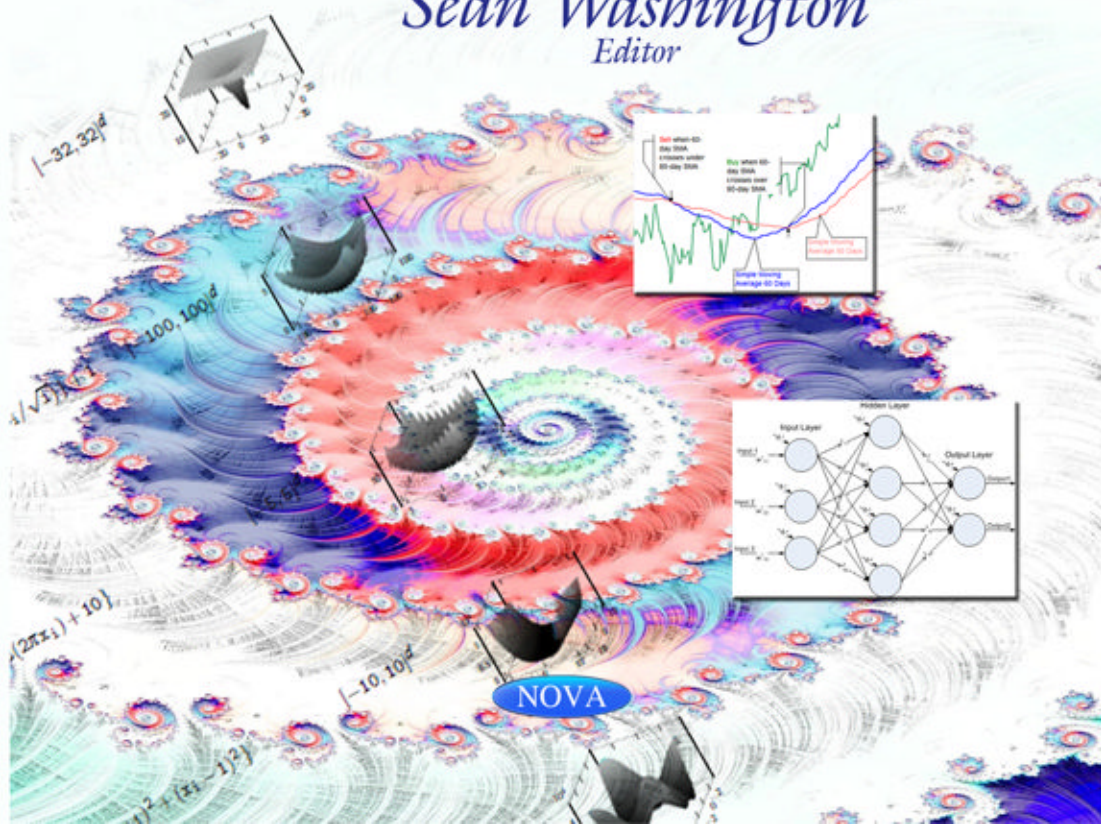
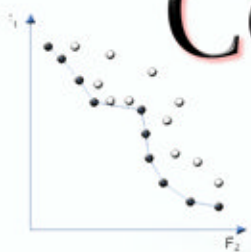




Computer Science, Technology and Applications

New Developments in Evolutionary Computation Research

Sean Washington
Editor



COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

**NEW DEVELOPMENTS
IN EVOLUTIONARY
COMPUTATION RESEARCH**

No part of this digital document may be reproduced, stored in a retrieval system or transmitted in any form or by any means. The publisher has taken reasonable care in the preparation of this digital document, but makes no expressed or implied warranty of any kind and assumes no responsibility for any errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of information contained herein. This digital document is sold with the clear understanding that the publisher is not engaged in rendering legal, medical or any other professional services.

COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

Additional books in this series can be found on Nova's website
under the Series tab.

Additional e-books in this series can be found on Nova's website
under the e-books tab.

COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

**NEW DEVELOPMENTS
IN EVOLUTIONARY
COMPUTATION RESEARCH**

**SEAN WASHINGTON
EDITOR**



Copyright © 2015 by Nova Science Publishers, Inc.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system or transmitted in any form or by any means: electronic, electrostatic, magnetic, tape, mechanical photocopying, recording or otherwise without the written permission of the Publisher.

For permission to use material from this book please contact us:
nova.main@novapublishers.com

NOTICE TO THE READER

The Publisher has taken reasonable care in the preparation of this book, but makes no expressed or implied warranty of any kind and assumes no responsibility for any errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of information contained in this book. The Publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or in part, from the readers' use of, or reliance upon, this material. Any parts of this book based on government reports are so indicated and copyright is claimed for those parts to the extent applicable to compilations of such works.

Independent verification should be sought for any data, advice or recommendations contained in this book. In addition, no responsibility is assumed by the publisher for any injury and/or damage to persons or property arising from any methods, products, instructions, ideas or otherwise contained in this publication.

This publication is designed to provide accurate and authoritative information with regard to the subject matter covered herein. It is sold with the clear understanding that the Publisher is not engaged in rendering legal or any other professional services. If legal or any other expert assistance is required, the services of a competent person should be sought. FROM A DECLARATION OF PARTICIPANTS JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Additional color graphics may be available in the e-book version of this book.

LIBRARY OF CONGRESS CATALOGING-IN-PUBLICATION DATA

New developments in evolutionary computation research / editor, Sean Washington.
pages cm. -- (Computer science, technology and applications)
Includes index.
ISBN: ; 9: /3/85685/747/9 (eBook)
1. Evolutionary computation. 2. Engineering mathematics. I. Washington, Sean.
TA347.E96N49 2014
006.3'823--dc23

2014042395

Published by Nova Science Publishers, Inc. † New York

CONTENTS

Preface		vii
Chapter 1	Multi-Objective Optimization of Trading Strategies Using Genetic Algorithms in Unstable Environments <i>José Matias Pinto, Rui Ferreira Neves and Nuno Horta</i>	1
Chapter 2	Promoting Better Generalisation in Multi-Layer Perceptrons Using a Simulated Synaptic Downscaling Mechanism <i>A. Brabazon, A. Agapitos and M. O'Neill</i>	73
Chapter 3	Plant Propagation-Inspired Algorithms <i>A. Brabazon, S. McGarraghy and A. Agapitos</i>	107
Chapter 4	Topographical Clearing Differential Evolution Applied to Real-World Multimodal Optimization Problems <i>Wagner F. Sacco, Ana Carolina Rios-Coelho and Nélio Henderson</i>	133
Chapter 5	Robotics, Evolution and Interactivity in Sonic Art Installations <i>Artemis Moroni and Jônatas Manzolli</i>	159

Chapter 6	An Analysis of Evolutionary-Based Sampling Methodologies <i>Yoel Tenne</i>	183
Index		215

PREFACE

A common approach for solving simulation-driven engineering problems is by using metamodel-assisted optimization algorithms, namely, in which a metamodel approximates the computationally expensive simulation and provides predicted values at a lower computational cost. Such algorithms typically generate an initial sample of solutions which are then used to train a preliminary metamodel and to initiate an optimization process. One approach for generating the initial sample is with the design of experiment methods which are statistically oriented, while the more recent search-driven sampling approach invokes a computational intelligence optimizer such as an evolutionary algorithm, and then uses the vectors it generated as the initial sample. This book discusses research and new developments on evolutionary computation.

Time-ordered sequences of data (Time Series data), have arisen across a broad range of applications in nearly all domains. In Chapter 1, extensive experiments using real-world data obtained from one of the most dynamic environments is used – Financial Markets. Additionally, a Multi-Objective Evolutionary System is used to predict future asset price evolution.

Therefore, in this study, a Genetic Algorithm (GA)-based Multi-Objective Evolutionary System to optimize a Trading or Investment Strategy (TS) was developed. The goal was to determine potential buy, sell, or hold conditions in stock markets while still yielding high returns at a minimal risk. Fair and established metrics were used (as described in the text) to evaluate both the returns and the linked risk of the optimized TS. Additionally, these TS are evaluated in several markets using data from the main stock indexes of the most developed economies, such as: NASDAQ, S&P 500, FTSE 100, DAX 30, and the NIKKEI 225. The Pareto Fronts obtained with the training data

during the experiments clearly showed the inherent tradeoff between risk and return in financial management.

Furthermore, the achieved results clearly outperformed both the B&H and S&H strategies. Regardless, the experimental results suggest that the positive connection between the gains for training data and test data, which was usually implied in the single-objective proposals, may not necessarily hold true in all circumstances.

Due to the fact that the objective in this kind of problem is to find the best (optimized) solution to conduct investment in stock markets, this chapter will begin with a review of the most recent advances in Computational Problem Solving Techniques, as well as the traditional ones. In this review, the various existing techniques of Intelligent Computing currently used to solve various optimization problems are presented. The various existing techniques, especially in the fields of time series forecast and systems that learn by example, are briefly reviewed.

A key concern when training a multi-layer perceptron (MLP) is that the final network should generalise well out-of-sample. A considerable literature has emerged which examines various aspects of this issue. In Chapter 2 the author's draw inspiration from theories of memory consolidation in order to develop a new methodology for training MLPs in order to promote their generalisation capabilities. The *synaptic homeostasis hypothesis* proposes that a key role of sleep is to downscale synaptic strength to a baseline level that is energetically sustainable. As a consequence, the hypothesis suggests that sleep acts not to actively strengthen selected memories but rather to remove irrelevant memories. In turn, this lessens spurious learning, improves the signal to noise ratio in maintained memories, and therefore produces better generalisation capabilities. In this chapter the author's describe the synaptic homeostasis hypothesis and draw inspiration from it in order to design a 'wake-sleep' training approach for MLPs. The approach is tested on a number of datasets.

Plants represent some 99% of the eukaryotic biomass of the planet and have been highly successful in colonising many habitats with differing resource potential. The success of plants in "earning a living" suggests that they have evolved robust resource capture mechanisms and reproductive strategies. In spite of the preponderance of plant life, surprisingly little inspiration has been drawn from plant activities for the design of optimisation algorithms.

In Chapter 3 the author's focus on one important aspect of plant activities, namely seed and plant dispersal. Mechanisms for seed and plant dispersal have

evolved over time in order to create effective ways to disperse seeds into locations in which they can germinate and become established. These mechanisms are highly varied, ranging from morphological characteristics of seeds which can assist their aerial or animal-mediated dispersion, to co-evolved characteristics which "reward" animals or insects who disperse a plant's seeds. At a conceptual level, dispersal can be considered as a "search process", wherein the seed or plant is searching for "good" locations and therefore, inspiration from dispersal activities of plants can plausibly serve as the design inspiration for optimisation algorithms.

Initially, the author's provide an overview of relevant background on the seed dispersal process from drawing on the ecology literature. Then the author describe a number of existing optimisation algorithms which draw inspiration from these processes, and finally the author's outline opportunities for future research.

Many real-world optimization problems are multimodal, requiring techniques that overcome local optima, which can be done using niching methods. In order to do so, in Chapter 4 the author's describe a niching method based on the clearing paradigm, Topographical Clearing, which employs a topographical heuristic introduced in the early nineties, as part of a global optimization method. This niching method is applied to differential evolution, but it can be used in other evolutionary or swarm-based methods, such as the genetic algorithm and particle swarm optimization. The algorithm, called TopoClearing-DE, is favorably compared against the canonical version of differential evolution in real-world optimization problems. As the problems attacked are quite challenging, the results show that Topographical Clearing can be applied to populational optimization methods in order to solve problems with multiple solutions.

Focusing on the interactivity that a robotic interface establishes between the virtual and the real world, some sensory systems and mobile robotic platforms were developed for the AURAL project, a robotic evolutionary environment for sound production. From the AURAL perspective, human and robots are agents of a complex system and the sonification is the emergent propriety produced by their interaction and behavior. One way to characterize types of interactions is by looking at ways in which systems can be coupled together to interact. The representation of the interaction between a person and a dynamic system as a simple feedback loop faces the role of information looping through both a person and a system. Two different sonification paradigms were applied in AURAL environment. In the first case, the sonification is generated by an evolutionary mapping of the robot trajectories

into sound events. In the second case, the sound production is the result of a generative process. As such the sonification here is not seen as an isolated aspect of AURAL, but as a representation of the synergetic capacity of the agents to collaborate and produce a complex product. A comparison between the results obtained with both approaches is presented in Chapter 5. The structure/novelty tradeoff has been approached.

A common approach for solving simulation-driven engineering problems is by using metamodel-assisted optimization algorithms, namely, in which a metamodel approximates the computationally expensive simulation and provides predicted values at a lower computational cost. Such algorithms typically generate an initial sample of solutions which are then used to train a preliminary metamodel and to initiate optimization process. One approach for generating the initial sample is with the design of experiment methods which are statistically oriented, while the more recent search-driven sampling approach invokes a computational intelligence optimizer such as an evolutionary algorithm, and then uses the vectors it generated as the initial sample. Since the initial sample can strongly impact the effectiveness of the optimization process, Chapter 6 presents an extensive comparison and analysis between the two approaches across a variety of settings. Results show that evolutionary-based sampling performed well when the size of the initial sample was large as this enabled a more extended and consequently a more effective evolutionary search. When the initial sample was small the design of experiments methods typically performed better since they distributed the vectors more effectively in the search space.

Chapter 1

MULTI-OBJECTIVE OPTIMIZATION OF TRADING STRATEGIES USING GENETIC ALGORITHMS IN UNSTABLE ENVIRONMENTS

José Matias Pinto^{}, Rui Ferreira Neves[†] and Nuno Horta[‡]*

Instituto de Telecomunicações, Instituto Superior Técnico,
Torre Norte, Lisboa, Portugal

Abstract

Time-ordered sequences of data (Time Series data), have arisen across a broad range of applications in nearly all domains. In this study, extensive experiments using real-world data obtained from one of the most dynamic environments is used – Financial Markets. Additionally, a Multi-Objective Evolutionary System is used to predict future asset price evolution.

Therefore, in this study, a Genetic Algorithm (GA)-based Multi-Objective Evolutionary System to optimize a Trading or Investment Strategy (TS) was developed. The goal was to determine potential buy, sell, or hold conditions in stock markets while still yielding high returns at a minimal risk. Fair and established metrics were used (as described in the text) to evaluate both the returns and the linked risk of the optimized TS. Additionally, these TS are evaluated in several markets using data from the main stock indexes of the most developed economies, such as: NASDAQ, S&P 500, FTSE 100, DAX 30, and the NIKKEI 225. The Pareto Fronts obtained with the training data during the

^{*} E-mail address: josempinto@tecnico.ulisboa.pt.

[†] E-mail address: rui.neves@tecnico.ulisboa.pt.

[‡] E-mail address: nuno.horta@lx.it.pt.

experiments clearly showed the inherent tradeoff between risk and return in financial management.

Furthermore, the achieved results clearly outperformed both the B&H and S&H strategies. Regardless, the experimental results suggest that the positive connection between the gains for training data and test data, which was usually implied in the single-objective proposals, may not necessarily hold true in all circumstances.

Due to the fact that the objective in this kind of problem is to find the best (optimized) solution to conduct investment in stock markets, this chapter will begin with a review of the most recent advances in Computational Problem Solving Techniques, as well as the traditional ones. In this review, the various existing techniques of Intelligent Computing currently used to solve various optimization problems are presented. The various existing techniques, especially in the fields of time series forecast and systems that learn by example, are briefly reviewed.

Keywords: Multi-Objective Optimization, Stock Market Prediction, Technical Analysis, Financial Markets, Moving Average, Dynamic Systems

A. Part A. Review of the Main Problem Solving and Optimization Techniques

In this study, it is advantageous to solve the established problem in a way that uncovers the best possible solution. Therefore, a review of the main traditional (historic) and the most recent developments in the field of Computational Problem Solving Techniques will be helpful.

Therefore, in this section, a review of the main approaches found in traditional and relatively recent publications concerning combinatorial optimization and problem-solving techniques are presented, which can be listed as:

- Newton's Method
- Exhaustive Search
- Random Search
- Quadratic Programming (QP)
- Expert Systems
- Artificial Neuronal Networks (ANN or NN)
- Metaheuristic Methods
 - Hill Climber (HC), Simulated Annealing (SA), and Stochastic Hill Climber (SHC)
 - Tabu search (TS)

-
- Evolutionary Computing (EC) / Genetic Algorithms (GA)
 - Memetic Algorithms / Hybridization
 - Ant Colony Optimization (ACO)
 - Particle swarm optimization / Swarm intelligence
-
- Fuzzy Logic
 - Agents
 - Support vector machine (SVM)
 - Other: (Classification Systems, Guided Local Search, GRASP)

Metaheuristic methods are computational methods or techniques to optimize or find an optimal or near-optimal solution to a given problem. This is done by iteratively improving a candidate solution with regard to a given estimate of the quality (usually called fitness). Metaheuristic methods make few assumptions or have absolutely no knowledge about the problem to be optimized and search the entire space of possible solutions. Many metaheuristic methods are stochastic. Contrary to the classic optimization methods, it is not required that the optimization problem is differentiable. Therefore, metaheuristic methods can be very helpful when optimizing problems that are partially irregular, noisy, or dynamic over time.

A.1. Newton's Method

Newton's method was discovered in 1669, but was not published until 1711. In 1690, it was enhanced by Joseph Raphson, giving birth to the Newton-Raphson method. In optimization, Newton's method is focused on finding fixed points of differentiable functions, which are the zeros of the derivative function.

This technique requires the problem to be mathematically formulated; after the problem is formulated, the functions should be derived.

The downside is that many real problems are usually formulated in the form of a data series, whereas to solve it using this method, a mathematic formulation is needed.

A.2. Exhaustive Search

The Exhaustive Search method is also a deterministic technique. Exhaustive Search (also known as the "brute force" method), as the name implies, consists, in

a combinatorial problem in the complete enumeration of all possible problem solutions, as well as its corresponding evaluation. This method guarantees that the best solution is always found. This method is expeditious for small problems, but for combinatorial problems, where the search space is huge, it is impractical. Therefore, the solution found is deterministic, and is always assured to be the best one possible.

A.3. Random Search

The Random Search method can be classified as a variation of the Exhaustive Search method; instead of enumerating all of the solutions, the search space is randomly sampled by casual solutions that are generated and tested. This method does not guarantee that the best solution will be found. This method can be used for small and large problems; however, its accuracy is proportional to the number of random samples tested.

A.4. Quadratic Programming

Quadratic programming (QP) requires the problem to be mathematically formulated; after the problem is formulated, the problem is mathematically solved in a precise way. Hence, this is a particular type of mathematical optimization problem. The problems solved by QP are problems related to maximizing or minimizing a quadratic function of several variables subject to linear constraints on the variables, as expressed in Equation 1.

Equation 1. Quadratic Problem Formulation:

$$f(X) = \frac{1}{2} X^T Q X + c^T X \quad (1)$$

Subject to one or more constraints, in the form:

$$\begin{aligned} A X &\leq b && \text{(Inequality constraint)} \\ E X &\leq d && \text{(Equality constraint)} \end{aligned}$$

where X is the vector of unknown variables to optimize and X^T is the vector transpose of X , while the notation means that every entry of vector $A(x)$ is less than or equal to the corresponding entry of vector b .

Several tools are freely available on the Internet that, after the problem is correctly formulated, can solve the problem and find the optimal exact solution [1].

The main advantage of QP is that when a feasible solution is found; it is guaranteed that it corresponds to the optimal solution.

The downside is that, typically, in financial computing, the available information about the problem is usually in the form of time series of data, whereas to solve the problem using QP, a mathematic formulation is needed.

QP has been extended and used as a standard to compare diverse approaches to solve financial problems. QP solves Portfolio Optimization Problems in a proficient and optimal way if all the constraints are linear. However, no systematic method exists that can solve this kind of problem when the restrictions are non-linear, such as with asset cardinality, transaction costs, or minimum and maximum weights, in addition to others that are present in real-world problems.

A.5. Expert Systems

An Expert System is a system that imitates the reasoning of a human specialist. Expert systems solve complex problems by building chains of reasoning based on known facts. The thinking process follows the chain of thinking of a trained professional, which is sometimes more intricate and dissimilar from the typical thinking of a computer programmer (making an analogy to traditional programming). Expert systems were quite popular in the 1970s and 1980s and are considered one of the first successful forms of AI software.

Typically, an Expert System is composed of two parts: the inference engine and the knowledge database. The engine reasons about the facts in the knowledge base and obtains conclusions. The knowledge database accumulates rules written by a programmer, or by an expert, or even those acquired from some other method. In more recent versions, Expert Systems have been improved with another component: an interface to communicate or conduct some form conversation with the users.

The rules accumulated in the knowledge base are expressed in natural language and represented in the form of: "IF...THEN..." clauses. An instance of such a rule is Descartes's famous aphorism: "IF I think, THEN I exist". This formulation has the advantage of using current and natural language that makes it attractive to people not accustomed to computer science (e.g. familiar with classic program coding). Rules expressing knowledge are exploited by the expert system. Other alternative formulations for the rules are used by many systems, although some of them aren't expressed using everyday language and are only

comprehensible by experts. Some rules are engine-specific and consequently not recognized by different systems. The goal of the knowledge base is to collect knowledge, which is sometimes unconsciously used by the specialists (but important to the inference process). The system of rules can be extended to allow probabilistic reasoning, and can therefore accept rules like the following one: “IF the sky is full with black clouds, THEN there is a strong probability (0.75) that in the next four hours, it will rain”.

The inference engine produces reasoning or draws conclusions from the rules stored in the knowledge base. It is usually a computer program that is designed to accomplish the desired task. The reasoning of the inference engine is based on logic, namely propositional logic, epistemic logic, fuzzy logic, or other types. An example of a well-known propositional logic engine is PROLOG; another powerful expert system is the NEXPERT system, [2] which was made available by Neuron Data.

As stated above, Expert Systems were popular in the 1970s and 1980s, but there has been almost no recent activity in this field, so it is difficult to find recent publications exploring it further.

A.6. Artificial Neuronal Networks

Artificial Neuronal Networks (ANN), or simply Neuronal Networks (NN) [3] are a computational technique developed in the field of Artificial Intelligence that imitates the workings of the human brain, namely the neuronal cells. In certain aspects, it is considered more efficient for solving certain kind of problems than traditional computational techniques.

The essential unit of an NN is the neuron, represented in Figure 1. A neuron receives one or more inputs from dendrites or synapses, which are represented in the figure by arrows.

The neuron multiplies the inputs by a factor (weight) and sums them to produce a weighted sum of all the inputs. This result is fed to a non-linear function, which usually has a sigmoid profile, and the output of the neuron is established.

For a given neuron, with n input signals, x_1 through x_n , and weights, w_0 through w_n , the output of the neuron is provided by Equation 2. Usually, the input x_0 is assigned a value of +1, which makes it a biased input with $w_0^k = b^k$; φ is the transfer function (sigmoid).

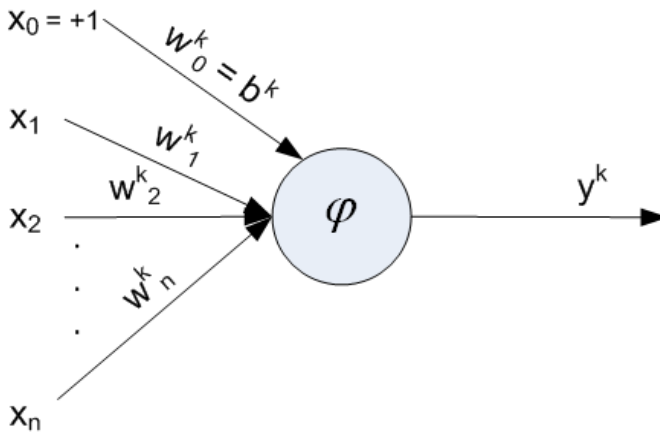


Figure 1. The Model of an Artificial Neuron.

Equation 2. Output of a Neuron:

$$y^k = \varphi\left(\sum_{i=0}^n w_i^k * x_i\right) \quad (2)$$

In a NN, several neurons are coupled together. The connections between the neurons are represented by arrows and to each connection, its weight is coupled, which corresponds to its influence / linking on the following neuron.

Figure 2 represents a typical NN with 3 layers, more specifically the input layer, the output layer, and a hidden or intermediary layer. In each layer, the neurons are represented by circles. Usually, the number of neurons in the input layer is equal to the number of problem inputs; the number of neurons in the output layer is equal to the number of problem outputs. The number of neurons in the intermediate or hidden layer is variable, although this number should be proportional to the required processing power. Each neuron in the hidden layer is connected to all the neurons in both the input layer and the output layer.

The NN in Figure 2 is a “feed forward” network, although other different configurations are possible, namely with back propagation, where the output of a neuron is connected to itself, or to the input of another neuron in a previous layer. In this illustration some weight terms are omitted for clarity, this network has 3 inputs, 4 nodes in the hidden layer, and 2 outputs.

NN are essentially useful both when there is no information about the shape of the output function $[f(x)]$ in advance, it is computationally difficult to uncover it, and there is a representative sample of inputs and outputs available to use as a

training set. On the other hand, if additional information on the function $f(x)$ is known, then other estimation techniques are likely to work better. In brief: NN are useful for recognizing patterns in complex data.

