Solve Continuous Optimization Problems". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 223.12, pp. 2919–2938.

Pham, D. T. and M. Castellani (2014). "Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms". In: *Soft Computing* 18.5, pp. 871–903.

Pham, D. T. and M. Castellani (2015). "A comparative study of the Bees Algorithm as a tool for function optimisation". In: *Cogent Engineering* 2.1.

Pham, D. T., M. Castellani, and A. Fahmy (2008). "Learning the inverse kinematics of a robot manipulator using the bees algorithm". In: *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*. IEEE, pp. 493–498.

Pham, D. T., M. Castellani, and H. A. Le Thi (2014). "Nature-inspired intelligent optimisation using the bees algorithm". In: pp. 38–69.

Pham, D. T. and A. H. Darwish (2008). "Fuzzy selection of local search sites in the Bees Algorithm". In: *Proceedings of the 4th Virtual International Conference on Intelligent Production Machines and Systems*.

Pham, D. T. and H. A. Darwish (2010). "Using the bees algorithm with Kalman filtering to train an artificial neural network for pattern classification". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 224.7, pp. 885–892.

Pham, D. T., A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi (2006). "The bees algorithm - A novel tool for complex optimisation".

In: *Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference (3-14 July 2006)*. sn.

Pham, D. T. and E. Koç (2010). "Design of a two-dimensional recursive filter using the bees algorithm". In: *International Journal of Automation and Computing* 7.3, pp. 399–402.

Pham, Q. T., D. T. Pham, and M. Castellani (2012). "A modified bees algorithm and a statistics-based method for tuning its parameters". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 226.3, pp. 287–301.

Piotrowski, A. P. (2018). "Across Neighborhood Search algorithm: A comprehensive analysis". In: *Information Sciences* 435, pp. 334–381.

Rabbani, T. and F. Van Den Heuvel (2005). "Efficient hough transform for automatic detection of cylinders in point clouds". In: *Isprs Wg Iii/3, Iii/4* 3, pp. 60–65.

Raguram, R., J.-M. Frahm, and M. Pollefeys (2008). "A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus". In: *Computer Vision – ECCV 2008*. Ed. by D. Forsyth, P. Torr, and A. Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 500–513. ISBN: 978-3-540-88688-4.

Ross, S. M. (2014). *Introduction to probability models*. Academic press.

Roth, G. and M. D. Levine (1993). "Extracting geometric primitives". In: *CVGIP: Image Understanding* 58.1, pp. 1–22.

Roth, G. and M. D. Levine (1994). "Geometric primitive extraction using a genetic algorithm". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 16.9, pp. 901–905.

Rousseau, L.-M., M. Gendreau, and G. Pesant (1999). "Using constraint-based operators with variable neighborhood search to solve the vehicle routing problem with time windows". In: *Proceedings of the 1st Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*.

Sareni, B. and L. Krahenbuhl (1998). "Fitness sharing and niching methods revisited". In: *IEEE transactions on Evolutionary Computation* 2.3, pp. 97–106.

Schnabel, R., R. Wahl, and R. Klein (2007). "Efficient RANSAC for point-cloud shape detection". In: *Computer graphics forum*. Vol. 26. 2. Wiley Online Library, pp. 214–226.

Schwefel, H.-P. (1981). *Numerical optimization of computer models*. John Wiley & Sons, Inc.

Shatnawi, N., S. Sahran, and M. Faidzul (2013). "A memory-based Bees Algorithm: an enhancement". In: *J Appl Sci* 13, pp. 497–502.

Shir, O. M. (2012). "Niching in evolutionary algorithms". In: *Handbook of natural computing*, pp. 1035–1069.

Spiers, A. J., M. V. Liarokapis, B. Calli, and A. M. Dollar (2016). "Single-grasp object classification and feature extraction with simple robot hands and tactile sensors". In: *IEEE transactions on haptics* 9.2, pp. 207–220.

Stibor, T., J. Timmis, and C. Eckert (2006). "On the use of hyperspheres in artificial immune systems as antibody recognition regions". In: *International Conference on Artificial Immune Systems*. Springer, pp. 215–228.

Storn, R. and K. Price (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4, pp. 341–359.

Sveier, A. (2016). "Primitive Shape Detection in Point Clouds". In:

Swan, J., S. Adriaensen, M. Bishr, E. K. Burke, J. A. Clark, P. De Causmaecker, J. Durillo, K. Hammond, E. Hart, C. G. Johnson, et al. (2015). "A research agenda for metaheuristic standardization". In: *Proceedings of the XI metaheuristics international conference*.

Taillard, E. D., L. M. Gambardella, M. Gendreau, and J.-Y. Potvin (2001). "Adaptive memory programming: A unified view of metaheuristics". In: *European Journal of Operational Research* 135.1, pp. 1–16.

Thomsen, R. (2004). "Multimodal optimization using crowding-based differential evolution". In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*. Vol. 2. IEEE, pp. 1382–1389.

Ugolotti, R., G. Micconi, J. Aleotti, and S. Cagnoni (2014). "GPU-based point cloud recognition using evolutionary algorithms". In: *European Conference on the Applications of Evolutionary Computation*. Springer, pp. 489–500.

Whitaker, R. (1983). "A fast algorithm for the greedy interchange for large-scale clustering and median location problems". In: *INFOR: Information Systems and Operational Research* 21.2, pp. 95–108.

Wilson, E. B. (1900). *The cell in development and inheritance*. 4. The Macmillan Company.

Wolpert, D. H. and W. G. Macready (1997). "No free lunch theorems for optimization". In: *IEEE transactions on evolutionary computation* 1.1, pp. 67–82.

Wong, K.-C., C.-H. Wu, R. K. Mok, C. Peng, and Z. Zhang (2012). "Evolutionary multimodal optimization using the principle of locality". In: *Information Sciences* 194, pp. 138–170.

Yang, X.-S. (2012). "Swarm-based metaheuristic algorithms and no-free-lunch theorems". In: *Theory and New Applications of Swarm Intelligence*. InTech.

Yuce, B., M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase (2013). "Honey bees inspired optimization method: the bees algorithm". In: *Insects* 4.4, pp. 646–662.