The most important goal of neuronal networks is to uncover problem resolutions using the same processes that living organisms use; this method is founded, essentially, in the trial-and-error method.

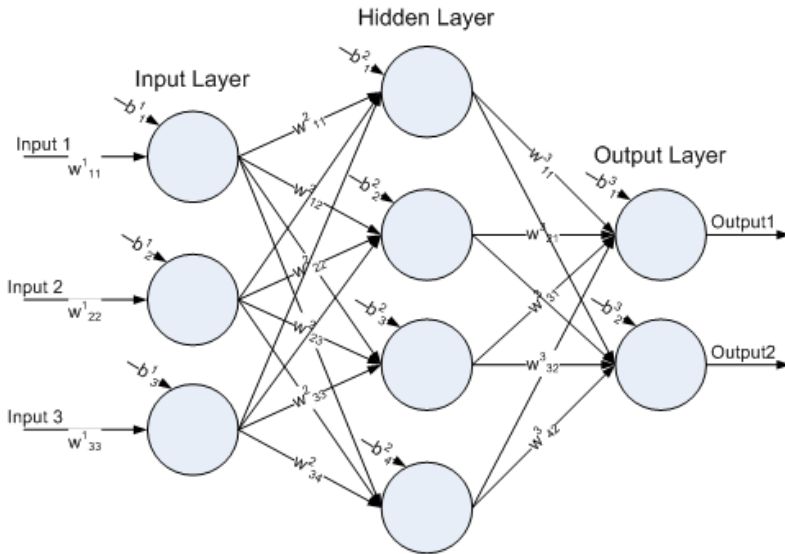


Figure 2. Artificial Neuronal Network With Weight and Bias Terms Labeled.

Its accuracy increases as more data examples are provided. The main benefit of this approach is that special knowledge about the problem is not necessary to solve it, as the connections between the pieces of information are discovered and explored by the neuronal network itself. It is simply necessary to provide the input data used to train the network.

The main critiques of this approach is that the rules discovered by the NN cannot be easily described in a way that is easily understandable by humans; the model cannot be easily translated and analyzed, even by experts, to possibly be improved.

Several articles have successfully demonstrated good results when applying ANN to discover technical trading rules, particularly in the case of Fernando Fernández-Rodríguez et al. [4] and A. Skabar et al. [5]. In [6], a GA was used to optimize several parameters of a NN, which generated interesting results.

In some studies, the neuronal networks were used to pre-process or post-process data, and were used in combination with other optimization techniques. Some examples of this combinative approach include William Leigh et al. [7] and Takashi Kimoto et al. [8]. Other works have showed that GP is superior to ANN, particularly the study conducted by Hitoshi Iba et al. [9].

A.7. Hill Climber (HC), Simulated Annealing (SA), and Stochastic Hill Climber (SHC)

A straightforward method for finding the best solution to a problem would be starting from any point in the search space (random), testing all the points in that point's neighborhood, move to the best point, and repeat this process until no further improvement is possible. This process is called Iterative Improvement or Hill Climber (HC) and is represented by Figure 3.

This algorithm stops when it finds a local optimum, as it only scans the neighborhood of the current solution. The performance of the iterative improvement method is poor. Therefore, alternative techniques have been developed to avoid algorithms from becoming trapped in local optima; this can be done by adding mechanisms that allow algorithms to escape from local optima. This resulted in the emergence of one of the oldest strategies to avoid algorithms becoming trapped in a local optimum, which is called Simulated Annealing (SA).

The fundamental idea behind SA is to allow movement to solutions of lower quality than the current solution, which is done stochastically. The probability of accepting a new solution varies with the relative performance of the new solution, for example, $rp = f(v) - f(u)$, where $f(v)$ is the evaluation or performance of the new solution and $f(u)$ is the performance of the current solution.

This probability is usually calculated using a Boltzmann distribution expression. An example that implements this idea is presented in Equation 3.

Equation 3. SHC Prob. of Accepting a New Solution:

$$p(rp) = \frac{1}{1 + e^{\frac{rp}{T}}} \quad (3)$$

In this equation, $p(rp)$ is the probability of accepting a new solution, T is is: the probability of accepting a new solution, and rp is the Relative Performance. Figure 4 represents several plots of Equation 3 for several values of T , function of rp . In this figure, for small values of T , the probability of accepting a new solution

is almost deterministic with a new solution always being accepted when it is better than the previous one. Conversely, for higher values of T, the function is practically flat at around 0.5, with the probability of accepting the new solutions being almost random.

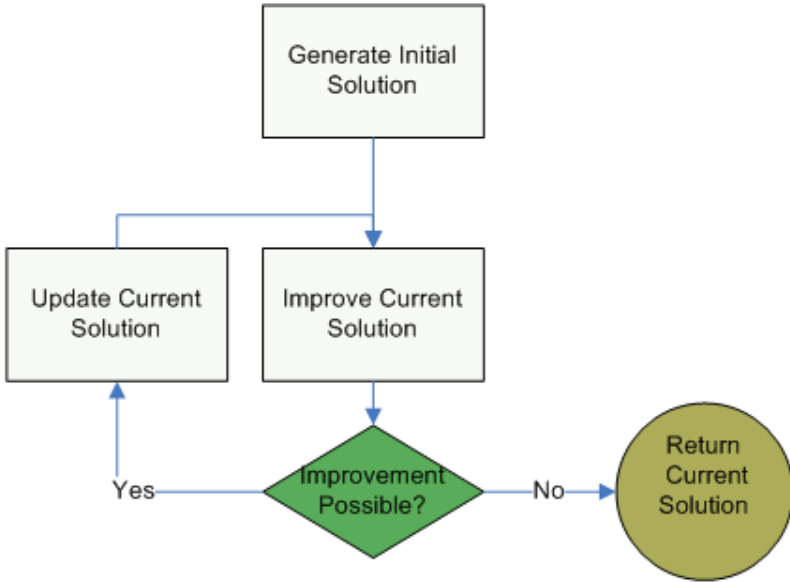


Figure 3. Algorithm: Iterative Improvement Algorithm.

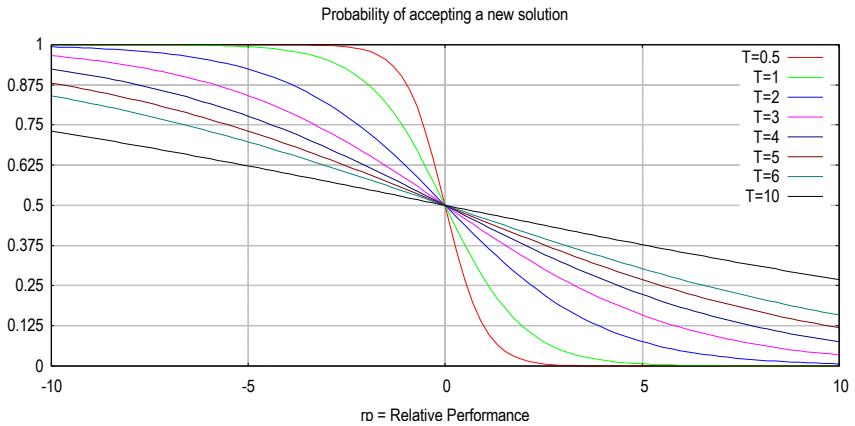


Figure 4. SHC: Probability of Accepting a New Solution.

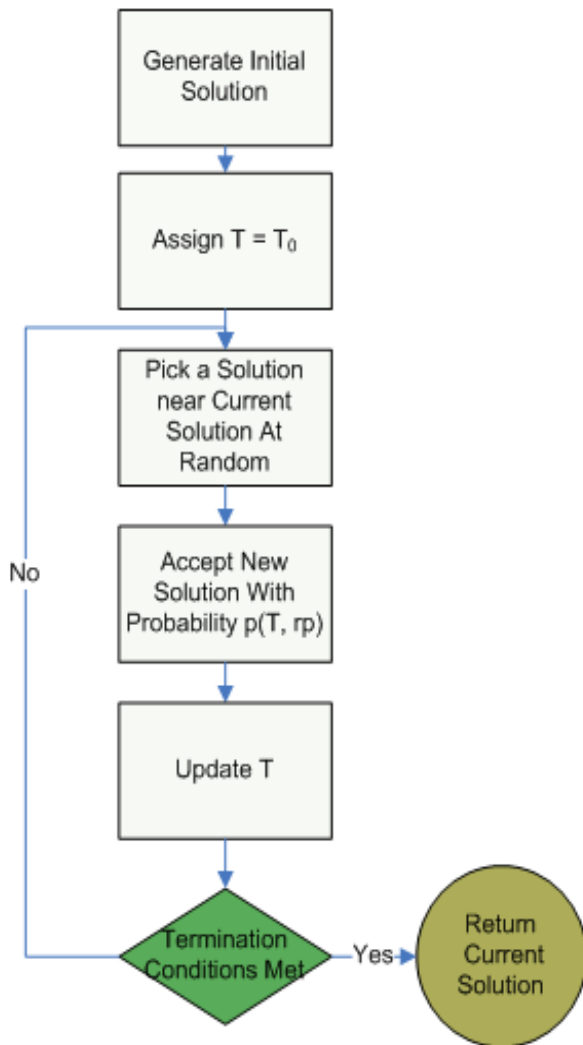


Figure 5. Algorithm: Simulated Annealing.

Figure 5 illustrates how the SA algorithm works; from this illustration, it is easy to observe that the choice of an appropriate value of T is crucial for the performance of the algorithm. In the SA method, the value of T is varied, and is typically set to a high value in the beginning of the process and then decreases during the progression, but elaborate cooling schemes can incorporate a sporadic rise in the temperature.

The introduction of the probability factor makes the process stochastic, but this allows the process to escape from local optima and instead find near-optimal solutions.

This also entails that the termination criteria must be more elaborated upon than simply reaching a local optimum. Possible termination conditions can be the same as presented in subsection A.9.1. Possible Termination Criterion.

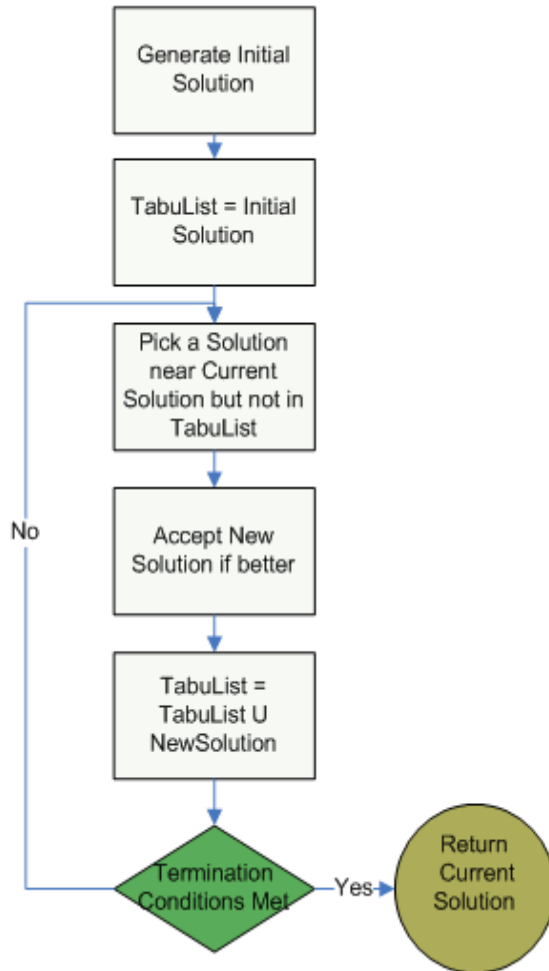


Figure 6. Algorithm: Taboo Search.

This process is similar to the annealing process of metals and glass, which supposes a low-energy configuration when cooled with a suitable cooling schedule. Concerning the algorithm, it is the result of two joint strategies: random walk and greedy search. At the beginning of the search process, the probability of making improvements is low, which favors the exploration of the search space. As the end of the search approaches, the random component is slowly reduced; consequently, the search must converge on an optimal point (which can be either local or global).

A small variation of this algorithm exists, albeit less sophisticated, where the temperature T is constant for the entire run; this method is called Stochastic Hill Climber (SHC). As the name implies, it is almost identical to the HC, except for the introduction of a small unpredictable factor during the climb (that is: the probability of accepting a new solution).

A.8. Taboo Search (TS)

The simple TS algorithm, depicted in Figure 6, applies the basic HC strategy and uses short-term memory to avoid repeating points that have been previously tested, consequently avoiding cycling through solutions. The short-term memory “tool” is implemented as a taboo list that remembers the most recently visited solutions and forbids moving to them.

The neighborhood of the current solution is therefore limited to the solutions that are not in the taboo list. In each iteration, the best solution from the permissible set is chosen as the new current solution. Furthermore, this solution is added to the taboo list. If the taboo list is full, then one previous solution must be removed from the taboo list (usually an older one). The algorithm terminates when the termination clause is met. It may also terminate if there are no allowed moves, that is: all the solutions near the current solution are forbidden, as they are already in the Taboo List.

A.9. Evolutional Computing / Genetic Algorithms

Evolutionary Computation is a field within a broader area called Computational Intelligence or Artificial Intelligence. Evolutionary Computation is primarily focused on discovering the best solution for a given problem.

The Evolutionary Algorithm (EA) [10-12] is a subset of the Evolutionary Computation and is a broad, population-based, metaheuristic optimization algorithm. All EA have certain facts in common, such as being population-based

and using some mechanisms inspired from Darwin's theory of evolution, namely, selection (survival of the fittest) and reproduction (with mutation and recombination).

Some of the most widely used EA techniques are: Genetic Algorithm (GA), Genetic Programming (GP), Evolutionary Programming (EP), and Evolutionary Strategies (ES).

Although some confusion may exist in the scientific community regarding the accurate meanings of all these terms, some level of common consensus may be drawn about the following facts (even if diverse interpretations may still be found):

- All are inspired by the laws of Natural Evolution, namely on Darwin's theory of evolution and relate to evolving a set (population) of potential problem solutions.
- In GA, the problem solution is encoded in the form of strings of numbers (traditionally binary, but others can be used – this is problem-dependent). The use of structures of variable size organized in the list also fit in this technique, according to Angan Dass *et al.* [13].
- In GP, the solutions are usually structures in tree form to symbolize computer programs, and the goal is to solve a computational problem. John Koza popularized this method with many available papers that applied these fresh techniques to many research fields [14].
- EP is similar to GP in terms of goals, but the structures denoting the solution are of a fixed size. Therefore, only the numerical parameters are permitted to evolve.
- In ES, the solution to the problem is usually encoded in the form of strings of real numbers, while variable parameters are used to adjust some control variables.

There are good reasons for the existence of all these related terms, and the differences between them, as some seem to correspond only to minor or cosmetic changes, while others really mark the difference and the birth of a new generation of approaches.

The first instances of what is presently called EA appeared in the early 1960s and was programmed on computers by biologists who were explicitly seeking to replicate aspects of natural evolution. In the early 1970s, EA became widely recognized as an optimization method. In 1975, John Holland's paper [15] greatly contributed to the popularity of GA. Another great contribution came in 1990 by John Koza [14], with his extensive list of papers that explained and applied GP to

many fields of study. More recently, Kalyanmoy Deb [16-18] has also made sizeable contributions to the field.

All EA optimization algorithms develop and optimize a possible solution to a given problem. These algorithms encode the possible problem solution in a data structure. To extend the analogy of natural evolution, these data structures are called chromosomes, and each data element is called a gene. The term “population” is used to label all of the possible problem solutions found up to a given point. Essentially, a chromosome groups several genes, and a population is a set of chromosomes.

These EC machine learning techniques are all a repetitive process that begins with a population of solutions for a given problem. This process uses the same mechanisms of natural evolution to evolve this population, namely through selection and reproduction.

Initializing the population, meaning to form the first generation of solutions, is usually a random generation process. Occasionally, the first set of solutions may be created using some heuristics in areas where previously existing problem knowledge is available and where it is relatively easy to generate these solutions. A good example of this can be found in [13], where the whole process was accelerated and the exploration of impossible solutions was avoided (according to what the authors said).

The process of selection evaluates the individuals of the population (problem solutions) according to their capability to perform or solve the concrete and given task. The best performing individuals are selected for reproduction, while the worst ones are left to die or are replaced by new ones; this corresponds, in natural evolution, to the concept of “survival of the fittest”. After that initial “weeding out”, the best-selected individuals are then used for reproduction.

In the reproduction process, the individuals used for reproduction are called fathers and give birth to new offspring. Connecting the analogy to natural evolution once again, the genes encoded in the selected fathers are used as a suggestion for the new offspring. There are two main processes in current use for reproduction: crossover and mutation (although many others are possible).

For the crossover process, a minimum of two ancestors are necessary and the process gives birth to one or two sons. The crossover exchanges parts of both parents to produce the offspring. In the mutation process, random changes are made in the father’s chromosome; this process generates one offspring, and in this case, only one ancestor is needed. The main aim of the mutation process is to preserve population variety and escape from local optima. It is important to stress that both of these methods are methods of changing individuals, so must therefore alter the individuals.

This is done to introduce new solutions to the population, hoping that some may be better than their predecessors (improving, in this way, the fitness of the next generation).

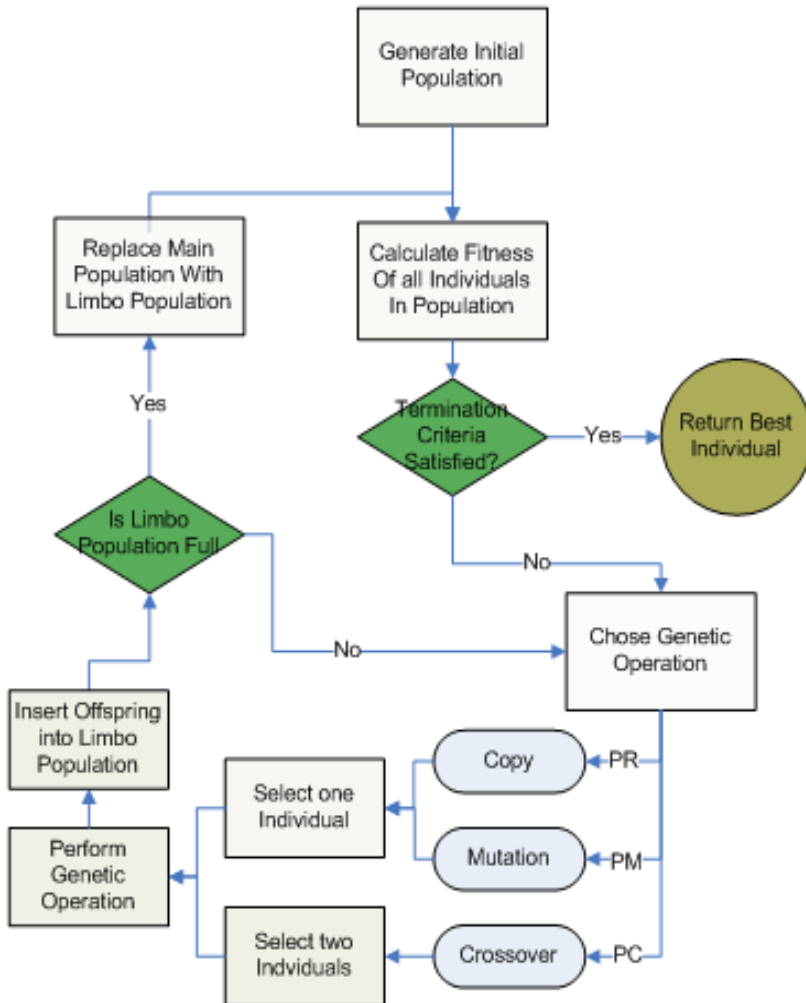


Figure 7. Algorithm: Evolutionary Algorithm.

A.9.1. Possible Termination Criterion

This process is repeated until a termination criterion is met, which can be:

- A solution is found that satisfies a minimum fitness assessment.
- A fixed number of generations is attained.
- A predetermined budget (computation time/money) is reached.
- It is detected that subsequent iterations no longer generate better results.
- Some combination of the above.

This iterative method is provided above in Figure 7, although other variations are possible. However, the process is always alike.

Some of the procedures involved in the process are stochastic processes, which means that the subsequent state is non-determinist and is controlled by a random process.

EA are often viewed as function optimizers, although the range of problems to which genetic algorithms can be applied is fairly extensive.

As amazing and counterintuitive as it may appear [11], it has been proven that EA are a potent and successful problem-solving approach, which demonstrates the power and validity of the evolutionary principles. EA has been used in a broad diversity of fields to evolve solutions to problems of similar or even superior complexity than those solved by human professionals. Furthermore, the solutions they find are often more effective than anything a human engineer would create.

Compared to traditional optimization methods, EA are faster and more adaptable to changes in the environment. This is because any information learned up to any point about how to solve a problem is enclosed in the population of solutions that has survived up to that point.

One of the big differences between GA and NN is that the rules obtained by the GA are more easily understandable by humans.

Some examples of EA applied to financial computing can be found in the works of Wang et al. [19], Badawy et al. [20], and Fernández-Blanco et al. [21]. GA can also be used to uncover and optimize new investment strategies as achieved by Bodas-Sagi et al. in [22], or to optimize asset weight in a portfolio optimization problem, such as in Gorgulho et al.'s study [23]. Leigh et al. [24] proposed the use of a GA to preprocess the input data before feeding it to a NN.

EA are usually applied to complex problems where the search space is vast to perform an exhaustive or other kind of search when there is no alternative analytical problem solution. Even when the number of alternative solutions can be considered not high enough to do an exhaustive search, the EA are still helpful, as they more efficiently explore the search space and come to a solution faster than an exhaustive search algorithm.

A.10. Memetic Algorithms (MA) / Hybridization

Memetic Algorithm (MA) [25] is one of the recent emergent areas of research in EC. The term MA is now widely used in an effort to combine several evolutionary techniques; the population-based ones, as well as the individual learning or local improvement techniques for problem solving. Quite often, MA is also referred to, in the literature, as cultural algorithms, genetic local searches, or hybrid GA.

A meme is a cognitive or behavioral pattern that can be transmitted from one individual to another. From a simplified point of view, one can interpret memetics as being another variance of the EA procedures where the concept of a chromosome is replaced by the concept of meme. The major difference between memes and chromosomes is that chromosomes can only be transmitted from parent to son ("vertical transmission"), whereas memes can be passed between any two individuals ("horizontal transmission" or "multiple parenting").

MA is often referred to as a hybrid algorithm, as it is a marriage between a population-based global search algorithm and individual evolutionary strategies in the form of local refinement.

MA intends to use hybrid metaheuristic techniques for optimization in continuous and discrete optimization domains. Memetic Computing is proposed to be where the newest results in Natural Computation, Artificial Intelligence, Machine Learning, and Operational Research join together in a fresh technique to go beyond the inherent boundaries of a single subject.

One of the issues in the tuning of an MA strategy is to decide when in the evolutionary process and which individuals should experience a local improvement (every generation, every 100 generations, all the generations but only a small set of chosen individuals, the individuals that have several copies of themselves in the population, etc.).

In relation to Financial Computing, [26] and [27] presented memetic algorithms to be used for portfolio selection. Under the point of view that the combination of an EA with a local search technique is an MA, although not being claimed as such, the works of Diego J. Bodas-Sagi et al. [28] and P. Fernández-Blanco et al. [29], can also be considered to be as such, as they included in their EA, a "local search operator" to improve the solutions, used "Every 100 generations".

A.11. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) or simply Ant System (AS), as it was presented in its original idea, was a technique for problem solving aimed at finding an

optimal path through a graph based on the behavior of ants. Later, the original idea was enhanced to solve a wider class of numerical problems.

Again, this algorithm was inspired by the natural analogy of how ants look for food: ants initially wander (randomly) for food and when something is found, they return to the colony carrying it; meanwhile, they leave pheromone trails to mark their way back. Another ant, looking for food, can get lucky and follow another ant's pheromone trails instead of wandering randomly; this also reinforces the former pheromone trails if it does turn out to lead to food. However, over time, the pheromone trails will evaporate, consequently reducing its strength of attraction. With more time and more ants looking for food, the most short and promising itineraries will be the most widely used, while the pheromone trails of less promising paths will evaporate without being reinforced. Due to this positive feedback being given, all ants will rapidly pick the shorter path.

The first algorithm that can be classified in this group was presented by A. Colomi et al. in [30] and by G. di Caro et al. in [31]. Since then, several different variants of the same basic principle have been proposed.

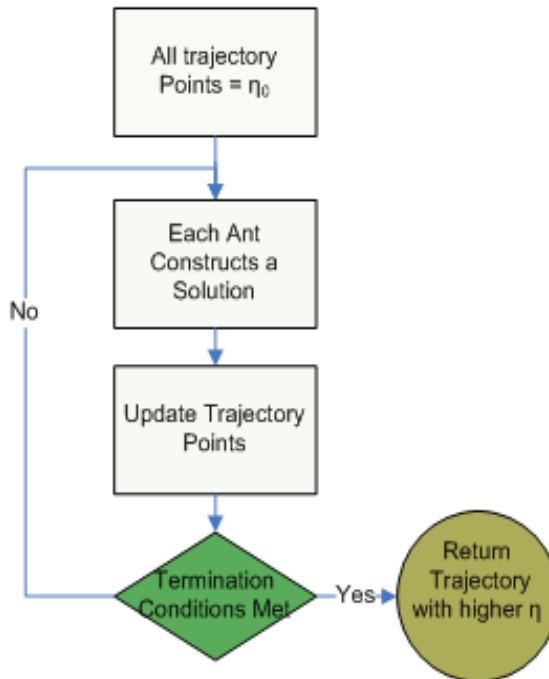


Figure 8. Algorithm: Ant System.

Figure 8 depicts a possible algorithm implementing an AS.

At the beginning, the pheromones of all trajectory points are set to an equal initial value; next, all the ants start building a solution. Figure 9 presents how each ant constructs its solution to a certain level of detail. Every ant chooses its next move based on the pheromones it finds at nearby points; this is done probabilistically, with a higher probability for the points with higher pheromones, and those not already in the ant move set (to avoid entering a cycle and repeating points already visited). When an ant finds a solution, the pheromones in the solution are updated (incremented) by a certain value.

The AS also borrows some of the techniques used in older algorithms, such as SHC and TS, since every ant has a Tabu List of paths, which corresponds to the list of all the already visited points; when choosing from the list of available points to do the next move, this is done stochastically.

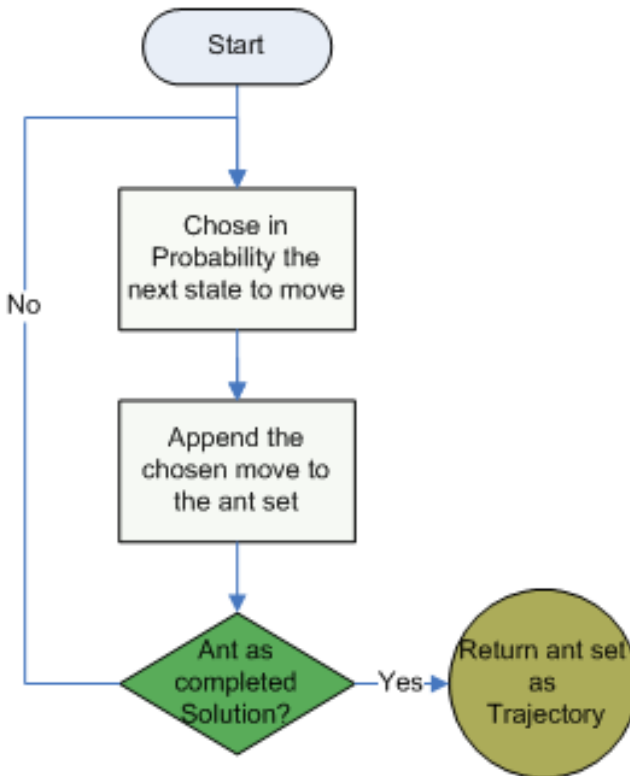


Figure 9. Algorithm Detail: Each Ant Constructs a Solution.

The algorithm has several parameters to tune, such as the amount of the pheromone increase when a path is used, the ratio of the pheromone decrease with time, and so forth. When the termination criteria are met, the path with higher η (pheromones) is returned, as this should be the most efficient one, since it was the most popular among the ants. More information about ACO and AS is available in [32, 33].

A.12. Particle Swarm Optimization / Swarm Intelligence

Particle Swarm Optimization (PSO), or simply Swarm Intelligence [34], is similar to EA, in the sense that a population of candidate solutions is also developed. In PSO, the individuals (or chromosomes) are replaced by the term “particle”, whereas a population is called a “Swarm”.

The natural analogy with what occurs in nature was taken from the movement of groups of animals like swarms of bees, shoals of fish, or flocks of birds. In PSO, all the particles explore the search space, moving on it with a specified velocity and in a certain direction according to few simple formulae. The particles move towards the best known solution and, when a new best solution is found, this one becomes the new swarm guide. This process is repeated, hoping that a reasonable solution will ultimately be uncovered (but that is not guaranteed).

The direction of the vector governing the particle movement is towards the best known swarm point, but is distorted by some random factors added to it. This way, all the particles come close to the best known solution, but in its walk, they explore the space around it.

Figure 10 provides an algorithm implementing PSO:

PSO is initialized with a set of particles (solutions) being dropped randomly in the search space; then, it searches for the optimum point through a series of generations. In every iteration, each particle is updated to follow two "best" values. One is the best solution (fitness) that it has achieved so far, called *pBest*. The other "best" value that is kept by the algorithm is the best value obtained so far by any particle in the population. This best value is a global best and is called *gBest*. After finding the two best values, the particle velocity and position are updated according to Equation 4 and Equation 5, below.

Equation 4. PSO Vel. Update:

$$\vec{v} = \vec{v} + c_1 \times \text{rand}() \times (\overrightarrow{pBest} - \overrightarrow{present}) + c_2 \times \text{rand}() \times (\overrightarrow{gBest} - \overrightarrow{present}) \quad (4)$$

Equation 5. PSO Pos. Update:

$$\overrightarrow{present} = \overrightarrow{present} + \vec{v} \quad (5)$$

where $v[]$ is the vector representing the particle velocity, $present[]$ is the current particle position. $pPest[]$ and $gGest[]$ are the current Population and Global Best found and are maintained as explained before, while $rand()$ is a random number between 0 and 1.

On these same equations, $c1$ and $c2$ are parameters, it is usual to set both parameters to the value of 2 (although other values are possible). The choice of suitable learning factors (parameters $c1$ and $c2$) is of crucial importance for PSO algorithm performance, primarily to avoid being trapped in a local optima due to premature convergence. To avoid having all of the swarm converge to a single point, there are implementations where the particles, rather than converging to the global optima, converge to its neighborhood's best known. This entails the existence of a communication network within a given topology that the particles use to communicate between themselves.

In recent years, PSO has effectively been used in many research and application areas. It has been confirmed that PSO can get better results, faster and in an easier way, as compared to other techniques. A good book about this subject was written by James Kennedy et al. [35] where more information can be found about PSO.

R. Hassan et al.'s [36] study showed that PSO-based algorithms find solutions of matching quality to those solutions ones found by GA-based algorithms; however, PSO is computationally more efficient. Margarita et al.'s [37] studies distinguished two major explanations for PSO popularity: (1) they are relatively easy to implement, and (2) they are very efficient in a variety of applications with superior results and low computational effort.

In terms of financial computing, Matthew Butler et al. [38] used a PSO-based approach to tune the parameters of a financial indicator (BB); Antonio C. Briza et al. [39] proposed a PSO system to perform MultiObjective Optimization (MOPSO) applied to stock trading.

A.13. Outline of the Metaheuristic Methods

To conclude this review of Metaheuristics optimization methods, a summary of the Metaheuristics optimization methods can be found in [40]. In Figure 11, a diagram presenting the diverse classifications of the metaheuristic methods is shown.

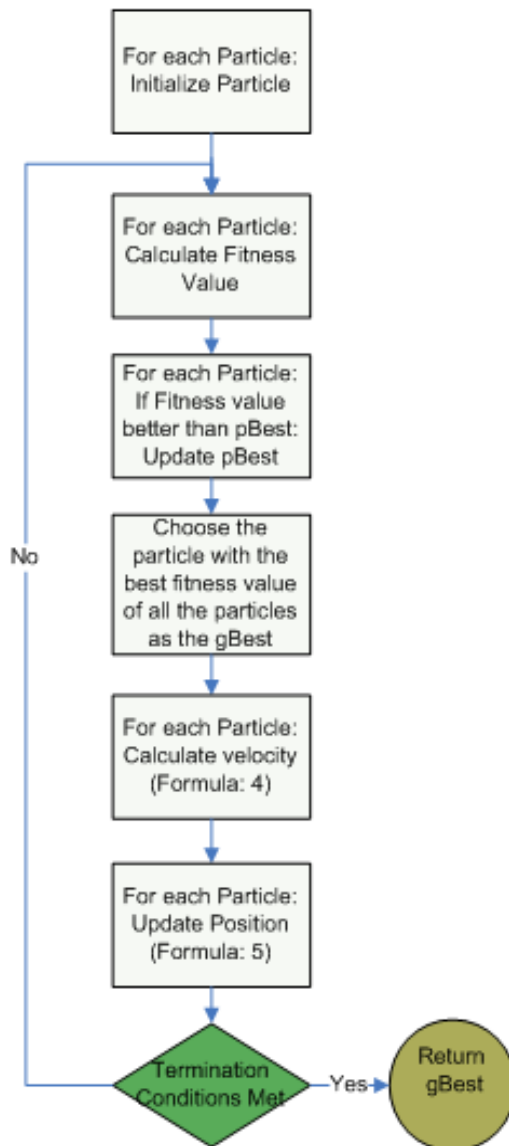


Figure 10. Algorithm: Particle Swarm Optimization.

A.14. Fuzzy Logic

Fuzzy logic is a way of reasoning that presents results in an approximate way, rather than a fixed or exact value. It differs from traditional logic, because in traditional logic (binary), only two values are possible – true or false. However, in Fuzzy logic, the results are expressed in a “truth value” that varies (progressively) in the range between 0.0 and 1.0. Thus, this Fuzzy value should be taken as a degree of truth.

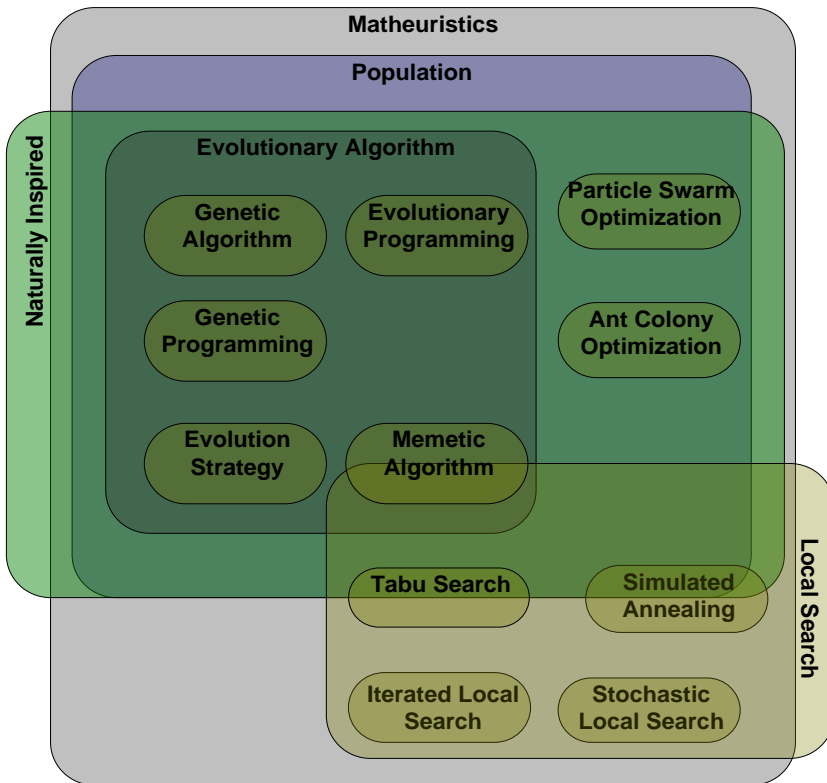


Figure 11. Classification of Metaheuristic Methods.

Probabilistic Logic and Fuzzy Logic may be confused, as they share some similarities (both have truth values varying between 0.0 and 1.0), but are conceptually distinct, owing to dissimilar interpretations and mathematical handling. In Fuzzy logic, a given result corresponds to a certain "degree of truth", while in probabilistic logic, this corresponds to a "probability or likelihood".

Because they are different, Fuzzy Logic and Probabilistic Logic are different models that can be used to process the same real-world circumstances.

Therefore, Fuzzy Logic is ideal for processing incomplete data and presenting approximate resolutions to problems that are too complex to solve using alternative techniques.

This description establishes Fuzzy logic as an interesting idea to explore in the Fields of Financial Computing.

In Financial Computing, the vagueness is the ruler, so instead of providing an absolute answer to the user, like “Buy” or “Sell”, it would be more advisable to deliver a reply in terms of “degree of truth”, such as “may buy”, “may sell”, or “definitely sell”.

Even though the use of Fuzzy Logic might be an especially interesting approach to be used in Financial Computing, in recent work, essays using this technique are very rare.

A.15. Agents

The idea of a software agent is a way to describe a software entity that is capable of acting with some degree of autonomy to accomplish certain tasks. A software agent should be autonomous, which means that agents should be able to do some decision-making without human intervention, like task selection and prioritization.

What distinguishes agents from an arbitrary program is that agents react to the environment, have autonomy, are goal-orientated, and are persistent in their goals.

Therefore, an agent system should exhibit some features of Artificial Intelligence (such as learning and thinking) and should be autonomous.

A system can also be built around multi-agent systems; in this case, we have distributed agents that aren't able to achieve an objective alone, and must therefore communicate and collaborate together to reach their goals.

Figure 12 presents a diagram depicting the main characteristics that an agent should have, as well as its classification according to them.

As in object-oriented programming, an agent is also an abstraction or a concept. The concept of an agent provides a powerful way to describe a software entity capable of acting with some degree of autonomy to accomplish the tasks for which it has been conceived. Contrasting with objects, which are defined in terms of methods and attributes, an agent is defined in terms of its behavior.

In Financial Computing, the idea of software agents has been considered as a motivating idea and an interesting abstraction to explore; some examples are the

papers of Cyril Schoreels et al. [41] to [44], where the agent is a chromosome trained to trade securities using TI's to make its choices.

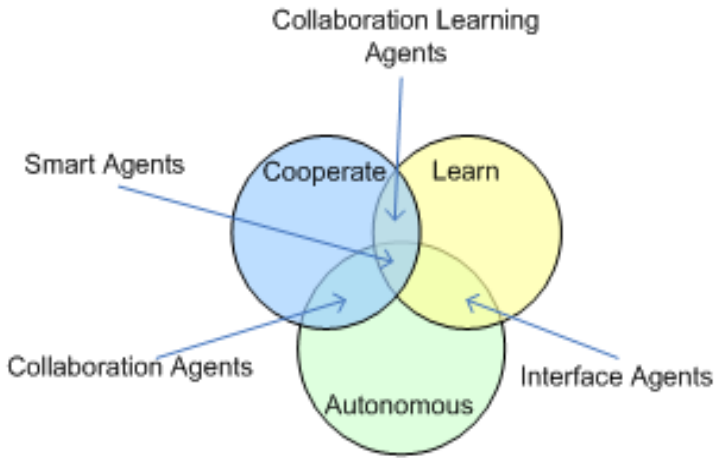


Figure 12. Classification of Agents.

The study of H. Subramanian et al. [45] also presented agents that were based on a combination of trading rules, trained by GA and GP; then, the agents' proficiency was evaluated by making them compete against other programmed agents in the Pen-Lehman Automated Trading Project (PLAT) [46, 47]. Additionally, there was also R. Fukumoto et al.'s study [48] that used a GA-based Multi-Objective Optimization approach to train intraday-trading agents for two objective purposes (profit and variance of the profit); here, the agents were tested in the U-Mart [49, 50], an artificial market simulator.

A.16. Support Vector Machine (SVM)

The Support Vector Machine (SVM) [51] is a model employed to name a series of supervised learning techniques used to recognize patterns and analyze data. Methods of classification and regression from statistics are used. The goal of SVM is to take a set of input data and predict, for each given input, to which of two possible classes the input belongs. Therefore, an SVM system is primarily a classifier that categorizes the inputs into two possible outputs. Only two output values are possible. Although SVM has its roots in statistics, it is a non-probabilistic binary linear classifier.

In an SVM model, the examples are represented as points in space; these points should be mapped so that the examples belonging to the different categories should be divided by a clear gap that is as broad as possible. The new examples are then mapped into the same space and are predicted to belong to either one, or to another category, based on the side of the gap to which they are closer. For a two-dimensional problem, Figure 13 depicts how SVM categorizes the inputs.

The goal is to separate and classify the white dots from the black ones. The full line represents the line that best divides the space between the dots. This line represents the most reasonable choice, as it is the best beeline that corresponds to the largest division, or has the bigger border, between the two classes of dots. Consequently, the choice for the line that maximizes the distance from it to the nearest dots on each side is the preferred choice. This is called the “maximum-margin line” and the linear classifier it defines is known as the “maximum margin classifier”.

The case illustrated in the figure is for a two-dimensional problem, but the problem is studied and solved for the generalized case of n dimensions. In this case, the line that separates the dots is a hyperplane.

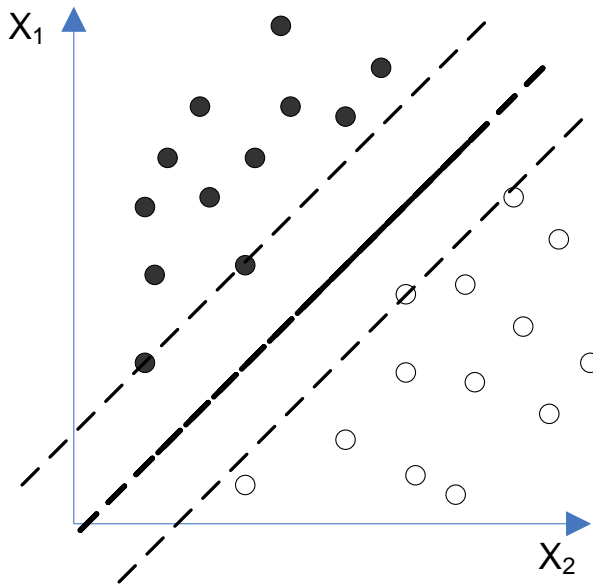


Figure 13. Illustration of How SVM Classify Inputs.

The example used to illustrate the work was for a case where a linear classification was possible, meaning that it was possible to separate the samples using a beeline or a hyperplane. Nevertheless, there are situations where this is not possible, or if used, the classification would be erroneous for some samples. Therefore, a nonlinear transformation can be applied on the dot points, allowing a linear classification to work. For the two-dimensional example, the beeline separating the dots is replaced by a bent line.

Initially, SVM was developed to resolve pattern identification problems with limited applicability in Financial Computing. However, with the introduction of “Vapnik’s ϵ -insensitive loss function” (Vapnik’s ILF) [52], SVM has been enhanced to resolve nonlinear regression inference problems and has been used with good results in financial times-series prediction [53]. In particular, in [54], SVM was applied to perform portfolio optimization, and in [55], it was utilized to make stock tendency forecasts. In [56], SVM was compared to GP, which demonstrated that GP is superior for Financial Portfolio Optimization.

A.17. Other: (Classification Systems, Guided Local Search, GRASP)

In optimization, or in automated problem solving in general, there are many other proposed approaches. This is illustrated by tools like “Scatter Search”, “Variable Neighbourhood Search”, “Guided Local Search”, or “GRASP”.

Applied to Financial Computing, there are also several other computational techniques available, some of which have been applied with good results. Examples of these are some existing optimization algorithms, namely segmentation or clustering, machine learning, classification [57] (including decision trees, K-Nearest Neighbor [57]), and statistical analysis.

A.18. Conclusion

In this chapter section, a review of the main advances found in the literature concerning combinatorial optimization and problem-solving techniques was offered. This section started with a brief review of some traditional techniques, and then went on to talk about some of the relatively recent ones. Some of the discussed concepts were Newton’s Method, Exhaustive Search, Random Search, Quadratic Programming, Expert Systems, and Artificial Neuronal Networks, followed by a discussion of some of the Metaheuristic Methods, as well as Fuzzy Logic, Agents, and Support vector machines.

B. Part B. A Review of Main Multi-Objective Optimization Techniques

In optimization (or, in the general case, in the search for the best possible solution to a given problem), it is found that many real-world problems are not limited to a single objective. There are many situations where a set of objectives (or trade-offs) has to be balanced among the multiple and interacting objectives. In addition, it is not uncommon to have situations where these objectives are conflicting. To deal with this kind of problem, an alternative methodology has been developed, called Multi-Objective Optimization [18].

B.1. Introduction

Multi-Objective Optimization is the process of finding a set of solutions that optimizes several objectives. The notion of an optimum solution is different in Multi-Objective problems from what is usually used in single-objective problems, since instead of reaching a single global optimum (or solution), a set of solutions or a trade-off is achieved.

In this kind of problem, it is not always possible to say when one solution is better than another. It is possible to say if one solution might be better at one specification and if another solution is better at another objective. However, a matter arises: How can that be done for many solutions and when there are many objectives? To help us in our reasoning, a number of concepts are commonly used:

- One solution *dominates* another if it is not worse than the second solution with respect to all objectives and, at the same time, is better than the second solution, at least for one objective. It is important to highlight that the domination relation is not a concept of ordering (sorting) and two solutions can be mutually non-dominant if neither dominates the other.
- The set of solutions that are not dominated by any of the other solutions is called the *Pareto Frontier* (other alternative names are sometimes found, which have the same meaning, such as *Pareto Set* or *Pareto Front*). This set of solutions ultimately represents the best set of solutions that best addresses all the trade-offs involved in the problem.

Therefore, and in conclusion, in Multi-Objective Optimization problems, the optimum solution is the set of all not dominated solutions. A not dominated solution is called a Pareto point and the set of all Pareto points is called the Pareto

Front (this represents the optimal set of trade-off solutions). Selecting a given solution on the Pareto Front always implies that selecting any one of them in place of another will sacrifice the quality for at least one objective, while improving the quality of at least one other objective.

Figure 14 represents an example of a Pareto Front in a two-dimensional space, for an optimization problem intended to minimize the two cost functions.

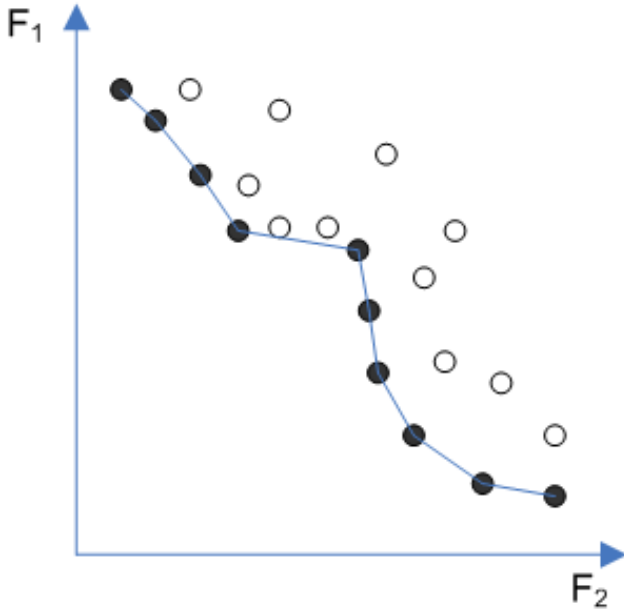


Figure 14. Example of a Pareto Front for a 2-D Space (2 Objectives).

The black dot points belong to the Pareto Front and represent the best trade-offs between the two objectives, while the white dots represent dominated solutions (dominated solutions are solutions that are dominated by some Pareto point).

The decision space consists of all possible values that the decision variables can have in order to attain the best possible outcome. Therefore, the formulation of this type of problem can be formulated as shown in Equation 6 and Equation 7.

Equation 6. Multi-objective Formulation:

$$\text{minimize } \vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})] \quad (6)$$

Subject to:

Equation 7. Multi-objective Constraint Formulation:

$$g_j(\bar{x}) \leq 0, \quad j = 1, 2, \dots, m. \quad h_k(\bar{x}) = 0, \quad k = 1, 2, \dots, p \quad (7)$$

where \bar{x} is a vector including the decision variables, f_1, f_2, \dots, f_k functions map the decision space to and represent all the objective functions, while g_i and h_k also map and symbolize the constraint functions of the problem ($j = 1, 2, \dots, m$), ($k = 1, 2, \dots, p$).

In Multi-Objective Optimization, the number of objective functions is equal to the number of specifications or goals that must be addressed and optimized in a particular problem.

Almost any of the optimization techniques previously exposed in Part A of this chapter can be used to perform Multi-Objective Optimization. The most frequently used method to adapt any of these techniques to do Multi-Objective Optimization is the use of a weighted combination of the objectives, but other variations are possible with the underlying optimization technique. Examples of the use of SA techniques to perform Multi-Objective Optimization can be found in [59] and [60].

Nevertheless, any of the population-based methods seem to be more adequate (see: subsection A.13. Outline of the Metaheuristic Methods and Figure 11. Classification of Metaheuristic Methods) to perform Multi-Objective Optimization [18], because they, in a single iteration, develop a complete set of solutions (population). Therefore, in the following subsections, our attention will be focused on the Population Based Multi-Objective Optimization methods, namely on Multi-Objective Optimization Using Evolutionary Algorithms.

B.2. Main Multi-Objective Optimization Algorithms Using EA

In the following subsections, a brief review of the main approaches found in relatively recent publications used to deal with Multi-Objective Optimization using Evolutionary based Algorithms will be conducted.

B.2.1. Vector Evaluated Genetic Algorithm (VEGA)

The VEGA algorithm is clearly an adaptation of a single goal GA to perform Multi-Objective Optimization. The original proposal appeared in [61] as an extension of a simple genetic algorithm to handle multiple objectives in a single run. According to

the original description, the whole population is randomly divided in m ($m =$ number of objectives) subpopulations and each subpopulation is evaluated and probabilistically selected based on one of the optimization objectives. After the probabilistic selection, the selected individuals are mixed up and pooled together to form the antecessors of the next generation. The process continues with the trivial crossover and mutation operations on the population and repeats until the termination condition is met. This process is illustrated in Figure 15.

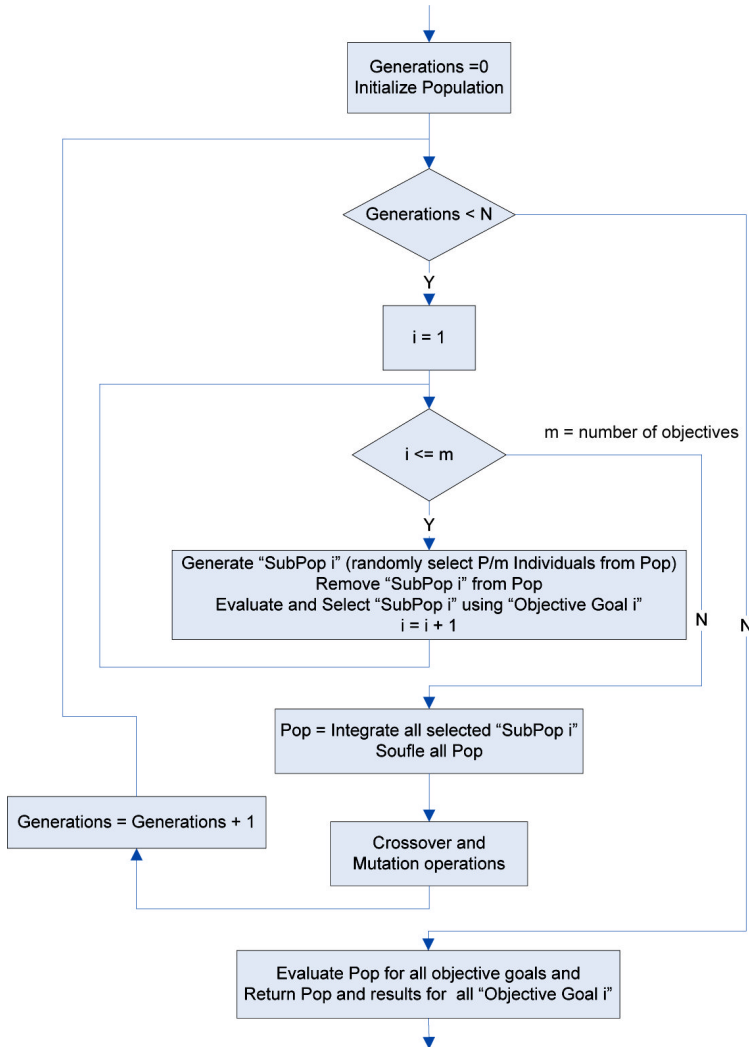


Figure 15. Algorithm: Vector Evaluated Genetic Algorithm (VEGA).

VEGA tends to converge towards one objective best solution [62] with the pool of the final solutions clustering near single-objective best solutions instead of spreading along the Pareto frontier. This is quite unsuited for Multi-Objective Optimization, as the desire is to get the set of solutions that best balance the trade-offs between the objectives.

B.2.2. Multi-Objective Genetic Algorithm (MOGA)

MOGA uses the concept of ranking and assigns the smallest ranking value to all of the non-dominated solutions [63]. The remaining solutions (dominated) are assigned rankings based on how many individuals in the population dominate them.

Consequently, the fitness calculation starts with a value that is an inverse function of the Pareto Rank (Fit). In order to distribute the population of solutions uniformly along the Pareto frontier, the overall (final, global) fitness (Sharing Fit (SF)) value is adjusted by the sum of the "sharing distance" (SD). The sharing distance is inversely proportional to the metric distance between individuals in the objective domain.

The overall fitness value is calculated as follows:

Equation 8. MOGA overall fitness calculation:

$$SF(i) = \frac{Fit(i)}{\sum_j SD(i, j)} \quad (8)$$

where SF(i) is the overall fitness value of individual i, Fit(i) is the inverse of the Pareto Rank of individual i (typically: $i/\text{rank}(i)$). The overall fitness value obtained this way is used in the comparative and probabilistic selection of the individuals. The traditional MOGA implementation uses a roulette selection method. For each round, this algorithm needs to calculate all the Pareto ranks (for all the individuals in the population) and its sharing distance.

B.2.3. Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA 2 is an improvement of a previous implementation (SPEA) [64], although both implementations exploit two populations. One (P) is the population and the other (P') is the archive. On the first implementation (SPEA), all non-dominated individuals in P are copied to the archive (P'); the size of P' was made variable from generation to generation. On the other hand, in SPEA2, the size of P' was fixed and if the set of non-dominated individuals in P exceeds the capacity of P',

then they are truncated. Conversely, if they are less than P' , some dominated individuals are added to the archive P' . The above process of truncation and the addition of individuals both incorporate a strategy to distribute the solutions along the Pareto Front.

In the first generation, SPEA2 selects all non-dominated individuals for reproduction from the Population P . In the subsequent generations, the individuals are selected from the combined population P and the archive P' . Furthermore, SPEA2 always uses a deterministic selection method.

If the non-dominated individuals exceed the fixed size of P' , then the excess individuals are selected (for exclusion) based on the density estimation ($D(i)$). When the non-dominated individuals are too few in quantity to fill P' , then individuals from the next best Pareto Front are selected (for inclusion), this is done until the archive is filled. When the last Pareto Front surpasses the capacity of P' , the same truncation method is employed. The density estimation of an individual ($D(i)$) is calculated as follows:

Equation 9. SPEA2 Density estimation:

$$D(i) = \frac{1}{d(i) + 2} \quad (9)$$

where $d(i)$ is the distance of individual i from the nearest neighbor.

B.2.4. Non-Dominated Sporting Genetic Algorithm 2 (NSGAI)

NSGA2 is an improvement of a previous implementation from the same authors, Deb. et al. [17], [18] and [16]. The second version of the proposed algorithm also uses two populations of the same size: P for the parent population and P' for the offspring population. The two populations are combined and shuffled together before the selection process; in the selection process, the algorithm first selects the individuals that have a smaller Pareto rank (the non-dominated ones). In the last Pareto rank, the remaining individuals (required to fill up the selected population of P individuals) are selected based on the calculated Crowding Distance and the individuals far away from others are preferred for selection and reproduction.

The authors proposed the second version of their algorithm due to some comments criticizing that it was too complex (in terms of number of computations) and slow. In the second version, the authors demonstrated that it was competitive with the existing state-of-the-art solutions.

B.3. Conclusion

In [61], a comparative study of the existing Multi-Objective Optimization approaches was made. In this study, a concrete and real application to perform Multi-Objective portfolio optimization was used to test and benchmark the available approaches. To evaluate the quality of the approaches, metrics like the spread of the solutions across the Pareto frontier, the closeness of the solutions found (PF known), and the best-known Pareto frontier (PF true) are used. In the conclusion, the authors of this paper claim that the SPEA 2 is among the best Multi-Objective Evolutionary Algorithms (MOEA). They also conclude that SPEA2 is applicable for realistic portfolio optimization with real constraints.

A complete and comprehensive review of the existing solutions for Portfolio Management using Multi-Objective Evolutionary Algorithms can be found in the study of K. Metaxiotis [64]. In this same study, some clues are revealed concerning why there are a limited number of papers discussing the theme of Multi-Objective Portfolio Management.

C. Part C. A Case Study

In this part of the chapter, a Genetic Algorithm (GA) based on a Multi-Objective Evolutionary System to optimize a Trading or Investment Strategy (TS) is developed. Two conflicting objectives are set to be optimized: Maximize the Returns and minimize the risk. However, first, fair and established metrics should be set to be used to both evaluate the returns and the linked risk of the optimized TS. Then, these TS will be evaluated in several markets using data from the main stock indexes of the most developed economies, such as NASDAQ, S&P 500, FTSE 100, DAX 30, and the NIKKEI 225. Finally, the results are presented, where the Pareto Fronts obtained with the training data during the experiments clearly show the inherent trade-off between risk and returns.

This part will continue with a brief introduction to the problem; then a state of the art will be offered. This will be followed by the methodology explanation, and then, the results and the conclusions will be presented.

C.1. Introduction

In artificial intelligence [65], Genetic Algorithms (GAs) are a family of computational techniques that apply the Darwinian theory of evolution to develop

and optimize a possible solution to a given problem. These algorithms encode a probable problem solution on a data structure and apply selection techniques (survival of the fittest) and recombination operators (crossover and mutation) to these data structures. These GA machine learning techniques begin with a set of potential solutions (population) to the problem and are used to optimize this population according to a fitness function that evaluates the solutions according to their ability to perform or solve the specified task. Genetic algorithms are often viewed as function optimizers, although the variety of problems to which genetic algorithms can be applied is fairly wide.

Besides some unfavorable judgments [66], [67], Technical Indicators (TI) are still widely used as tools to perform the technical analysis of financial markets, exploiting the existence of trends to establish potential buy, sell, or hold conditions. Although S.B. Achelis [68] has made a complete reference that fully explains the most important TI's that one can identify and use, this study is still very tricky. Aside from that, the main difficulty of TI usage is deciding its suitable parameter values. In financial practice, it is not uncommon to see analysts conduct extensive manual analysis of historically well-performing indicators, looking for hidden interactions among variables that perform well in combination. When a person finds one of these interactions, he/she keeps it as a personal secret.

Thus, evolutionary computation appears to be a highly suitable alternative to extend the technical analysis of financial markets to tune the parameters of some chosen TI (or set of TI's), so that the desired goals are achieved to the maximum extent possible. In this environment, what the system should do can be viewed as some form of predicting future stock prices. Consequently, in this context, evolutionary computation emerges as a stochastic search technique able to deal with highly complicated and non-linear search spaces.

In the last decade, several financial crises have occurred that have had extensive consequences on the financial assets valorization, which has warned investors that risk should also be taken into consideration when making any decision. This situation was the principal motivation for this study: tune an Investment or Trading Strategy (TS) to be able to achieve the highest returns with minimal risk. The simultaneous achievement of both these goals is supposed to correspond to obtaining solutions that are more robust.

The goals of this specific study are to tune a TS that is able to present the highest returns as existing single objective based approaches, and concurrently reduce risk by using a multi-objective evolutionary optimization approach.

Consequently, two objectives are set: the maximization of the Return on Investment (ROI), and the minimization of the related risk. The proposed framework is tested using data from the main stock indexes of the most developed

economies, such as NASDAQ, S&P500, FTSE100, DAX30, and the NIKKEI 225. The results are presented and some possible conclusions are outlined.

The next section will present the related work on Genetic Algorithms applied to Financial Markets. Section 3 explains the system architecture, defining the roles of the most relevant modules used to build the proposed framework; the chromosome encoding is also outlined. The TI adopted as the core building block of the evolved TS used in this work is also presented in this section. Section 4 presents the results and the most relevant outcomes are highlighted. Finally, the conclusions of this study are presented in Section 5.

C.2. Related Work

Stock market analysis has been one of the most attractive and active research fields where many Machine learning techniques are adopted. Generally speaking, one can distinguish two methods for anticipating future stock prices and the time to buy or sell; one is Technical Analysis [69] and the other is Fundamental Analysis [70]. Fundamental Analysis looks at stock prices using the financial statements of each company, economic trends, and so on; it requires a large set of financial and accounting data, which is difficult to obtain and is both released with some delay and often suffers from low consistency. Technical Analysis numerically analyzes the past movement of stock prices, is based on the use of technical stock market indicators that work on a series of data, usually stock prices or volume [68]. Consequently, this work will be focused on the use of Technical Analysis to anticipate future stock price movements.

One of the earliest proposals where genetic programming was applied to generate technical trading rules in the stock market was published by Allen et al. in 1995 [71] and in 1999 [72]. Later, many approaches based on evolutionary computation were proposed and applied to diverse fields of financial management to predict worth trends. Financial market prediction has been the subject of many studies, and in recent years, a combination of algorithms and methods has been extensively used. Table 1 summarizes some of the relatively recent approaches found in the vast available literature.

In an effort to summarize, in most of the below works, the generated returns are exclusively used as the only fitness metric, without accounting for the related risk. Some examples are the use of GAs to optimize TI's parameters, such as in Fernández-Blanco et al.'s [21] study, or to develop TS based on Tis, such as is found in Bodas-Sagi et al. [22], Gorgulho et al. [23], and Yan et al. [80].

According to what was stated for the first time in 1952 by Markowitz [81], any TS should have the highest possible profit with the minimal risk. Unfortunately, these two metrics are intrinsically conflicting by virtue of the risk-return tradeoff. Some previous articles proposed the combination of the two conflicting objectives into one single metric, particularly the proposals of Bodas-Sagi et al. [22], which used the Chicago Board Options Exchange (CBOE) Volatility Index (VIX) [82] and [83], as a metric for risk. Schoreels et al. [44] proposed the use of a Capital Asset Pricing Model (CAPM) [84] system, based on Markowitz's [81], portfolio theory to reduce risk through the balanced selection of securities. More recently, Pinto et al. [85] proposed and studied several alternatives to the classical fitness evaluation functions.

In terms of real Multi-Objective Optimization, some studies can be found, such as the paper of Ghada Hassan et al. [86] where a Multi-Objective system to maximize return as the annualized average of the returns, and minimize risk, as the standard deviation of the annualized average of the returns, was presented. In this same study, Genetic Programming (GP) was used to model equations that combine the time-series input data to score a given stock. Additionally, low-frequency trading was used, as the training data consisted of monthly data.

Again, in 2009, S.C. Chiam et al. [87] used a Multi-Objective system to maximize the total returns and to minimize the risk or the exposure to it. The proposed framework is tested using data acquired from only one stock market – the Singapore Exchange (Straits Times Index (STI)). Hence, some of the conclusions drawn from this study could be attributed to the market used for the test (some odd peculiarities exhibited by this market); additionally, the metric used to evaluate the return is particularly unusual, therefore making it difficult to compare the presented results with the results of other alternative applications.

The goal of this paper is to tune a TS using a Multi-Objective GA. In doing this, solutions that are more robust should be developed, and, consequently, the results in the out-of-sample period should be improved, while risk is supposed to be simultaneously minimized.

C.3. Methodology

The proposed system consists of a Multi-Objective Genetic Algorithm coupled with a market return evaluation module that performs the fitness evaluation. The fitness evaluation is performed based on the estimation of the two conflicting objectives, on the chosen markets, and on the specified time frames.

Table 1. Investment Approaches

Author(s)	Reference	Year	Methodology	Heuristics	Metrics	Risk Aware	Results Evaluation	Kind of Data Used	Data From	Period	Performance	Annualized Return (Average)	Investment Strategy
Wuthrich et al.	[73]	1998	k-NN and NN	Key words found on financial newspapers published on the Web	ROI	No	B&H & Mutual Funds	News	DJI, Nikkei, FTSE, HSI, STI	6 Dec 1997 To 6 Mar-1998	14% to 30 %	20.8%	Long and Short
D. J. Bodas-Sagi, et al.	[22]	2009	EA	RSI, MACD	ROI, #Transac., Risk	Trend Risk, VIX	B&H – Use of the TI alone	Price	Dow Jones ETFS Brent Oil	2000 to 2005	Sliding Window	--	--
Garcia et al.	[74]	2010	GP and GA (GAP)	Decision Tree, based on TI (MA, EMA, ROC, RSI, SO)	Mean Accumulated Return	Sharpe ratio	B&H and other	Price	S&P500 index	1988 to 2005	Mean Accumulated Return of 37.02%	--	Long
Butler et al.	[38]	2010	PSO	BB, MA, EMA	ROI, Sharp, Sortino, Accuracy	Sharp Ratio, Sortino Ratio, Accuracy	B&H and BB with typical parameters	Price	DJI	1990 to 2004	61.5 to 101.44 (return in 5 years)	-7.7% to 0.28%	Long and Short
Vincent Cho	[75]	2010	ARIMA, NN, regression	Several	Accuracy and ROI	No	Return	Several	HSI, DJI, and other data	2003 to 2007	Sharp ratio from 0.127 to 3.616	4.4% to 19.6%	Long

Table 1. Continued

Author(s)	Reference	Year	Methodology	Heuristics	Metrics	Risk Aware	Results Evaluation	Kind of Data Used	Data From	Period	Performance	Annualized Return (Average)	Investment Strategy
Gorgulho et al.	[76]	2011	GA	EMA, HMA, ROC, RSI, MACD, TSI, OBV	ROI	No	B&H and random strategy	Price and Volume	30 stocks from the DJI	01/01/03 to 31/06/09	16.68% to 25.29%	2.23% to 3.27%	Long and Short
Lei Wang et al.	[77]	2011	HLP-NN	Unknown	Prediction error	No	Prediction error	Price	Shanghai Index	11-18-1991 to 2-10-2009	7.16% - 9.65% (Prediction error)	--	Long
Kyoung-jae Kim et al.	[78]	2012	Uses GA to optimize NN	Selected features (Several TI)	Prediction Accuracy	No	Prediction Accuracy	Price	Korea Composite Stock Price Index (KOSPI).	January 1989 to December 1998	50.6% to 66.1% Prediction Accuracy	--	--
Canelas et al.	[79]	2013	GA	Pattern discovery	ROI	No	B&H and other	Price	99 stocks from the S&P500 index	1998 to 2004	36.4 % to 62.8% (Average ROI)	6.00% to 9.57%	Long and Short

C.3.1. Strategy and Parameters

The task of specifying buy and sell conditions for long and short positions means describing the TS. Putting this together with an optimization engine allows for the automatic exploration of trading strategies according to a specified criterion, which is evaluated and described by the given fitness functions.

The TS evolved in this study is based on the use of a TI. The elected TI to be used in this study is the Moving Average Crossover (MAC), which in turn is based on the use of two Moving Averages (MA) with different periods. One of the MA is formed by the shorter of the two periods and is called the "Fast MA". The other, formed by the one with the longer period, is the "Slow MA". The "Fast MA" reflects changes earlier than the "Slow MA". A buying (or sell short) signal is generated when the Fast MA crosses over the Slow MA.

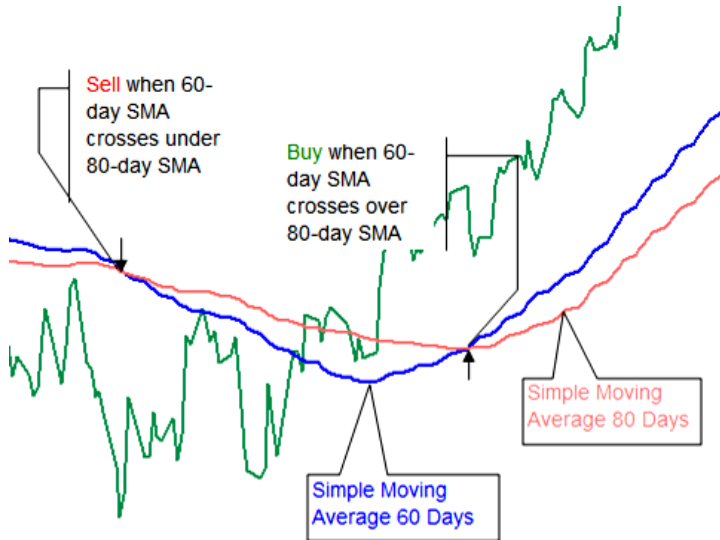


Figure 16. Illustration of the MAC operation.

Conversely, a sell (or a buy short) signal is generated when the Fast MA crosses under the Slow MA. This process is illustrated in Figure 16.

After defining the strategy, it is necessary to define the parameters of the MAC, which in this case are the type of the MAs and the corresponding period. These two parameters represent the variables of the underlying strategy. It is also important to stress that, for the type of MA to use, the GA has the freedom to choose between a Simple or an Exponential MA.

Although it is common to tune the parameters of a single TI and then use it to generate buy and sell signals for both long and short positions, in this work, the option of using a separate set of parameters for each of the possible actions was taken, more specifically, "enter long", "exit long", "enter short", and "exit short" were used.

It is also important to note that some preprocessing of the historical data was done. This applies, for instance, to the MA periods, which are calculated at the program start, and are limited to the following set of Simple or Exponential MA's: 1¹, 4, 8, 12, 14, 16, 20, 24, 28, 32, 36, 40, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, and 250 days.²

C.3.2. Genetic Encoding

The chromosome structure must represent the MAC indicator used; this way, one MAC chromosome is represented by two genes: one represents the type and the period of the Fast MA and the other does the same for the Slow. These entries are natural numbers in the interval of values between 0 and 65, as it encodes, in a single entry (integer variable), the type of MA and its period.

Therefore, there are four possible actions, and a set of MAC parameters is used for each of these possible actions, which implies that a total of eight parameters or genes must be represented in the chromosome structure. In Table 2, the chromosome structure is summarized.

C.3.3. Fitness Evaluation

The fitness evaluation process is concerned with simulating the performance of each trading agent in the evolving population and calculating the corresponding total returns and the related risk. The resultant fitness values of the trading agent must be evaluated under some established and fair metric, as will be discussed in the following subsections.

C.3.4. Return Metric

The profits generated by any given TS can be measured in different ways, as will be discussed below.

¹ In reality, the MA of 1 (one) day, is not an MA, but the day security price; this is a trick to allow the GA to choose between one of the available MAs or the actual quote.

² This set of periods has been chosen because it covers the most widely used long and short-term MA periods found in books and recommended by experts [68].

For instance, the potential profits can be estimated by simply summing the area under the total asset graph during the trading period [42]. Alternatively, another return metric could be the total final assets; this means the available capital, plus the value of all holdings, at the end of the investment period [88] and [19]. Unfortunately, both preceding metrics have the obstacle of always being attached to the initial cash invested.

Table 2. Chromosome representation

Parameters	Enter long position		Exit long position		Exit short position		Enter short position	
	Fast MA	Slow MA	Fast MA	Slow MA	Fast MA	Slow MA	Fast MA	Slow MA
Chromosome	0..65	0..65	0..65	0..65	0..65	0..65	0..65	0..65

Therefore, an alternative metric exists that, instead of considering the absolute value of the holdings, rather consider its relative value. This metric is a ratio and is called the Rate of Return (ROR), also known as the Return on Investment (ROI), rate of profit, or sometimes just return. This ratio represents the money gained or lost (whether realized or unrealized) on an investment relative to the amount of money invested. ROI is usually expressed as a percentage, and for one period of time, by definition, is calculated by **Error! Reference source not found.** In this equation, “Profit” is the amount of money gained or lost and is sometimes referred to as interest, gain/loss, or net income/loss; “Initial Investment” is the money invested, and may, alternatively, be called the asset or capital.

Equation 10. ROI Calculation:

$$\begin{aligned}
 ROI &= \frac{\textit{Profit}}{\textit{Initial Investment}} \\
 &= \frac{\textit{Final Assets} - \textit{Initial Investment}}{\textit{Initial Investment}} \\
 &= \frac{\textit{Final Assets}}{\textit{Initial Investment}} - 1
 \end{aligned} \tag{10}$$

ROI still has the problem that, for multi-period investments, it is difficult to compare it with the results one would get in a single period of time. Therefore, a

metric that could be compared with similar alternative investments, like investment funds or bank deposits, should be used instead. This way, in this study, the Annualized ROI will be used. The Annualized ROI is nothing more than the “Geometric Average of the Ratio of the Returns”, also known as the “True Time-Weighted Rate of Return”. Mathematically, an investment lasting for N periods with full reinvestment is computed by Equation 11:

Equation 11 Annualised ROI calculation:

$$\text{Annualised}(ROI) = \sqrt[N]{(ROI + 1)} - 1 \quad (11)$$

In this equation, N is the number of periods, or more specifically, the number of years that the investment lasts.

C.3.5. Risk Metrics

Risk is usually seen as the volatility or the uncertainty of the expected returns over the investment period. Therefore, the linked risk of any investment technique can be estimated in several ways, as will be examined in the following section.

The most traditional risk metric is inherited from statistics and from the Markowitz Mean-Variance Model [81]. It consists of the use of the results variance as a metric for the risk. This variance can be calculated using the standard deviation or the variance between the returns; in finance, this statistical measure of the dispersion of the results is usually called volatility.

Instead, risk can also be computed as the exposure to it [89]. Specifically, it can be measured by the proportion of trading days when a position is maintained open on the market, and is, mathematically, the ratio between the time the agent is on the market and the total available trading time (Equation 12). Essentially, staying in the market longer corresponds to a higher exposure to risk, such as market crashes and other disastrous events, while shorter periods in the market correspond to lower risk exposure and greater liquidity (as the capital is engaged for a smaller time). In Equation 12, $t_{i, \text{exit}}$ and $t_{i, \text{entry}}$ denote the time at which the trading agent enters and exits the market, for each i^{th} trade, and T refers to the total length of the trading period.

Equation 12. Risk Exposure calculation:

$$\text{Risk Exposure} = \frac{1}{T} \sum_{i=1}^n (t_{i, \text{exit}} - t_{i, \text{entry}}) \quad (12)$$

Other alternative metrics have been proposed and used, as, for instance, the use of Maximum Drawdown (MDD) as an estimate for risk. The Drawdown (DD) [90] calculates the decline from a past historical peak in our variable (the evaluation of the total assets) to its current value. The DD can be calculated in terms of absolute or relative values. In the next pseudo code, how the DD and the corresponding Maximum Drawdown (MDD) are calculated, in terms of relative values are presented:

```
MDD = 0
peak = -inf
for i = 1; i < N; ++I do
  if (assets[i] > peak) then
    peak = assets[i]
    DD[i] = 0
  else
    DD[i] = 100.0 * (peak - assets[i]) / peak
  if (DD[i] > MDD) then
    MDD = DD[i]
  End if
End if
End for
```

Additional alternative metrics for risk can be found in the literature, such as the use of some risk-adjusted return metrics, like the Sharpe ratio (also known as the Sharpe index), Sortino ratio, Sterling ratio (SR), Calmar ratio (CR), or VIX [82] and [83]. All previous metrics compute the net profitability after discounting the associated risk [91] and [92]. In short, these risk metrics are, in reality, alternative methods to combine the two conflicting objectives faced in this kind of problems (risk and return) into one single objective (metric).

C.3.6. Selection of the Risk Metric to Use

In preparation for this study, some preliminary tests were conducted using several combinations of the previously exposed metrics (return and risk); the next paragraphs summarize the main conclusions drawn from the tests conducted and the results observed.

In the first stage, the metrics that combine the two conflicting objectives (SR, CR, VIX, ...) into one single metric have been discarded, since the goal of this study is to do Multi-Objective Optimization and the use of a metric that is the

aggregation of two conflicting objectives does not make sense. In addition to being unhelpful, this could also correspond to benefitting one of the objectives while damaging another.

Some preliminary tests have been conducted using the MDD as a metric for risk; however, strange behavior in the results was observed. This strange behavior observed in the tests conducted can also be confirmed by the varied results obtained in the study of O'Neill [93]. Recall that MDD records the maximum of the losses in the trading period. Consequently, this metric is not a good representation of the overall performance, since the performance of the TS can be good almost all the time, but, if even for a small period of time, some losses are incurred, this can result in a big value when this metric is used. Although this metric can be a good sentiment, independent, or auxiliary metric to further evaluate a strategy, computationally speaking, as a multi-objective goal, it does not seem to be a good main risk metric.

Before conducting any tests using the Volatility to gauge risk (it can be either the standard deviation or the variance between the returns), it must be decided how the period of study should be subdivided in order to calculate its variance (days, weeks, months, years, etc.). At this point, and due to the lack of additional information, as this detail is usually omitted from most of the available literature, the decision was made to adopt the approach described in Article 46 of reference [94]. Consequently, from now on, in this text, when the term “Variance (of the results)” is used, it refers to the “Annually adjusted standard deviation” of the “actual weekly profits” exactly as described in reference [94]. In Equation 13, the cited formula is recalled.

Equation 13. Annually adjusted standard deviation calculation:

$$Ann\ adj.\ std.\ dev. = \sqrt{\left(\frac{1}{T-1} \sum_{t=1}^t (r_t - \langle r \rangle)^2\right) * \sqrt{52}} \quad (13)$$

where, r_t symbolizes the actual weekly returns or profits in the period t ; T is the number of weeks in the trading period, coinciding with the period used for the calculation of profits, and $\langle r \rangle$ is the simple arithmetic mean of the actual profits in the trading period.

By its turn, the actual weekly profits are calculated by Equation 14.

Equation 14 – Actual Profits calculation:

$$Actual\ profits = \left[\frac{Final\ Assets}{Initial\ Assets} \right] - 1 \quad (14)$$

In this equation, Final Assets represents the total value of the holdings at the end of the reference period, while Initial Assets stand for the value of the holdings at the start of the reference period.

Therefore, it remains to be decided whether to use the Variance (of the results) or the Risk Exposure as the elected risk metric in the remainder of this study.

Consequently, some preliminary tests were conducted, using the weekly variance of the returns and risk exposure, as both exhibited fairly good results using the training data. Afterwards, and in order to have a better understanding of how these two variables relate together, as well as to determine which one could lead to better results, a correlation test between these two apparently uncorrelated variables was conducted.

The method used to make this correlation test was the Monte Carlo method, which consisted of generating 15,000 random TS. Subsequently, the performance of the randomly generated TS are evaluated according to these two objectives in the different markets. With the collected data, graphs like the ones shown in Figure 17 were produced. In this figure, so that any correlation between these two variables can be visually inspected, the Variance of the results is plotted in the X-axis and the Risk Exposure is on the Y-axis. The plots shown in Figure 17 were built using training data from Nikkei and FTSE indexes, but similar graphs were obtained using the other indexes tested in this study.

The main conclusion from the plots below is that these two apparently uncorrelated variables are, in practice, highly linked together. Thus, the choice of any of these risk metrics as the one elected to be used in this study should have minimal influence on the final results. Therefore, the decision was to select the simplest of the formulas, which is risk exposure. Computationally speaking, everyone knows that simpler functions and algorithms should be preferred, as they consume less computer resources, are faster, and less prone to errors (if coded properly).

Hence, in the remainder of this present work, the exposure to the risk will be used as the risk metric, more specifically, the ratio between the number of trading days a position is maintained open on the market and the total available trading days, and is calculated according with Equation 12.

C.3.7. Optimization Kernel

This study is focused on the simultaneous optimization of TS that must achieve two objectives: the maximization of a Return Metric and the minimization of a Risk Metric. Therefore, in this situation, the proposed framework must consider and simultaneously balance the two objectives, as these two objectives interact

between them and are conflicting. To deal with this kind of problem, an Evolutionary Algorithm-based technique has been developed, called Multi-Objective Optimization (MOEA) [18].

Multi-Objective (MO) optimization is the process of finding a set of solutions that optimizes several objectives.

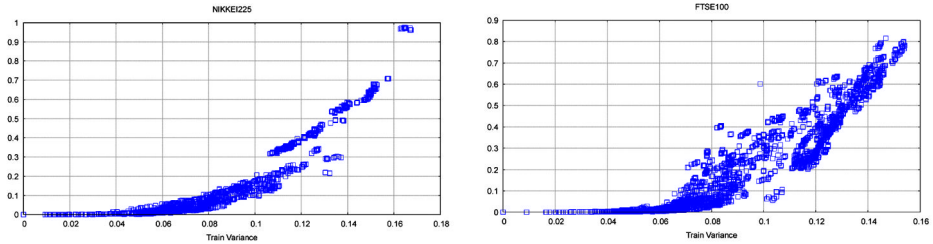


Figure 17. Plots showing the correlation between Risk Exposure and Results Variance.

The notion of an optimum solution is different in MO problems from what is usually used in single-objective problems, since in MO optimization, instead of getting a single global optimum (or solution), a set of solutions or trade-offs is supplied to the user.

In MO optimization, it is not always possible to say when one solution is better than another. It is straightforward to say whether one solution might be better at one specification and if another solution is better at another objective. However, a matter arises: How does one do that for many solutions? To help in the understanding of what follows, some terms of general use in MO optimization should be introduced, as follows:

One solution dominates another if it is not worse than the second in all objectives, and, at the same time, is better than the second in at least one objective. It is important to note that the domination relation is not a concept of ordering (or sorting) and that two solutions can be mutually non-dominating if neither dominates the other. This can be mathematically formulated, for an m objectives minimization problem, as shown in Equation 15 and Equation 16. In the case of Equation 15 and Equation 16 we say that, when both equations are satisfied, that solution x dominates solution y .

Equation 15. MO Formulation:

$$f_i(x) \leq f_i(y), \forall i = 1, 2, \dots, m \quad (15)$$

and:

Equation 16. MO Formulation:

$$\exists i \in \{1, 2, \dots, m\} : f_i(x) < f_i(b) \quad (16)$$

where the f_i functions map the decision space to and represent the objective functions that should be minimised. and are vectors representing all decision variables. The set of solutions that are not dominated by any of the other solutions is called the Pareto Frontier (PF) (or Pareto optimal set or Pareto Front). This set of solutions ultimately represent the best set of solutions that address all the trade-offs considered in the problem.

Hence, and in contrast to single-objective optimization, the optimal solutions to a Multi-Objective Optimization problem exist in the form of a set of solutions (PF); this set is the set of all non-dominated solutions that balance all trade-offs. A given solution belonging to the PF can only have one of its objective components improved by degrading at least one of its other objective components.

In the concrete case of this study, the decision space consists of all possible values that the chromosome parameters can have, in order to find the best possible set of trade-offs (Risk and Return) using training data (from the training period). Therefore, the formulation can be expressed as shown in Equation 17 and Equation 18.

Equation 17. Problem MO Formulation:

$$\begin{cases} \text{maximise } f_1(x) := \textit{Annualised(ROI)} \\ \text{minimise } f_2(x) := \textit{Risk Exposure} \end{cases} \quad (17)$$

Subject to:

Equation 18. Problem MO Formulation:

$$\begin{aligned} g_j(\vec{x}) &\leq 0, \quad j = 1, 2, \dots, m \\ h_k(\vec{x}) &= 0, \quad k = 1, 2, \dots, p \end{aligned} \quad (18)$$

where f_1 represents the objective function linked to return, which should be maximized, according to Equation 11; f_2 does a similar function with respect to the objective function coupled with risk, which should be minimized, and is calculated as shown in Equation 12. In Equation 17, represents the chromosome

parameters as already introduced in Table 2. Finally g_i , h_k map and symbolize the constraint functions of the problem ($j = 1, 2, \dots, m$), ($k = 1, 2, \dots, p$). Some examples of g_i and h_k functions are:

- $x_i \in \mathbb{N}$ (All x_i are natural numbers),
- $0 \leq x_i \leq 65$ (The interval of values is between 0 and 65),
- $x_1 \leq x_2$; and $x_3 \leq x_4$; and $x_5 \leq x_6$; and $x_7 \leq x_8$ (the period of the fast MA cannot be greater than the slow).

Additional problem-specific constraints should be considered. For instance, it does not make sense to have a strategy that could be simultaneously on the market as long and short, and also, one cannot invest more than the available cash, and so on, but these kinds of constraints can only be considered when the simulation is done by the “Investment Simulator”.

The multi-objective fitness evaluation process is concerned with finding the optimal set of trade-offs between the risk metric and the linked return metric for each TS in the evolving population of chromosomes during the set trading period. During the trading period, the performance of each TS is evaluated by simulating its actions of buying or selling the assets; its related score is later calculated.

The Multi-Objective Genetic Algorithm elected to be used in this study was a version of the Non-Dominated Sorting Genetic Algorithm 2 (NSGAI2) [17] and [18]. NSGAI2 parameters are as follows: population size is 500, the crossover probability is fixed at 0.8, with parents selected by tournament selection. Each run on the training data continued for 300 generations and the probability of real mutation was set to 0.1. These parameters were selected based on a series of preliminary investigations and parameter tuning. The NSGAI2 algorithm was the selected evolutionary algorithm because this MOEA is acknowledged as one of the most efficient and most commonly applied algorithms, incorporating several prominent characteristics to speed up the search and the solution space exploration.

C.3.8. The Investment Simulator

The Investment Simulator Module simulates an investment in the user-specified index, including long and short positions. This stock market index can be bought (“go Long”), sold and after stay out of the market (“Stay Out”), or even sold without owning any (“go Short”), hoping to profit from a decline in the price of the assets between the sale and the repurchase.

Since daily data was available, training consisted of formulating a TS, giving the agent some initial cash to spend, and simulating the agent's performance every day. The resulting total assets are calculated by summing the cash plus the evaluation of the assets at the current stock closing price.

The actions of buying or selling are determined by the strategy encoded in the chromosome, when suggested by the indicator to buy or sell, when buying invests all of the capital, and when selling releases all the securities owned (full reinvestment). Securities that are sold or bought are converted into capital at their current closing price.

At the end of the training period, the total assets that the given TS can achieve are evaluated. Transaction costs and dividends were not included in the simulation. The environment is also assumed as discrete and deterministic in a liquid market.

C.4. Results

Since a Multi-Objective Evolutionary Optimization of TS is considered in this essay, the maximization of a Return Metric and the minimization of its related Risk Metric are the goals. In this kind of problem, the optimal solutions exist in the form of a set of trade-offs known as the Pareto-optimal set; any objective belonging to a solution of this set cannot be improved without degrading the other objective. The problem will be directly modeled as a Multi-Objective Optimization problem by simultaneously optimizing returns and risk; an example of a possible PF is illustrated in Figure 18. This figure clearly represents the risk-return tradeoff, the Efficient Frontier, which is always faced in these kinds of problems.

In this figure, each point denotes a Strategy evolved by the GA. The black circles and the white crosses represent non-dominated and dominated solutions, respectively. The set formed by the former solutions is the Pareto optimal solution set because their returns cannot be improved any further without compromising risk. In the context of a single-objective optimization, where the return maximization is the only goal, the evolutionary process will ultimately drive the solutions towards the extreme point B.

This is not applicable to conservative investors, who may prefer a lower level of risk at the cost of lower returns. Point A represents the extreme case of a conservative investor with zero returns due to total risk adversity.

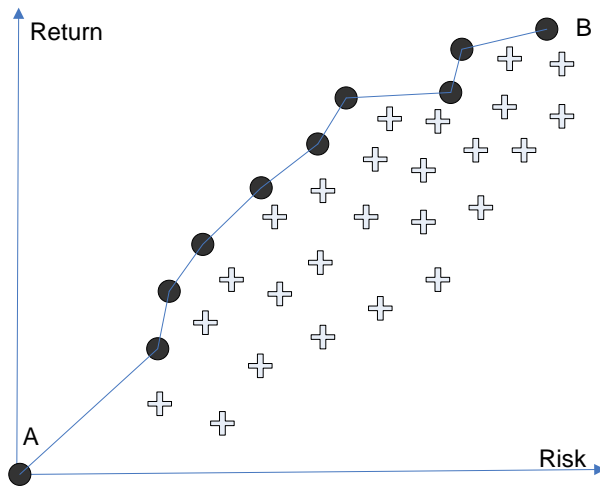


Figure 18. Risk-Return Tradeoff.

C.4.1. Training and Testing Data Sets

Historical daily prices obtained from the finance.yahoo.com database were used. The system was tested with the main stock indexes of the most developed economies, namely the S&P 500 (USA), FTSE 100 (England), DAX 30 (Germany), NIKKEI 225 (Japan), and NASDAQ (USA). Data used in the system covers the time from January 4, 1999 to December 31, 2009, a period of more than 10 years. The period of time chosen for training was from January 3, 2000 to December 31, 2007, consisting of eight years of daily data (about 80% of the data used). This period was assumed to be sufficient to evolve a competitive population, as it exhibited significant movement, including several boom and crash periods. For testing, or validation period, two years of data, from January 2, 2008 to December 31, 2009 (about 20%) were used. Furthermore, it is important to note that 250 days of prior historical data is required before training can be started, in order to calculate and have valid all moving averages.

C.4.2. Analysis of the Performance in the Training Period

Figure 19 presents the PFs evolved for the 5 indexes tested in this study in one of the experimental runs performed. Although the various solutions sets vary in terms of Pareto dominance and optimality, all clearly illustrate the inherent trade-off between return and risk. Furthermore, the evolved TS are able to generate high

returns in open positions in less than 100% of the trading period; for instance, the observable annualized ROI of about 10% with a risk exposure of around 0.6.

In Financial Computing, when analyzing the performance of a given TS, it is common to compare it against the “Buy & Hold” (B&H) and “Sell & Hold” (S&H) strategies. B&H strategy is a long-term strategy that consists of buying the stocks at the beginning of an investment period and holding it for the entire time, regardless of any market fluctuations. S&H strategy does the opposite and consists in selling assets (without owning them) at the start of an investment period (selling short), and repurchasing them at end of the investment period. In the latter case, the profits result from a decline in the assets price between the sale and the repurchase.

When the ROI performance of the evolved TS (see Figure 19) is compared against both B&H and S&H approaches (see B&H and S&H annualized ROI calculation in Table 3) during the training period, it is easy to conclude that, in this context, both B&H and S&H strategies are undoubtedly suboptimal. It is also important to remember that B&H and S&H strategies both correspond to a risk exposure of 1 (one), since the capital is all time engaged.

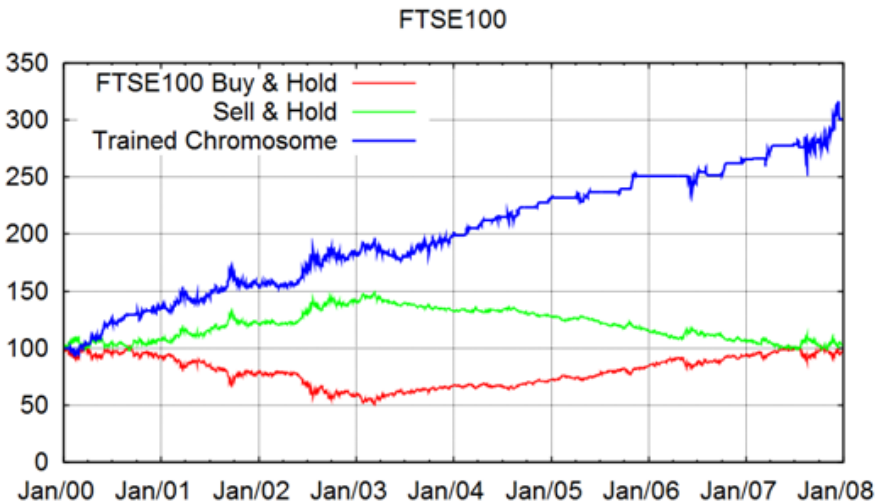


Figure 20 presents an example of the eight-year financial data used to optimize the strategy; in the current case, it is the FTSE100 index. The line labeled “Buy & Hold” characterizes the performance of the B&H strategy; this same line is coincident with the current index evaluation at close price. In this same illustration, the performance of the S&H strategy is exposed by the curve tagged “Sell & Hold”. An example of the trading performance of one of the optimized strategies is also shown in this figure by the line labeled “Trained Chromosome”. On this same

illustration, the X-axis is time, and the Y-axis represents the assets evaluation; the values on this axis are normalized so that they all start at 100.

In order to have better insight into the results, 30 (thirty) experimental runs were performed, the results were collected, and then discrete intervals of 0.1 risk exposure were considered. With this data, charts similar to the one shown in Figure 21 were built. Figure 21 plots an example of the observed distribution of the Annualized ROI as a function of the risk exposure; the example shown is for the case of the DAX index. This illustration shows the First Quartile of data (Q1), the Third Quartile of data (Q3), and the Median, with the whiskers located, respectively, at 10% and 90% of the data in the 30 independent runs. Again, in this figure, the risk-return trade-off is evident, where the Median of the Annualized ROI increases for higher levels of risk exposure.

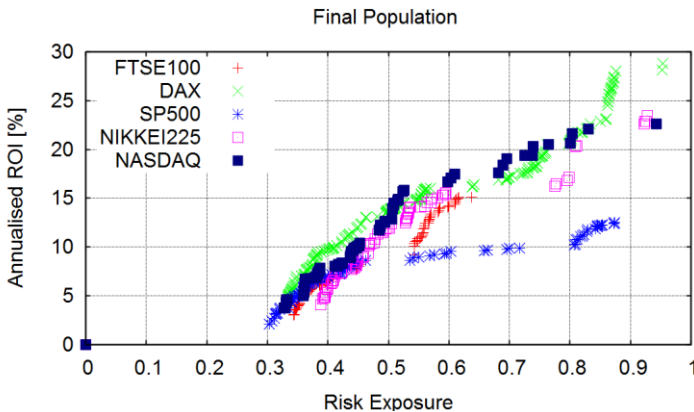


Figure 19. Evolved Pareto Fronts for the 5 Indexes Tested.

The lack of solutions on the risk exposure range of $[0.1, 0.3]$ can be due to the difficulty in optimizing the chosen TI to exploit the price movements in order to create strategies in this region. Similar results were observed for the additional indexes also being tested.

C.4.3. Correlation Analysis of Training and Test Performance

The results presented in the previous subsection showed that it is possible to tune a TS to attain attractive returns at various levels of risk exposure. Despite this, the effectiveness of any approach will depend on being able to extend these interesting returns to unseen data, which is usually recognized as its generalization performance.

Table 3. Annualized ROI for B&H and S&H strategies in the training period

	NIKKEI 225	FTSE 100	S&P500	DAX30	NASDAQ
Index Value at Start	19002.86	6662.90	1455.22	6750.76	4131.15
Index Value at End	15307.78	6456.90	1468.36	8067.32	2652.28
B&H Absolute Return	-3695.08	- 206.00	13.14	1316.56	-1478.87
B&H ROI [%]	-19.44%	- 3.09%	0.90%	19.50%	- 35.80%
B&H Annualized ROI [%]	-2.67%	-0.39%	0.11%	2.25%	-5.39%
S&H Absolute Return	3695.08	206.00	-13.14	-1316.56	1478.87
S&H ROI [%]	19.44%	3.09%	- 0.90%	- 19.50%	35.80%
S&H Annualized ROI [%]	2.25%	0.38%	-0.11%	-2.67%	3.90%

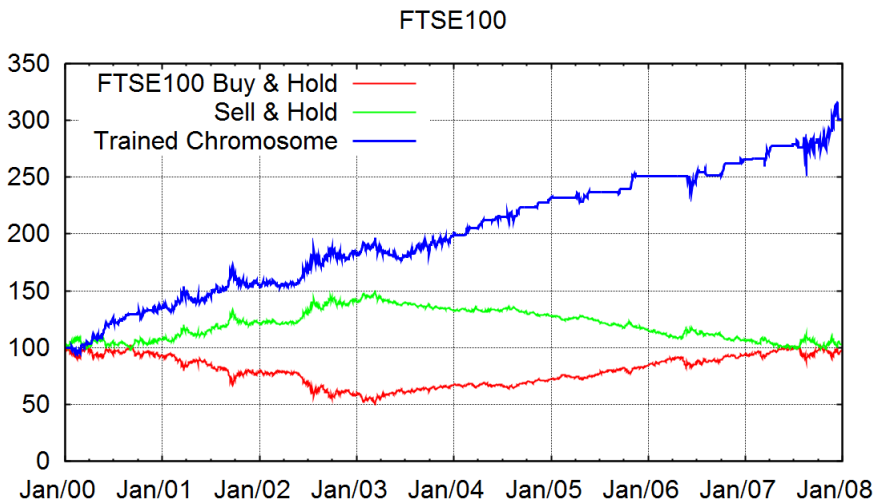


Figure 20. Example of daily closing prices and the performance of one trained TS, for FTSE100 index, in the training period.

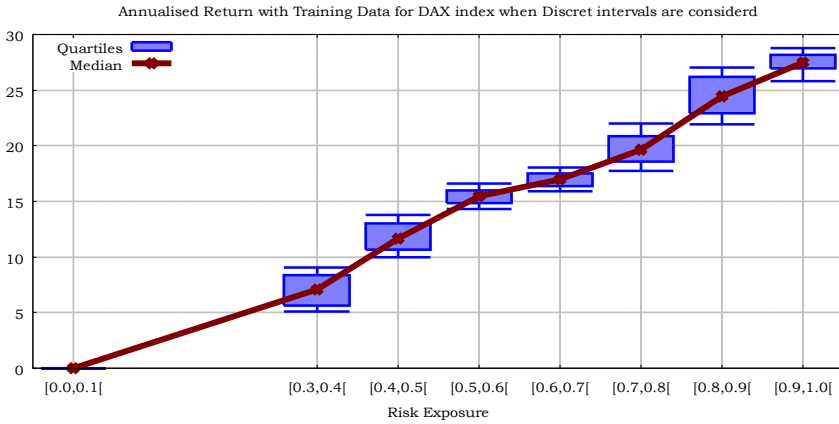


Figure 21. Annualized ROI when discrete intervals of 0.1 Risk Exposure are considered.

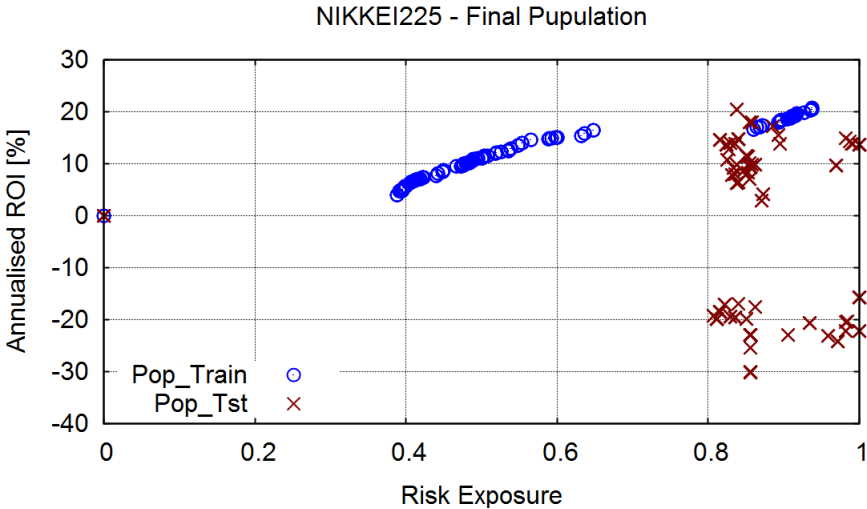


Figure 22. Pareto Fronts for training and test data.

In order to evaluate the engine generalization performance, the available trading data is portioned into two independent sets of data, the training and test data sets, as explained in subsection C.4.1. Training and Testing Data Sets. In the training phase of the evolutionary process, the fitness of the TS will be trained, tuned, and evaluated using only the training data. After having been trained, the developed strategies obtained in the final generation will then be applied to the test data set and its generalization performance will be evaluated. This is an indicator of the framework’s real effectiveness in achieving good results using unseen data.

The plot of the risk-return PFs for the training data acquired in one of the experimental runs is presented in Figure 22. The marks labeled “Pop_Train” represent the performance of the final population evolved after 400 generations, while the points tagged “Pop_Tst” represent the results of this same population when applied to the test data set.

Again, in this plot, the risk-return trade-off is clearly evident with training data. However, this type of correlation disappears when the same strategy is applied to test data. For instance, an annualized ROI of 20% is realizable at a risk level of about 0.7 with the training data, while large losses are suffered at the same level of risk with the test data.

The most evident conclusion is that positive returns with the training data do not necessarily match positive returns with the test data. The example shown in Figure 22 is for the NIKKEI index, but similar plots were observed with the other indexes tested. This low relation between training and testing results was also observed in previous studies [91] and [87].

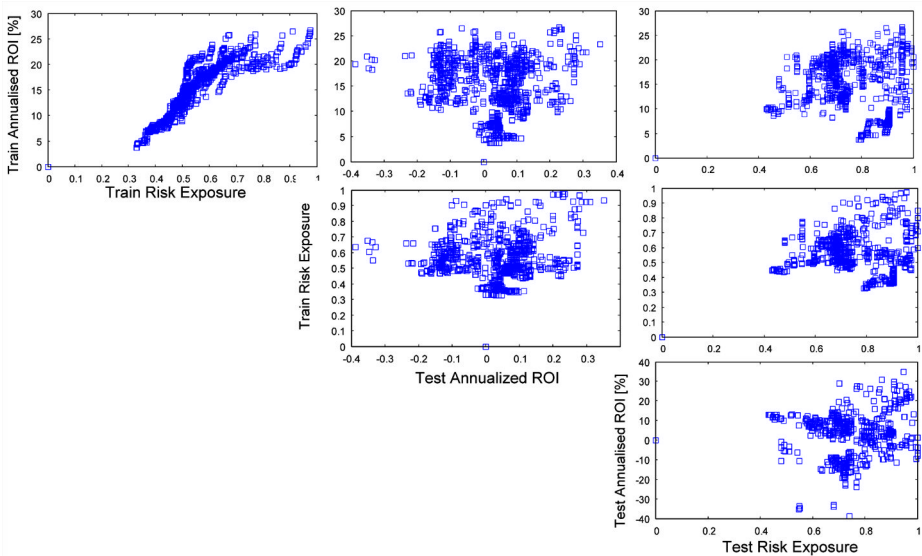


Figure 23. Plots showing the correlation between training returns, training risk, test returns, and test risk.

This strongly suggests the need to better understand how the training and testing data correlate in order to examine the generalization performance of the evolved TS. This suggests the need for a correlation analysis between the four variables involved: training ROI, training risk, test ROI, and test risk.

To better clarify the results, 30 independent experimental runs were performed. With the results observed in these experimental runs, graphs similar to the ones shown in Figure 23 were built. In these graphs, the four involved variables are plotted and any potential correlations can be visually inspected.

Once more, the plot of training ROI and training risk accurately shows the risk-return trade-off. Although an almost random plot is obtained when the test returns against the test risk are plotted; this suggests the existence of a low correlation between training ROI and test ROI.

Contrasting to traditional theories in single-objective approaches, where higher training returns are coupled with higher test returns, this relationship is missing from these plots. Instead, higher training returns correspond to increased volatility in the observed test returns; this is clearly observable in the graphs of Figure 24.

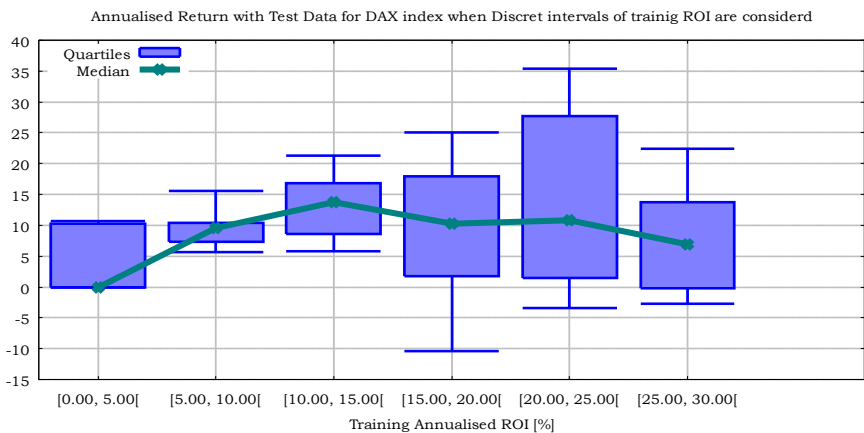


Figure 24. Statistical distribution of test returns at discrete intervals of training returns for DAX index.

In Figure 24 are plotted the quartiles of data (Q1-Q3), the median, and also the whiskers, located at 10% and 90%, respectively, of the observed results, when the training returns are divided into discrete intervals of 5%. In this figure, it is observable that the median of the test returns does not increase when the values of training returns increase. In its place, there is a visible increase in the variance of the results that is denoted by the taller vertical bars (both whiskers and boxes).

In conclusion, the positive correlation that is typically implicit in conventional single-objective approaches, to perform the optimization of TS between training and test returns, is not necessarily true for all cases.

Similar conclusions can be extracted from the plots in Figure 25 and Figure 26, where the Median, Q1, Q3, and whiskers of the test returns are plotted. This time, the results observed in the 30 independent runs are summarized at discrete intervals of 0.1 training risk.

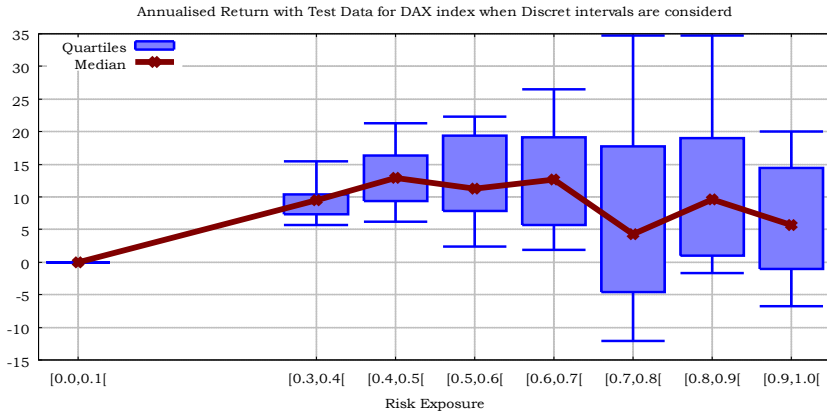


Figure 25. Statistical distribution of test returns at discrete intervals of training risk, for DAX experiments.

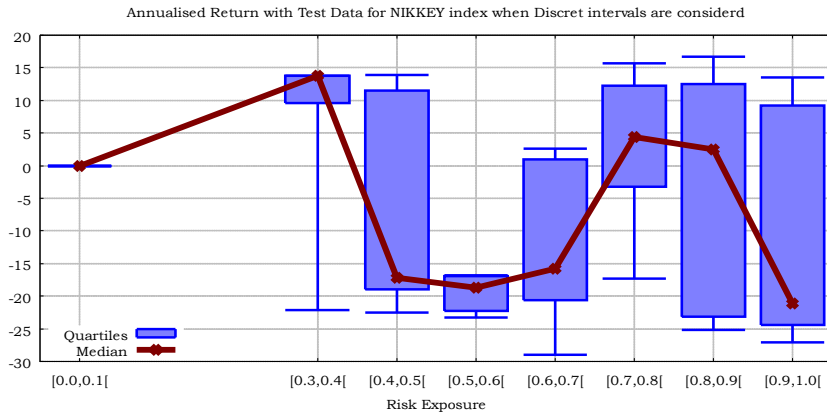


Figure 26. Statistical distribution of test returns at discrete intervals of training risk, for NIKKEY experiments.

Again, the Median of the test returns does not increase when the training risk increases.

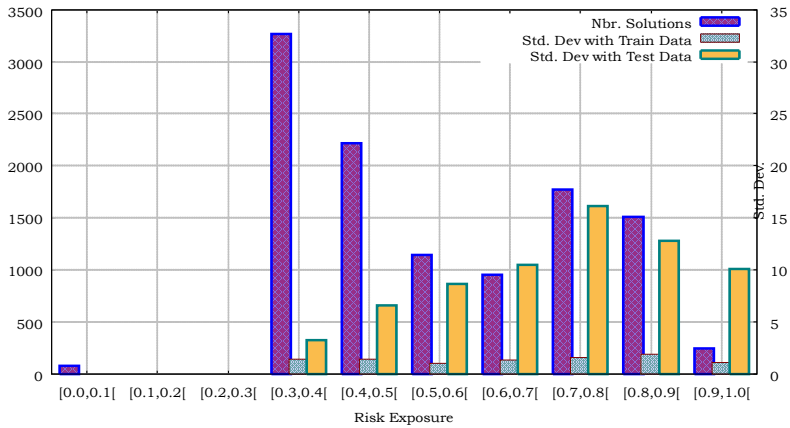


Figure 27. Number of Solutions and Standard Deviation of returns when discrete intervals of 0.1 risk exposure are considered, observed with the DAX Index.

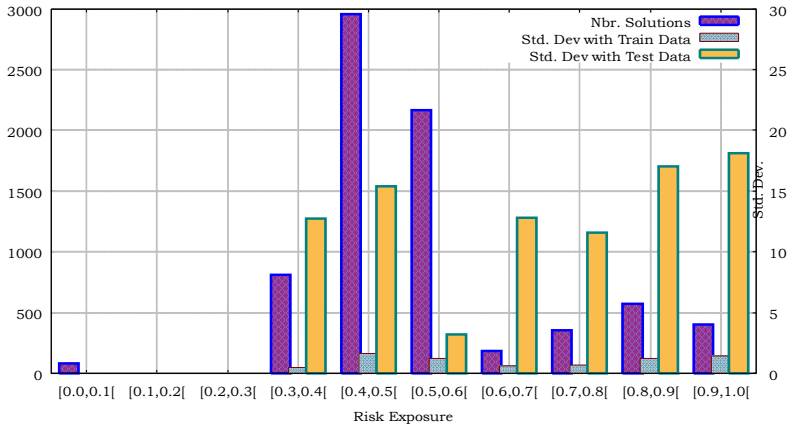


Figure 28. Number of Solutions and Standard Deviation of returns when discrete intervals of 0.1 risk exposure are considered, observed with the NIKKEI index.

Although a steady increase is clearly observable in the variance of the test returns from the plots (Figure 25, Figure 26, Figure 27, and Figure 28), which confirms the claim that higher training returns correspond to increased volatility in the test returns results.

The apparent drop in the results volatility observed in the DAX results, for risk levels above 0.8, is statistically irrelevant, as there are few solutions in this region (Figure 27). The plots presented were built with the DAX and NIKKEI

results, but similar plots were also observed for the remaining indexes also tested in this study.

Conclusion

This chapter began (Parts A and B) with a quick review of the most important existing Computational Problem Solving Techniques. This review was necessary speedy, as it is possible to write (and there are available) complete books explaining in detail each of them. Therefore, in Part A, the various existing techniques in the fields of time series forecast and systems that learn by example were briefly reviewed. Part B, was devoted to Multi-Objective Systems.

This document continued, in Part C, presenting and investigating a multi-objective evolutionary approach to perform the optimization of a set of TS. In this part of this work, fair and established metrics were used to evaluate both returns and the related risk. Both metrics were simultaneously optimized and a popular TI frequently used by real-world professionals were used as the foundational building block of the core strategy. Furthermore, the TS were trained, and afterwards tested, using data coming from five main stock indexes, representative of the world's most developed economies. The PFs obtained by the algorithm using testing data correctly depicted the intrinsic trade-off between risk and return.

Ideally, a multi-objective evolutionary framework should be able to evolve a set of TS with different levels of risk aversion to suit the diverse profiles of investors, from the most risky to the most conservative. However, the low correlation between training returns and test returns suggest a low potential in the framework generalization capability.

Consequently, the experimental results reveal that the positive connection usually assumed between training and testing returns in conventional single-objective approaches of TS optimization does not necessarily hold true for all cases.

However, some interesting conclusions can be extracted, namely the conclusion that higher training returns correspond to increased volatility in the test return results.

The MAs have the disadvantage of being a trend follower indicator, and signals we can get from such indicators always come with some delay. Further tests should be conducted using other TI; those achieved results should be seen as a benchmark to further improvements with the use of other TI, or even the use of multi-TI strategies. Additionally, further experiments should be conducted to

better clarify the reason for the dramatic difference between the PF achieved with training versus testing data.

Additionally, the reader must also be aware that in single-objective approaches, the system automatically picks a solution without showing the user its related risk. This chosen solution can be an intermediate solution (maybe a solution that balances and finds a reasonable compromise between the two conflicting objectives), which, when analyzed in the test period, gives reasonable results. This can be an explanation for both the common belief that a positive correlation between training returns and test returns exists, and for the reasonable results presented in the literature in such a context. Furthermore, in single-objective approaches, the solution picked and presented to the user is rarely evaluated in the test period, according to the two distinct metrics (risk and return); consequently, it is difficult to tell where it would lie if plotted in a risk/return diagram.

References

- [1] “Quadratic Programming” “*Solvers and scripting (programming) languages*”. Available at http://en.wikipedia.org/wiki/Quadratic_programming.
- [2] Robert J. Dunn: 1985, "Expandable Expertise for Everyday Users". *InfoWorld*, Volume 7, Issue39, pp. 29—31, September 30, 1985.
- [3] Harry K. D. H. Bhadeshia: 1999, "Neural Networks in Materials Science". *ISIJ International*, pp. 966—979, (1999).
- [4] Fernando Fernández-Rodríguez, Christian González-Martel and Simón Sosvilla-Rivero: 2000, "On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market". *Economics Letters*, Vol. 69, Issue 1, pp. 89–94, (October 2000).
- [5] Andrew Skabar and Ian Cloete: 2002, "Neural Networks, Financial Trading and the Efficient Markets Hypothesis ". Proceedings of the *twenty-fifth Australasian conference on Computer science (ACSC2002)*, Volume 4, January-February 2002, pp. 241–249, (2002).
- [6] Kyoung-jae Kim, Hyunchul Ahn. 2012. “Simultaneous optimization of artificial neural networks for financial forecasting”. In *Applied Intelligence*, 36, 4, (June 2012), 887- 898.
- [7] William Leigh, Russell Purvisb and James M. Ragusaa: “Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support”. *Decision Support Systems*, vol. 32, pp. 361–377, 2002.

-
- [8] Takashi Kimoto, Kazuo Asakawa, Morio Yoda and Masakazu Takeoka: "Stock Market Prediction System with Modular Neural Networks". *IJCNN International Joint Conference on Neural Networks*, 1990, San Diego, CA , USA, vol. 1, pp. 1–6, Jun 17–21, 1990.
- [9] Hitoshi Iba and Takashi Sasaki: "Using Genetic Programming to Predict Financial Data". Proceedings of the *1999 Congress on Evolutionary Computation (CEC99)*, volume 1, pages 244–251, Washington D.C., USA., Jul. 06–09, 1999.
- [10] Zbigniew Michalewicz: "Genetic Algorithms + Data Structures = Evolution Programs". *Springer-Verlag*, 2nd edition, 1994.
- [11] Adam Marczyk: "*Genetic Algorithms and Evolutionary Computation*, by Adam Marczyk". Available at <http://www.talkorigins.org/faqs/genalg/genalg.html>, 2004.
- [12] Agoston E. Eiben and J. E. Smith: "Introduction to Evolutionary Computing". Springer, *Natural Computing Series*, corrected 2nd printing, 2007.
- [13] Angan Dass and Ranga Vemuri: "GAPSYS: A GA-Based Tool for Automated Passive Analog Circuit Synthesis". Proc. of *IEEE International Symposium on Circuits and Systems (ISCAS 2007)*, New Orleans, Louisiana, USA, May 27–30, 2007.
- [14] John R. Koza: "*Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*". Computer Science Department of Stanford University, June 1990. Available at <ftp://reports.stanford.edu/pub/ctr/reports/cs/tr/90/1314/CS-TR-90-1314.pdf>.
- [15] John H. Holland: "Adaptation in Natural and Artificial Systems". *MIT Press*, Cambridge, MA, USA, 1975.
- [16] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, T Meyarivan: 2000, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II", *KanGAL Report No. 200001*, 2000.
- [17] Kalyanmoy Deb, Amrit Pratap, Samir Agrawal, T. Meyarivan: 2002, "A fast and elitist multi-objective genetic algorithm: NSGA-II". In *IEEE Transactions on Evolutionary Computation*, Computer Society, Washington, DC, USA Vol. 6, Issue 2, pp. 182–197, (Apr. 2002).
- [18] Kalyanmoy Deb: 2001, "Multi-Objective Optimization Using Evolutionary Algorithms". *John Wiley & Sons*, Chichester, UK, (June 2001).
- [19] Jung-Hua Wang, Shiuan-Ming Chen: 1998, "Evolutionary Stock Trading Decision Support System Using Sliding Window". In Proceedings of the *1998 IEEE International Conference on Computational Intelligence*, IEEE

- Computer Society, Washington, DC, USA pp. 253–258, Anchorage, AK, USA., (4–9 May, 1998).
- [20] Fahima A. Badawy, Hazem Y. Abdelazim, Hazem Y. Abdelazim, Mohamed G. Darwish, “Genetic Algorithms for Predicting the Egyptian Stock Market”. *Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on Information and Communications Technology*, pp. 109–122, Dec. 2005.
- [21] Pablo Fernández-Blanco, Diego J. Bodas-Sagi, Francisco J. Soltero, J. Ignacio Hidalgo: 2008, “Technical Market Indicators Optimization using Evolutionary Algorithms”. In Proceedings of the *GECCO conference companion on Genetic and evolutionary computation (GECCO '08)*, Atlanta, Georgia, USA, pp 1851–1857, July 12–16, 2008.
- [22] Diego J. Bodas-Sagi, Pablo Fernández, J. Ignacio Hidalgo, Francisco J. Soltero, José L. Risco-Martín: 2009, “Multiobjective Optimization of Technical Market Indicators”. In Proceedings of the *11th Annual Conference Companion on Genetic and Evolutionary Computation Conference (GECCO '09)*, Montréal, Québec, Canada, pp 1999–2004, July 8–12, 2009.
- [23] António Gorgulho, Rui Neves, Nuno Horta: 2009, “Using GAs to Balance Technical Indicators on Stock Picking for Financial Portfolio Composition”. In Proceedings of the *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference (GECCO '09)*, Montreal, Québec, Canada, pp. 2041–2046, July 8–12, 2009.
- [24] William Leigh, Cheryl J. Frohlich, Steven Hornik, Russell L. Purvis, and Tom L. Roberts: “Trading With a Stock Chart Heuristic”. *IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans*, Volume 38, Issue 1, pp 93–104, Jan. 2008.
- [25] Pablo Moscato: “On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms”. *Caltech Concurrent Computation Program* (report 826), 1989.
- [26] Peter Winker and Dietmar Maringer: “The Hidden Risks of Optimizing Bond Portfolios under VaR”. *The Journal of Risk*, Volume 9/ Number 4, Summer 2007, pp 1–19, 2007.
- [27] Claus de Castro Aranha and Hitoshi Iba: “Using Memetic Algorithms To Improve Portfolio Performance In Static And Dynamic Trading Scenarios”. Proceedings of the *11th Annual conference on Genetic and evolutionary computation (GECCO '09)*, Montréal, Québec, Canada, pp 1427–1434, July 8–12, 2009.

-
- [28] Diego J. Bodas-Sagi, Pablo Fernández, J. Ignacio Hidalgo, Francisco J. Soltero and José L. Risco-Martín: “Multiobjective Optimization of Technical Market Indicators”. Proceedings of the *11th Annual Conference Companion on Genetic and Evolutionary Computation Conference (GECCO '09)*, Montréal, Québec, Canada, pp 1999–2004, July 8–12, 2009.
- [29] Pablo Fernández-Blanco, Diego J. Bodas-Sagi, Francisco J. Soltero and J. Ignacio Hidalgo: “Technical Market Indicators Optimization using Evolutionary Algorithms”. Proceedings of the *2008 GECCO conference companion on Genetic and evolutionary computation (GECCO '08)*, Atlanta, Georgia, USA, pp 1851–1857, July 12–16, 2008.
- [30] Alberto Colomi, Marco Dorigo and Vittorio Maniezzo: “Distributed Optimization by Ant Colonies”. Proceedings of *European Conference On Artificial Life (Ecal91)*, Paris, France, pp 134–142, 1991.
- [31] Gianni di Caro and Marco Dorigo: “AntNet: Distributed Stigmergetic Control for Communications Networks”. *Journal of Artificial Intelligence Research*, Volume 9, December 1998, pp 317–365, 1998.
- [32] Vittorio Maniezzo, Luca Maria Gambardella and Fabio de Luigi: “Ant Colony Optimization”. Chapter book in: “*New Optimization Techniques in Engineering*”, by: Godfrey C. Onwubolu and B. V. Babu, Springer-Verlag, pp. 101–117, 2004, available at: <http://www.idsia.ch/~luca/aco2004.pdf>
- [33] E. Bonabeau, Marco Dorigo and G. Theraulaz: “Inspiration for Optimization from Social Insect Behaviour”. *Nature*, Vol 406, 6 July 2000, pp. 39–42, 2000.
- [34] James Kennedy and Russell Eberhart: “Particle Swarm Optimization”. Proceedings of *IEEE International Conference on Neural Networks*, pp. 1942–1948 vol.4, Nov. 27–Dec 01, 1995.
- [35] James Kennedy, Russell C. Eberhart and Yuhui Shi: “Swarm Intelligence”. *Morgan Kaufmann / Academic Press*, 2001.
- [36] Rania Hassan, Babak Cohanin and Olivier de Weck: “A COPMARISON OF PARTICLE SWARM OPTIMIZATION AND THE GENETIC ALGORITHM”. In *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Austin, Texas, April 2005, pp. 1–13, 2005.
- [37] Margarita Reyes-Sierra and Carlos A. Coello Coello: “Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art”. *International Journal of Computational Intelligence Research*, Vol.2, No.3, pp. 287–308, 2006.
- [38] Matthew Butler and Dimitar Kazakov: 2010, “Particle swarm optimization of Bollinger bands”. In Proceedings of the *7th international conference on*

- Swarm intelligence (ANTS'10)*, Springer-Verlag Berlin, Heidelberg, pp. 504–511, (September 2010).
- [39] Antonio C. Briza and Prospero C. Naval Jr.: “Design of stock trading system for historical market data using multiobjective particle swarm optimization of technical indicators”. *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation (GECCO'08)*, Atlanta, Georgia, USA, July 12–16, 2008.
- [40] Christian Blum and Andrea Roli: “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”. *ACM Computing Surveys (CSUR)*, Volume 35 Issue 3, ACM New York, NY, USA, September 2003.
- [41] Cyril Schoreels and Jonatham M. Garibaldi: “The effect of Varying Parameters on Performance for Adaptive Agents in Technical Equity Market Trading”. *3rd International Conference on Computational Cybernetics, 2005 (ICCC 2005)*, IEEE, pp. 243–248, 13–16 April, 2005.
- [42] Cyril Schoreels, Jonatham M. Garibaldi: 2005, “A Comparison of Adaptive and Static Agents in Equity Market Trading”. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT '05)*. Compiègne, France, (IEEE/WIC/ACM), pp. 393–399, 19–22 Sept., 2005.
- [43] Cyril Schoreels, Brian Logan and Jonathan M. Garibaldi: “Agent based Genetic Algorithm Employing Financial Technical Analysis for Making Trading Decisions Using Historical Equity Market Data”. *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004 (IAT 2004)*, pp. 421–424, 20–24 Sept., 2004.
- [44] Cyril Schoreels, Jonathan M. Garibaldi: 2006, “Genetic algorithm evolved agent-based equity trading using Technical Analysis and the Capital Asset Pricing Model”, In *Proceedings of the 6th International Conference on Recent Advances in Soft Computing (RASC2006)*. Canterbury, UK, pp. 194–199, 2006.
- [45] Harish Subramanian, Subramanian Ramamoorthy, Peter Stone, Benjamin J. Kuipers: “Designing safe, profitable automated stock trading agents using evolutionary algorithms”. *Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO'06)*, Seattle, Washington, USA., pp. 1777–1784, July 8–12, 2006.
- [46] Michael Kearns, Luis Ortiz: “The Penn-Lehman Automated Trading Project”. *IEEE Intelligent Systems*, Volume: 18 Issue: 6, pp. 22–31, Nov.–Dec. 2003.

-
- [47] Michael Kearns, Luis Ortiz: "*THE PENN-LEHMAN AUTOMATED TRADING PROJECT*". Available at: <http://www.cis.upenn.edu/~mkearns/projects/plat.html>.
- [48] Rikiya Fukumoto, Hajime Kita: "Designing Trading Agents for an Artificial Market with a Multi-Objective Genetic Algorithm". *Proceedings of the Fourth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '01)*, IEEE Computer Society, Washington, DC, USA., page 226, 2001.
- [49] Koichi Kurumatani, Yuhsuke Koyama, Takao Terano, Hajime Kita, Akira Na-matame, Hiroshi Deguchi, Yoshinori Shiozawa, Hitoshi Matsubara: "U-Mart: A Virtual StockMarket as a Forum for Market Structure Analysis and Engineering". In *Proc. 5th Joint Conference on Information Science (JCIS'00)*, 1st Int'l Workshop on Computational Intelligence in Economics and Finance, Vol. 2, pp. 957–960, 2000.
- [50] Yoshinori Shiozawa, et al.: "*U-Mart Project*". Available at <http://www.u-mart.org>.
- [51] Corinna Cortes, Vladimir N. Vapnik: "Support-Vector Networks". *Machine Learning*, Volume 20, Number 3, pp. 273–297, 1995.
- [52] V. Vapnik: "The Nature of Statistical Learning Theory". *Springer*, 1999.
- [53] Kuan-Yu Chen, Chia-Hui Ho: "An Improved Support Vector Regression Modeling for Taiwan Stock Exchange Market Weighted Index Forecasting". *The IEEE International Conference on Neural Networks and Brain, 2005 (ICNN&B 2005)*, Beijing, Chine, Volume 3, pp. 1633–1638, (13–15 Oct., 2005).
- [54] Alan Fan, Marimuthu Palaniswami: "Stock Selection using Support Vector Machines". In *Proceedings of the International Joint Conference on Neural Networks, 2001 (IJCNN 2001)*, Volume 3, pp. 1793–1798, (15–19 Jul., 2001).
- [55] Chenn-Jung Huang, Dian-Xiu Yang, Yi-Ta Chuang: "Application of wrapper approach and composite classifier to the stock trend prediction". *Expert Systems with Applications (ESWA)*, Volume 34, Issue 4, (May 2008), pp. 2870–2878, 2007.
- [56] Wei Yan, Martin Sewell, Christopher D. Clack: "Learning to Optimize Profits Beats Predicting Returns — Comparing Techniques for Financial Portfolio Optimisation". *Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO 2008)*, Atlanta, Georgia, USA, pp. 1681–1688, (12–16 July, 2008).

-
- [57] C. J. Huang, D. X. - Yang, Y. T. Chuang, “Application of wrapper approach and composite classifier to the stock trend prediction”, *Expert Systems with Applications*, vol. 34, pp. 2870–2878, (2007).
- [58] Kevin I. Smith, Richard M. Everson, Jonathan E. Fieldsend: “Dominance Measures for Multi-Objective Simulated Annealing”. *Congress on Evolutionary Computation, 2004. (CEC2004)*, Volume 1, pp. 23–30, (19-23 June, 2004).
- [59] Kevin I. Smith, Richard M. Everson, Jonathan E. Fieldsend, Chris Murphy, Rashmi Misra: “Dominance-Based Multiobjective Simulated Annealing”. *IEEE Transactions on Evolutionary Computation*, Volume: 12 Issue:3, pp. 323–342, IEEE Computational Intelligence Society, (June 2008).
- [60] J. David Schaffer: “Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”. *Proceedings of the 1st International Conference on Genetic Algorithms (ICGA'85)*, pp. 93–100, Pittsburgh, PA, USA, (19-23 July, 1985).
- [61] Prasadarnng Skolpadungket, Keshav Dahal, Napat Harnpornchai: “Portfolio Optimization using Multi-Objective Genetic Algorithms”. *IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 516–523, (19-23 June, 2007).
- [62] Carlos M. Fonseca, Peter J. Fleming: “Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization”. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, (Morgan Kaufmann, S. Forrest, ed.) pp. 416-423, (July 1993).
- [63] Eckart Zitzler, Marco Laumanns, Lothar Thiele: “SPEA2: Improving the Strength Pareto Evolutionary Algorithm”. Technical Report 103 (TIK-Report 103) *Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich*, (May 2001).
- [64] K. Metaxiotis, K. Liagkouras: “Multiobjective Evolutionary Algorithms for Portfolio Management: A comprehensive literature review”. *Expert Systems with Applications (ESWA)*, Vol. 39, Issue 14, pp. 11685–11698, (October, 2012).
- [65] Darrell Whitley 1991. “A Genetic Algorithm Tutorial”, *Statistics and Computing*, 4, 65– 85, (1991).
- [66] Eugene F. Fama: 1970, “Efficient capital markets: A review of theory and empirical work”. In *The Journal of Finance*, 25, 2, (May 1970), 383-417.
- [67] Jerzy Korczak, Patrick Roger: 2002, “Stock Timing Using Genetic Algorithms”, in Applied Stochastic models, In *Business and Industry*. 18, (January 15, 2002), 121-134.

-
- [68] S. B. Achelis: 2000, "Technical Analysis from A to Z", 2nd Edition. *McGraw-Hill*, New York. 2nd Edition, New York. (October 2, 2000).
- [69] John J. Murphy: 1999, "Technical Analysis of Financial Markets", *Prentice Hall Press*, New York Institute of Finance, New York. (January 4, 1999).
- [70] Graham, Benjamin, Zweig, Jason, Buffett, Warren E. 2003. The Intelligent Investor, Revised edition. Revised edition. *Collins Business*, (July 8, 2003).
- [71] Allen, Franklin, Karjalainen, Risto. 1995. Using genetic algorithms to find technical trading rules. Working paper, The Wharton School. *The Wharton School, University of Pennsylvania*, (October 9, 1995).
- [72] Allen, Franklin, Karjalainen, Risto. 1999. Using genetic algorithms to find technical trading rules. In *Journal of Financial Economics*, 51, (October 27, 1999), 245-271.
- [73] Wuthrich, B., Cho, V., Leung, S., Permunetilleke, D., Sankaran, K., Zhang, J., Lam, W. 1998. Daily Stock Market Forecast from Textual Web Data. In *1998 IEEE International Conference on Systems, Man, and Cybernetics*, 3, (11-14 Oct), 2720-2725.
- [74] Manuel E. Fernandez Garcia, Enrique A. de la Cal Marin, Raquel Quiroga Garcia. 2010. Improving return using risk-return adjustment and incremental training in technical trading rules with GAPS. In *Applied Intelligence*, 33, 2, (October 2010), 93-106.
- [75] Vincent Cho. 2010. MISMIS - A comprehensive decision support system for stock market investment. In *Knowledge-Based Systems*, 23, 6 (August, 2010), 626-633.
- [76] António Gorgulho, Rui Neves, Nuno Horta: 2011, "Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition". In *Expert Systems with Applications (ESWA)*, 38, 11, (October 2011), 14072-14085.
- [77] Wang, Lei, Wang, Qiang. 2011. Stock Market Prediction Using Artificial Neural Networks Based on HLP. In *Third International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. 1, (26-27 Aug. 2011), 116-119.
- [78] Kyoung-jae Kim, Hyunchul Ahn. 2012. Simultaneous optimization of artificial neural networks for financial forecasting. In *Applied Intelligence*, 36, 4, (June 2012), 887- 898.
- [79] António Canelas, Rui Neves, Nuno Horta: 2013, "A SAX-GA approach to evolve investment strategies on financial markets based on pattern discovery techniques". In *Expert Systems with Applications (ESWA)*, 40, 5, (April 2013), 1579-1590.

-
- [80] Wei Yan, Christopher D. Clack: “Evolving Robust GP Solutions for Hedge Fund Stock Selection in Emerging Markets”. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO)*. London, UK, 2234-2241, 2007.
- [81] Harry M. Markowitz: “Portfolio Selection”. In *The Journal of Finance*, 7, 1, (Mar., 1952), 77-91.
- [82] “VIX White Paper”. Available at: <http://www.cboe.com/micro/vix/vixwhite.pdf>.
- [83] “History of VIX development”. Available at <http://www.stock-options-made-easy.com/volatility-index.html>.
- [84] William Forsyth Sharpe: “Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk”. In *The Journal of Finance*, 19, 3, (Sep., 1964), Blackwell Publishing, American Finance Association, 425-442.
- [85] José Pinto, Rui. Neves, Nuno Horta: 2011, “Fitness function evaluation for MA trading strategies based on genetic algorithms”. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (GECCO '11)*. ACM, New York, NY, USA, 819-820, 2011.
- [86] Ghada Hassan, Christopher D. Clack: 2009, “Robustness of Multiple Objective GP Stock-Picking in Unstable Financial Markets”. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO)*. Montreal, Canada, 1513-1520, 2009
- [87] Swee Chiang Chiam, Kay Chen Tan, Abdullah Al. Mamun: 2009, “Investigating technical trading strategy via an multi-objective evolutionary platform”. In *Expert Systems with Applications*, 36, 7, (Sep., 2009), 10408-10423..
- [88] Graham Kendall, YanSu: 2003, “A multi-agent Based Simulated Stock Market – Testing on Different Types of Stocks”. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'03)*, 4, 2298-2305, 2003.
- [89] Richard L. Weissman: 2005, “*Mechanical Trading Systems: Pairing Trader Psychology with Technical Analysis*”. Wiley Trading. Wiley Trading, 2005.
- [90] Andreas Steiner: 2010, “Ambiguity in Calculating and Interpreting Maximum Drawdown”. *Research Note*, Andreas Steiner Consulting GmbH. Switzerland, (December 15, 2010), 1-7.
- [91] J. J. Korczak, Piotr Lipinski: 2004, “Evolutionary Building of Stock Trading Experts in a Real-Time System”. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*. Portland, USA, (19-23 June, 2004), 940-947.
- [92] J. Neely, Christopher: 2003, “Risk-Adjusted, Ex Ante, Optimal, Technical Trading Rules in Equity Markets”. In *International Review of Economics and Finance*, 12, 1 (Spring 2003), 69-87.

-
- [93] Michael O'Neill, Anthony Brabazon, Conor Ryan: 2002, "Forecasting market indices using evolutionary automatic programming: A case study". In: *Genetic algorithms and genetic programming in computational finance*, Springer US, pp 175-195, 2002.
- [94] "Real Estate Investment Funds CMVM Regulation 8/2002". *Diário da República*, II Série, 2, 8-06-2002, (18-06-2002), available at: http://www.cvm.pt/EN/Legislacao_Regulamentos/Regulamentos%20Da%20Cvm/2002/Pages/reg2002_08cons7-2007.aspx.

Chapter 2

**PROMOTING BETTER GENERALISATION
IN MULTI-LAYER PERCEPTRONS
USING A SIMULATED SYNAPTIC
DOWNSCALING MECHANISM**

A. Brabazon^{1,*}, *A. Agapitos*^{2,†} and *M. O'Neill*^{1,‡}

¹Complex Adaptive Systems Laboratory and School of Business
University College Dublin, Dublin, Ireland,

²Complex Adaptive Systems Laboratory and
School of Computer Science and Informatics
University College Dublin, Dublin, Ireland

Abstract

A key concern when training a multi-layer perceptron (MLP) is that the final network should generalise well out-of-sample. A considerable literature has emerged which examines various aspects of this issue. In this study we draw inspiration from theories of memory consolidation in order to develop a new methodology for training MLPs in order to promote their generalisation capabilities. The *synaptic homeostasis hypothesis* [29, 30] proposes that a key role of sleep is to downscale synaptic

*E-mail address: anthony.brabazon@ucd.ie

†E-mail address: alexandros.agapitos@ucd.ie

‡E-mail address: m.oneill@ucd.ie

strength to a baseline level that is energetically sustainable. As a consequence, the hypothesis suggests that sleep acts not to actively strengthen selected memories but rather to remove irrelevant memories. In turn, this lessens spurious learning, improves the signal to noise ratio in maintained memories, and therefore produces better generalisation capabilities. In this chapter we describe the synaptic homeostasis hypothesis and draw inspiration from it in order to design a ‘wake-sleep’ training approach for MLPs. The approach is tested on a number of datasets.

1. Introduction

A key concern when applying powerful machine learning methods such as MLPs to induce a model from a training dataset, is that the resulting model should generalise well out of sample. There are several issues that will impact on the generalisation capability of a MLP, including the sufficiency of the training dataset (i.e. does it contain sufficient explanatory inputs in order to allow construction of a predictive model for the target output), is the training data sufficiently representative of all out of sample data that could be presented to the model, is the target function smooth (non-smooth functions will be more difficult to model), and what choice of error criterion will promote good generalisation?

Another factor which will impact on how well an MLP will generalise is its internal structure. If too-large a network is employed, it will have many weights and will be prone to over training, thereby learning any ‘noise’ in the data. Increasing the number of weights will also add to the computational complexity of the training process. If too-small a network is used, it will not have sufficient power to adequately represent the structure in the data.

Of course, the importance of generalisation extends far beyond machine learning and statistics, and the ability to generalise from past learning to new situations is a key driver of evolutionary fitness in biological organisms. Hence, processes of learning, memory formation, and the integration of new experiences into existing memories in animals, are likely to be rich sources of inspiration for the design of algorithms with good generalisation capabilities.

It is widely thought that iterated wake-sleep states play an important role in memory formation and maintenance in animals. Despite the rich literature in neural networks concerning generalisation, relatively little attention has been paid to the possibility of drawing inspiration from iterated wake-sleep states in

order to design better training algorithms for neural networks.

1.1. Memory

Broadly speaking, learning can be considered as the process of acquiring new information, with memory referring to the persistence of learning in a state that can be revealed at a later time [27]. The processes of learning and memory formation have been widely studied in the literature of both psychology and neurobiology. In the latter case, the focus of research is on how memories are recorded and maintained in the physical structure of the brain. The basic structural unit of the brain consist of individual neurons. A critical aspect of learning and memory is that the connection structure between these neurons is plastic and is altered via the process of learning. The concept of plasticity was first suggested over a century ago by William James [7], and the *synaptic plasticity hypothesis* lies at the centre of most research on memory storage [20]. This hypothesis proposes that the strength of synaptic connections between neurons, which in turn determine the ease with which an action potential in one cell excites or inhibits its target cell, are not fixed but are modifiable or ‘plastic’.

While there are multiple types of neurons, the canonical model of information flow at a neuron (the ‘neuron doctrine’) is that the cell body of a neuron integrates the electrical signals which enter the cell through nerve fibres called dendrites. If the total input signal into a neuron in a time period exceeds a threshold level, the neuron ‘fires’ and sends an output electrical signal along its axon. In turn, the axon of a neuron is connected to the dendrites of other neurons. Consequently, the firing of an individual neuron can produce a cascade effect in other neurons.

A neuron typically has a dense web of input dendrites and these connect, via a synapse, to axon terminals of other neurons at small structures known as dendritic spines. These spines can grow or shrink and are constantly extending out of and retracting back into the dendrite. Hence, the precise network of connections between neurons in a brain is not fixed, but dynamically alters over time. Indeed, two individual neurons may have multiple and not just a single connection. As learning takes place, the network of connections adapts and changes take place at synaptic junctions which can enhance or reduce the ease with which electrical signals can cross the synaptic gap. Memory is stored in a network of linked neurons.

1.1.1. Memory Consolidation

The *memory consolidation hypothesis* was first proposed over a century ago by Müller and Pilzecker [15] and posits that new memories are initially fragile and are only gradually consolidated into long term memory. As noted by [14], while storage of new events in memory can occur very quickly (within seconds), slow consolidation of memories into long term storage (a process which can take days, weeks, or even longer) may be adaptive as it allows for a dynamic interplay between current experience and pre-existing memories.

The term memory consolidation is itself variously defined as, ‘a time-dependent, off-line process that stabilizes memories against interference and decay, allowing them to persist over time’ [14], a ‘process that transforms new and initially labile memories encoded in the awake state into more stable representations that become integrated into the network of pre-existing long-term memories’ [3], or as ‘the processes that stabilise the learning-induced changes in synaptic morphology that represent the biological substrate of memory’ [5].

In discussing memory consolidation, a distinction is drawn between:

1. cellular consolidation, and
2. systems consolidation.

Cellular consolidation arises from a series of biochemical events which take place in individual synapses, typically within a short time frame (minutes to hours) after the initial experience. System consolidation refers to events which take place over a longer time frame and which are thought to maintain the memory in long term memory storage.

Rudy (2014) [20] provides an excellent review of the current state of understanding of how memories are created and maintained. While there is still considerable debate concerning several aspects of this process, the most widely accepted view is that memory develops over a number of stages namely, generation, stabilisation, consolidation and maintenance.

Initially, there are changes in the synaptic strength of the effected neurons, resulting from a reorganisation of existing proteins in the relevant dendritic spine and axon terminal. For example, within minutes, the number of glutamate receptors in the spine is increased thereby facilitating the enhanced transmission of sodium ions (electrical signal) between the axon terminal and the spine. To consolidate the synaptic change further, in following hours transcription and translation processes are activated creating new proteins. These have several

effects including the enhancement of the degree of bonding between the spine and axon, and an alteration of the physical geometry of the spine. This further promotes the transmission of ions between the spine and axon. Typically, this process lasts for up to 24 hours and helps ensure that the physical changes in the synapses endure for several days.

While the above explains how synaptic changes initially occur and are subsequently stabilised, it does not explain how strengthened synapses that support memory outlive the molecules from which they are made. This is known as the ‘molecular turnover problem’ and is a active area of research inquiry. In order to maintain a memory, a variety of proteins need to be continually manufactured at the synapse, even in the absence of the original stimulus. Recent work by [12, 34] suggests that self-sustaining (self-copying) populations of proteins may be the key to maintaining the long-term synaptic changes that underlie memory.

Obviously, there is little reason to maintain a memory of most of the routine events which occur during a day, and indeed experience suggests that we will forget much of this detail within several days. It is speculated that memories are most likely to be maintained for the long term when either the behavioural experience is considered significant, is repeated, or when the memory is recalled [5]. As will be discussed later, it is thought that sleep plays an important role in long term memory consolidation.

1.1.2. Memory Systems

When discussing memory, it is important to note that the brain has multiple memory systems, depending on the nature of what is being learnt. Perhaps the best known system is that for declarative memory which includes both episodic memory (memory for facts and events) and semantic memory (supports memory for facts and provides an ability to generalise from multiple experiences). This system relies on an interplay between the neocortex, the hippocampus and its related cortical structures. Sensory information passes into the neocortex and in turn is processed and passed via a number of intermediate structures into the hippocampus. By the time the information passes into the hippocampus it is already highly processed and amodal (hippocampus neurons do not know whether they are receiving auditory, visual or other sensory inputs) [20].

Although it is known that the hippocampus plays a vital role in episodic memory, there is debate as to how exactly it does this. One theory is the ‘in-

dexing theory of episodic memory' [32]. According to this theory, the content of episodic memories are stored in the neocortex and the hippocampus creates indices to these memories by binding the inputs it receives from the different regions of the neocortex into a neural ensemble that represents the conjunction of their co-occurrence [20]. The hippocampus projects back to the neocortex when the index is activated.

In essence, the theory assumes that events create a memory trace by activating patterns of neocortical activity, which then project to the hippocampus, with the relevant synapses in the hippocampus responding to the neocortical inputs being strengthened via long term potentiation (LTP). Therefore, the hippocampus acts as an index to a 'memory' filing cabinet which enables the recall of memories, even when only a subset of the original neocortical pattern is received by the hippocampus. Although this may appear to be an unnecessarily complex process, it is posited that it may have arisen due to structural limitations of the neocortex as potential associative connectivity across neocortical regions is low [20]. It is also speculated that memories in the neocortex may potentially have more than one index associated with them, if the event is repeated or if the memory is reactivated (recalled). Hence, the more often an item is experienced or recalled, the more 'paths' to it may be generated in the hippocampus. This is known as the *multiple trace theory* [16].

1.2. Sleep and Memory Consolidation

At first glance being asleep would appear to be a potentially dangerous and costly activity as sleeping animals cannot forage for resources, take care of young, procreate, and are exposed to predation risk [4, 11]. Despite these drawbacks, sleep behaviours are widespread in the animal kingdom and it is evident that many animals spend a significant portion of their day in sleep or in sleep-like states. Evolution has even devised some extraordinary adaptations to accommodate sleep [31]. Perhaps the most unusual of these adaptations is exhibited by cetaceans (including whales, dolphins and porpoises) who can engage in unihemispherical (or 'half-brain') sleep, wherein one eye is kept open during sleep, with the contralateral side of the brain also remaining awake [18]. Other examples of unihemispheric sleep include some species of birds [19] which can keep one eye open during sleep, particularly if the predation risk is high.

Given the widespread nature of sleep behaviour, and the lengths to which evolution has gone in order to conserve sleep in some animals, one could well

ask what benefit does sleep provide that makes it crucial to living creatures?

Amongst the multiple potential functions of sleep, one of the most heavily researched is whether sleep plays a role memory formation and maintenance. In many species, the same regions of the brain that process sensory information are also important for memory formation. This poses a dilemma, as if these regions are busy processing sensory information during waking, then it is likely to be more difficult for processes such as memory consolidation to take place simultaneously, in turn leading to a suggestion that sleep may allow these conflicting activities to co-exist, leading to a claim that memory consolidation occurs predominately during sleep [1].

In this study we draw inspiration from the synaptic homeostasis hypothesis which is drawn from the literature on memory consolidation in order to design a training approach for a MLP which is capable of generalising from noisy data. Therefore, we simulate a wake-sleep cycle during which the MLP is presented with new sensory inputs (data) during the wake phase, leading to synaptic potentiation, with synaptic downscaling taking place during a simulated ‘sleep’ phase. Critically and in contrast to prior literature on weight-decay processes for training of MLPs, the training process takes place over a sequence of simulated wake-sleep phases.

1.3. Structure of Chapter

The remainder of this chapter is organised as follows. Section 2 provides some background on two theories of memory consolidation during sleep. Section 3 describes the model developed in this study and outlines the experiments undertaken. The results of these are presented and analysed in section 4, with conclusions and suggestions for future work being presented in section 5.

2. Background

In this section we provide some background on memory consolidation during sleep, and in particular, we describe the synaptic homeostasis hypothesis. We also overview some previous literature which has applied ideas from the process of memory consolidation for neural network training.

2.1. Sleep States

A common way to characterise sleep state is to examine the electrical activity of the brain recorded using an electroencephalogram (EEG). In mammals and birds sleep can be divided into two main phases namely, REM (rapid eye movement) and NREM (non rapid eye movement) sleep. REM sleep is characterised by high frequency, low amplitude, electrical activity in the brain, and this bears some similarity to the electrical activity of the brain during wakefulness. In contrast, NREM sleep is characterised by the propagation of low frequency (slow), high amplitude, electrical waves in the brain.

In humans, NREM sleep is divided into three successive stages [21], and the sleep cycle follows a typical ordering of stage 1 NREM, stage 2 NREM, stage 3 NREM, and finally REM sleep. The entire cycle lasts some 90-100 minutes and repeats itself several times during the night. As the sleep cycles progress, the portion of time spent in NREM sleep reduces and the portion of time in each cycle spent in REM sleep increases. Sleep during stage 3 of NREM sleep is termed slow wave sleep (SWS), and is characterised by delta wave activity brain activity, which produces the lowest frequency and highest amplitude patterns of electrical activity.

2.2. Active System Consolidation Hypothesis

There are currently two hypotheses concerning the mechanisms underlying the consolidation of memory during sleep. The active system consolidation hypothesis (ASCH) proposes that an active consolidation process results from the re-activation of selected memories during sleep [3], and the synaptic homeostasis hypothesis (SHH) assumes that consolidation may also occur during waking and that the role of sleep is to restore the encoding capabilities of synaptic connections (global synaptic downscaling) [1].

The ASCH arose from the standard model of systems consolidation for declarative memory [13]. Different regions of brain are responsible for different memories, with *declarative memory* (these memories are accessible to conscious recollection and include memories for facts and events) relying on the hippocampus and neocortical regions of the brain, and *procedural memory* (memories for skills that result from repeated practice e.g. riding a bike or playing a piano) relying on the striatum and cerebellum [3]. The standard two-stage theory for declarative memory consolidation proposes that there are two separate memory stores. One allows learning at a fast rate and serves as

an intermediate buffer to hold information temporarily. The other store learns at a slower rate and serves as long-term memory. For declarative memory, sensory information in the waking brain flows into the cortex and it is proposed that events are initially encoded in parallel in neocortical networks and also in transient neuronal assemblies in the hippocampus.

Although the theory did not initially outline a role for explicit recall in the consolidation of the long term memory, it has been suggested that during sleep, a two-way dialogue between the hippocampus and neocortex takes place in order to effect memory consolidation [3]. The hippocampus can be considered as a rapidly-encoded, sparse, memory system which allows for the formation of event memories, whereas the neocortex is a slowly-consolidating, dense, memory storage system. During NREM sleep, slow (electrical wave) oscillations, spindles, and ripples coordinate the reactivation and redistribution of hippocampus-dependent memories to neocortical sites. The newly-acquired memory traces are reactivated and it is claimed that information flows from the hippocampus to the cortex, such that connections in the neocortex are strengthened, forming more persistent memory representations. In REM sleep, it is proposed that the information flow reverses (from the neocortex back to the hippocampus). This two-way process iterates during the period of sleep [28], thereby modifying the representations in both stores, and integrating the new memory into pre-existing memories. This enables the extraction of invariant features, including the forming of new associations, and eventually insights into hidden rules and patterns [3]. Hence, through the repeated re-activation of the new memories during sleep, the fast learning store acts as an internal trainer of the slow learning store to gradually adapt the new memories to the pre-existing network of long term memories [3].

There is some evidence to support the ASCH, as we know from brain imaging studies that the spatio-temporal patterns of neuronal firing that occur in the hippocampus, during the exploration of a novel environment or during simple spatial tasks, are reactivated in the same order during subsequent sleep. However, we do not have a detailed understanding as to how these reactivations could stimulate the strengthening of links between neocortical storage sites, and specifically, how enduring synaptic changes could result in the neocortex [3]. In the standard two-stage theory, the consolidation process that takes place off-line relies on the re-activation of the neuronal circuits that were implicated in the initial encoding of the memory, and therefore consolidation involves the *reinforcement* of memory representations at the synaptic level. Long-term po-

tentiation (LTP) (Hebbian learning - the assumption that information is stored in the brain as changes in synaptic efficiency which occur when neurons fire synchronously together) is considered a key mechanism of synaptic consolidation. It is not certain whether memory re-activation during sleep promotes the redistribution of memories by inducing new LTP (at long-term storage sites) or whether re-activation merely enhances the maintenance of LTP that was induced during encoding. An assumption of the traditional two stage model is that LTP takes place in the long term memory store as a result of selective reactivation of memories during system consolidation.

Although we await further investigation of sleep dependent learning, recent work by [35] has indicated that sleep (specifically, NREM sleep) by mice after a motor learning task promoted new spine formation in the motor cortex of those mice.

It has been speculated that spindle oscillations which are concentrated in stage 2 NREM, open molecular gates to plasticity by evoking calcium entry in neocortical pyramidal neurons, priming the neurons for biochemical events that could lead to permanent changes in the network. Consolidation could then proceed by iteratively recalling and storing information in primed neural assemblies [22]. One interesting feature of reactivations during SWS is that they appear to be noisier, less accurate, and often happen at a faster firing rate than the related activity during the initial encoding phases. Plausibly this 'noisy' teaching could result in more robust memory in an analogue to using 'jitter' in training MLPs.

2.3. Synaptic Homeostasis Hypothesis

An alternative perspective which has gained a significant following in recent years is the *synaptic homeostasis hypothesis* (SHH) [29, 30, 31]. This hypothesis suggests that the primary memory function of sleep is to produce a global synaptic downscaling, and that memory consolidation is continuous (i.e. can occur during waking) and not limited to sleeping states.

The proponents of the SHH do not disagree that memories form as neurons that get activated together strengthen their links through synaptic potentiation, nor that brains replay newly-learned material at night, or that patterns of neural activity during sleep sometimes resemble those recorded while a subject is awake. However they question conventional wisdom that brain activity during sleep reinforces the synapses involved in storing newly-formed memories, noting that there is no strong evidence that synapses in replayed circuits get

strengthened during sleep [31]. Instead they claim that a critical driver of sleep is a need to restore the brain to a baseline state, by *weakening* the links between neurons during sleep, in order to preserve the brain's ability to learn and form new memories while it is awake. The weakening process is termed *synaptic downscaling*.

Brain tissue is metabolically expensive. In humans, the brain while accounting for only about 2% of total body mass, consumes some 20% of energy requirements during quiet waking [24]. Approximately 2/3 of this energy consumption goes to supporting and maintaining synaptic activity. Strong synapses consume more energy than weak ones and the energy budget available to brain tissue is not unlimited. During the day, the potentiation of synaptic circuits from sensory inputs results in an increase in the number and size of synapses, leading to a higher level of energy requirement [31]. Advocates of the SHH claim that a generalised depression of synapses during sleep would benefit the brain as it would decrease the energy cost of synaptic activity, eliminate weak and ineffective synapses, and reduce cellular stress [2].

An important part of effective learning is a corresponding 'forgetting' of irrelevant memories. Under the SHH, synaptic potentiation stemming from daytime learning is down regulated brain-wide during slow wave sleep. Crucially, it is assumed that this rescaling process preserves relative synaptic weight differences, and therefore may lead to forgetting because the downscaling may effectively silence, or even remove, spines with synapses that are only weakly potentiated. Down selection under the hypothesis promotes survival of only the fittest neural circuits, either because they were activated strongly and consistently during wakefulness, or because they were better integrated with pre-existing memories (for example, a new word in a known language). Synapses that were only mildly enhanced during wakefulness, or which fit less well with existing memories would be depressed, and leave no lasting trace in our neural circuitry.

While there is experimental evidence for several aspects of synaptic downscaling [19], including evidence from animal studies that the number and size of spines and related synapses reduces during sleep [31], there is as yet no direct evidence for a specific mechanism which selectively weakens activated synapses during sleep [31]. It is speculated that the slow waves of mammalian NREM sleep play a role. We know that at sleep onset, levels of SWA are elevated as a result of synaptic strength accrued during learning while awake. This increase in effective connectivity causes the slow-oscillations of neurons to be

more synchronous, and thereby levels of SWA to be high [19]. The large-scale slow oscillations of neuronal networks may produce synaptic downscaling, a global decrease in synaptic strength, and an increase the signal to noise ratios for important memories by eliminating synapses below a certain threshold. This may explain why performance on certain cognitive tasks increases following sleep [19]. Interestingly, synaptic downscaling is a self-limiting process because as synapses weaken, neurons oscillate less synchronously and consequently induce less downscaling [19] (p.265). It is also known that the chemistry of the brain changes during sleep and Tononi and Cirelli [31] have speculated that this could bias neural circuitry so that synapses becomes weakened rather than strengthened when signals flow across them.

The SHH, with its emphasis on an ‘active decay’ (forgetting) of irrelevant memories during sleep, provides an interesting alternative to the traditional idea of sleep-mediated synaptic strengthening of important memories. Most memories formed during the day are irrelevant and a decay process which ensured that unwanted and unneeded memories are removed could result in a lessening of spurious learning and better generalisation capabilities [5].

In this study, we do not claim that the SHH provides a more correct description of memory consolidation during sleep than the ASCH as current empirical evidence does not conclusively support the SHH. Indeed, it has been noted by Axmacher et al. [1] and by Diekelmann and Born (2010) [3] that the ASCH and the SHH are not necessarily mutually exclusive, as a sequential process could exist with active system consolidation integrating newly encoded memories with pre-existing long term memories thereby inducing conformational changes in the neocortex followed by global synaptic downscaling in order to avoid the saturation of synaptic networks. Rather, we draw inspiration from the SHH in order to design a training process for MLPs.

2.4. Synaptic Downscaling and Regularisation

The synaptic downscaling concept bears interesting comparison with some classical approaches to regularisation in the neural network literature. Broadly speaking, regularisation is any modification to a learning algorithm which aims to reduce the chance of overfit. Typically, the object is to smooth the response of the final model. Common methods for regularisation include early stopping, wherein training is stopped when the error measure on a hold-out validation sample begins to increase, or the inclusion of a penalty term in the error func-

tion for model complexity. In applications of the latter in MLPs, the error metric is usually defined as MSE plus an additional (weighted) term which consists of the sum of the squares of the weights. This alteration to the error function will tend to reduce weight sizes in the final network and therefore make the network's response smoother. In turn this will tend to reduce overfit as over-fitted mappings require high curvature and hence large weights. The general form of the regularised cost function in this case is given by:

$$E_{reg} = E_{mse} + \alpha\omega \quad (1)$$

where α is the regularisation parameter which controls the trade-off between reducing the error and increasing the smoothing. The term ω is a penalty function which captures the complexity of the underlying network. If the penalty is defined as the sum of the squares of the weights in the MLP, the approach bears similarity to ridge regression in linear models, and it effectively implements a form of 'weight decay' as in each epoch individual weights decay in proportion to their previous size, i.e. exponentially, unless the weight is changed in the learning process [17]. A wide number of variants on this basic approach have been examined including, 'weight elimination' [33], where the decay process is tuned in order to shrink small weight coefficients more heavily.

Although even basic weight decay approaches can notably improve generalisation capabilities [9], we cannot assume that it is optimal to apply the same decay constant to all weights in the network, and in particular, we could suppose that different decay constants should be applied to connections between input and hidden, hidden to hidden, and hidden to output nodes. Nor can we assume that it is optimal to apply the same decay constant(s) for the entire training process, and [33] illustrates an approach where the decay constant is iteratively updated during training.

Apart from reducing the values for weight parameters in a network, another way to attempt to improve generalisation is to directly restrict or seek to reduce the structural complexity of the network. This can be done by restricting the number of hidden layer nodes, or by 'pruning' individual node connections in a network. One approach is to set connections with small weights to zero, thereby 'tuning off' or 'pruning' that connection. After the relevant weights are deleted, the (reduced) network is retrained. A significant number of studies applying network pruning have resulted over the past 25 years following early work by [25, 26].

As noted by [10], there is a close link between weight decay and pruning, as an iterated pruning process effectively reduces to continuous weight-decay during training. A downside of these approaches is that the learning process can be slow due to the need for repeated re-training and there is an implicit assumption that deletion of connections with small weight values will not have much effect on model fit. A better, if often computationally prohibitive, approach would be to delete weights, whose deletion will have least effect on training error (or to train the network using all possible subsets of weights [8]). Of course, to determine which weight to delete, the MLP would need to be iteratively retrained with each weight being removed in turn.

A more computationally feasible approach to pruning was developed by Lecun et al. (1989) [10], namely the optimal brain damage (OBD) approach. In OBD the second derivatives of each weight parameter with respect to the error function are used in order to determine which weights to remove. As for other pruning methods, OBD proceeds in an iterative manner. Initially the full network is trained on the data, a pruning process is then applied, and the new network is then retrained.

From the above discussion, we can see that weight decay and pruning, both features of the SHH, are well-developed techniques in the neural network literature. It is interesting to note that the development of these techniques stemmed from a statistical rather than a biological perspective. An important aspect of the memory consolidation process that has not yet been embedded in the regularisation literature is the iterative nature of memory consolidation, with new memories only being slowly integrated into existing knowledge, with both memories being altered in this process.

2.5. Neural Network Derived from a Sleep Metaphor

As noted in the introduction to this chapter, relatively little attention has been paid to the use of 'sleep' metaphors for design of neural network algorithms. Perhaps the best known of these algorithms is the 'wake-sleep' algorithm of Hinton et al. [6] for unsupervised learning which draws on the standard model of systems consolidation for declarative memory. In Hinton's study, a multilayer network of simulated stochastic neurons is described, with bottom up recognition connections during the wake phase being used to produce a representation of inputs in one or more hidden layers. In the 'wake' phase, neurons are driven by recognition connections, and generative connections are adapted to increase

the probability that they would reconstruct the correct activity vector in the layer below. In the ‘sleep’ phase, neurons are driven by generative connections, and recognition connections are adapted to increase the probability that they would produce the correct activity vector in the layer above. By alternating activity in two directions, the hidden layer representations are modified until they produce an optimal representation of the original signal.

3. Model and Experiments

The general model we adopt for our experiments is a feed forward multi-layer perceptron (MLP). We create training data from four test functions, and for each input vector in the training set, we inject differing amounts of noise into the associated function output, thereby producing ‘learning’ problems of varying difficulty.

The MLP is exposed to a succession of non-overlapping ‘windows’ of training data during its wake cycles. During exposure to each training vector, a learning process takes place in which synaptic potentiation via the back propagation training algorithm is simulated. At the end of each data window, a sleep cycle is simulated during which synaptic downscaling takes place, and this in turn is followed by another wake cycle in which a new window of training data is presented to the MLP network. During downscaling each weight is decreased by a certain percentage.

Once the MLP has been trained, its out of sample performance on clean test data, generated using the relevant function, is assessed. This allows us to determine how well the MLP has performed in uncovering the correct underlying function, in spite of being presented with noisy data during training.

The results from the MLP developed using a simulated synaptic downscaling process are benchmarked against those produced by a feed forward MLP which has been trained in one pass over the training data.

3.1. Datasets

We selected a suite of four synthetic regression problems so that we can reliably generate data with specific amounts of noise. Figures 6, 7, 8 are graphical representations of the bivariate problems F_1 to F_3 , and F_4 respectively. In every synthetic dataset, we randomly sample 100 training examples of the form (x, y) , where the input vector $x \in R^d$, and the response variable $y \in R$. The goal is

to learn a target function f that maps x to y . The response variable of each example is corrupted by random noise drawn according to a Gaussian probability distribution with certain μ and σ . Thus each training set of examples takes the form $\{(x_i, z_i)\}_1^{100}$, where $z_i = f(x_i) + e_i$. $f(x_i)$ is the noise-free value of the target function and e_i is a random variable representing the noise. We experiment with a set of σ values defined as $\{0.01, 0.1, 1.0, 10.0, 30.0, 50.0\}$, and μ set to 0.0. The details of the sampling procedure used for generation of training and test data for different problems are given in Table 1. Note that noise is only added to the training data, whereas the data used to assess model generalisation is not contaminated.

Furthermore, the response value in each input-output pair is normalised within the $[0.0, 1.0]$ interval prior to training. Normalisation of a noise-corrupted value α is performed using $(\alpha - min)/(max - min)$, where min and max are the minimum and maximum values out of 100 training response values respectively. Figure 5(a) shows the histograms of the normalised response values for different regression problems. The same normalisation applies to testing data, however this time each response value is noise-free. Figure 5(b) shows the histogram of the normalised response values for different problems.

3.2. MLP Design

The regression problems F_1, F_2, F_3, F_4 are of two, five, two and two input variables respectively. The architecture of a MLP consists of an input layer with the same number of input nodes as the dimensionality of the input of a problem, a hidden layer of 10 nodes with \tanh activation functions, and an output layer of a single node with a \tanh activation function. Training is performed using standard back-propagation with a learning rate set to 0.005, iterated for 2,000 epochs. We are experimenting with the effect of the number of wake-sleep cycles during training, and tried the proposed method with 5, 10, 15, and 20 cycles. This effectively means that each set of training examples is divided into the respective number of non-overlapping subsets.

Table 1. Regression problems with the respective data sampling ranges for training and test datasets. Notation $x=\text{rand}(a,b)$ means that the x variable is sampled uniform randomly from the interval $[a, b]$.

	Problem	Training data	Test data
F_1	$f(x_1, x_2) = \frac{e^{-(x_1-1)^2}}{1.2+(x_2-2.5)^2}$	100 points $x_1, x_2=\text{rand}(-3.0, 3.0)$	10,000 points $x_1, x_2=\text{rand}(-3.0, 3.0)$
F_2	$f(x_1, x_2, x_3, x_4, x_5) = \frac{10}{5+\sum_{i=1}^5 (x_i-3)^2}$	100 points x_1, x_2, x_3, x_4, x_5 $=\text{rand}(-3.0, 3.0)$	10,000 points x_1, x_2, x_3, x_4, x_5 $=\text{rand}(-3.0, 3.0)$
F_3	$f(x_1, x_2) = x_1 * x_2 + \sin((x_1 - 1) * (x_2 - 1))$	100 points $x_1, x_2=\text{rand}(-3.0, 3.0)$	10,000 points $x_1, x_2=\text{rand}(-3.0, 3.0)$
F_4	$f(x_1, x_2) = \frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$	100 points $x_1, x_2=\text{rand}(-3.0, 3.0)$	10,000 points $x_1, x_2=\text{rand}(-3.0, 3.0)$

4. Results and Analysis

In the figures discussed in this section, we plot the average Mean Squared Error (MSE) that accrues from 50 runs of an MLP using different random weight initialisations as a function of the weight downscaling percentage that takes place during a sleep phase. For comparison purposes we also plot the average MSE that is obtained from the baseline MLP algorithm that uses no weight downscaling. Depending on the level of noise that is injected into each response variable, we categorise the learning problems into easy (Gaussian noise σ of 0.01 or 0.1), moderate (σ of 1.0 or 10.0), and hard (σ of 30.0 or 50.0). In addition, Tables 2, 3, 4, 5 present the standard errors for the out-of-sample MSE estimates.

Figure 1 presents the results for problem F_1 . An observation that is consistent across all different setups for the number of wake/sleep cycles is that for the easy and moderate problem formulations the proposed method outperformed standard MLP. In addition, results suggest no clear trend in the evolution of the MSE curve as a function of the downscaling percentage, however for the smaller levels of noise (i.e. 0.01, 0.1) increasing the percentage of downscaling seems to worsen the generalisation performance.

The results for problem F_2 are presented in Figure 2. Here the number of wake/sleep cycles exert an effect in the out-of-sample performance with their number set to 15 attaining the best generalisation improvement over standard MLP for all problem formulations but the one where noise σ is set to 1.0. Results also suggest that in the easiest case (i.e. noise σ of 0.01), the method of downscaling is difficult to improve performance over standard MLP and in most cases leads to performance deterioration.

Figure 3 presents the results for problem F_3 . In this case, contrary to the results observed in other problems, weight downscaling improves performance over standard MLP in the least noisy problems, whereas the performance deteriorates over that of standard MLP for the noisiest problem formulation. This increase in performance in the case of noise levels of 0.01 and 0.1 can be attributed to the discrepancy between the distributions of the response values between training and testing as can be seen in Figure 5(a) for Function 3 under noise level 0.01 and Figure 5(b) for the same function. This was due to random sampling for the values x_1 and x_2 that created relatively disjoint sets of examples to train and test a model. The median of the response values is approx. 0.03 for training, and 0.57 for testing. Out-of-sample performance is therefore im-

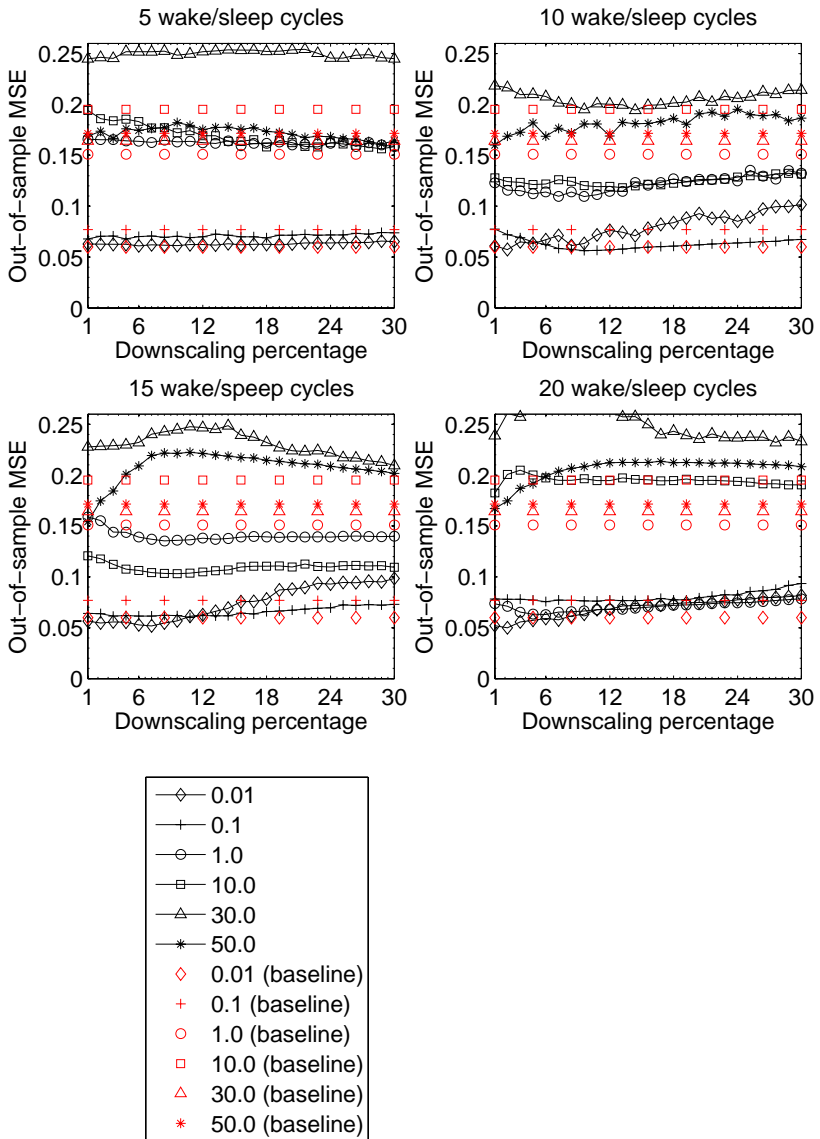


Figure 1. Out-of-sample results for problem F_1 with six different noise levels.

proved by relaxing the fit to the training examples. This incidental result should be regarded as a valid scenario of training and testing data distribution mismatch that can occur when dealing with real-world data. It reinforces the view that in case of overfit models, weight downscaling can improve out-of-sample performance.

Finally, Figure 4 presents the results for problem F_4 . We observe that the use of downscaling substantially improves the out-of-sample performance for the noisiest problem formulations. This is evident in the case where the number of wake/sleep cycles was the greatest, i.e., 15 and 20. For the easy and moderate cases, figures suggest that downscaling has the tendency to worsen performance. This particular problem also exhibits an interesting trend in the evolution of the MSE curve as a function of the downscaling percentage. More specifically, the out-of-sample error decreases as the downscaling percentage increases for the noisiest problems, whereas it decreases as a function of increasing percentage of the small and moderate levels of noise in the target.

4.1. Summary of Observations

The observations from the experiments can be summarised as follows:

1. The downscaling mechanism increases the generalisation performance for most cases of moderate and high levels of noise.
2. No advantage is accruing from the proposed method when used with small levels of noise in the target function. In most cases, performance deteriorates.
3. The optimal number of wake/sleep cycles and the level of weight downscaling appears to be problem dependent. A principled approach such as cross-validation should be applied to choose these effectively.
4. Overall, when training and testing over similar input-output distributions, weight downscaling exerts a negative effect by disrupting the fit of a model. On the other hand, in the case where there is discrepancy between training and testing input-output distributions, the downscaling mechanism improves generalisation.

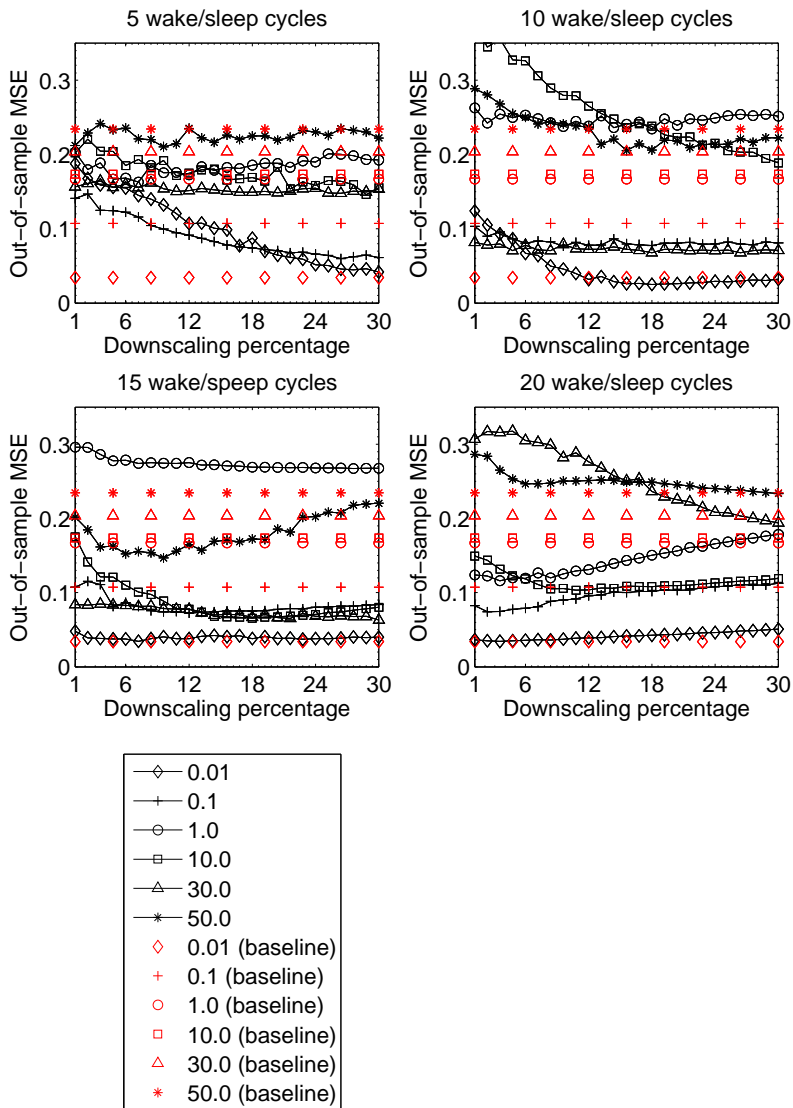


Figure 2. Out-of-sample results for problem F_2 with six different noise levels.

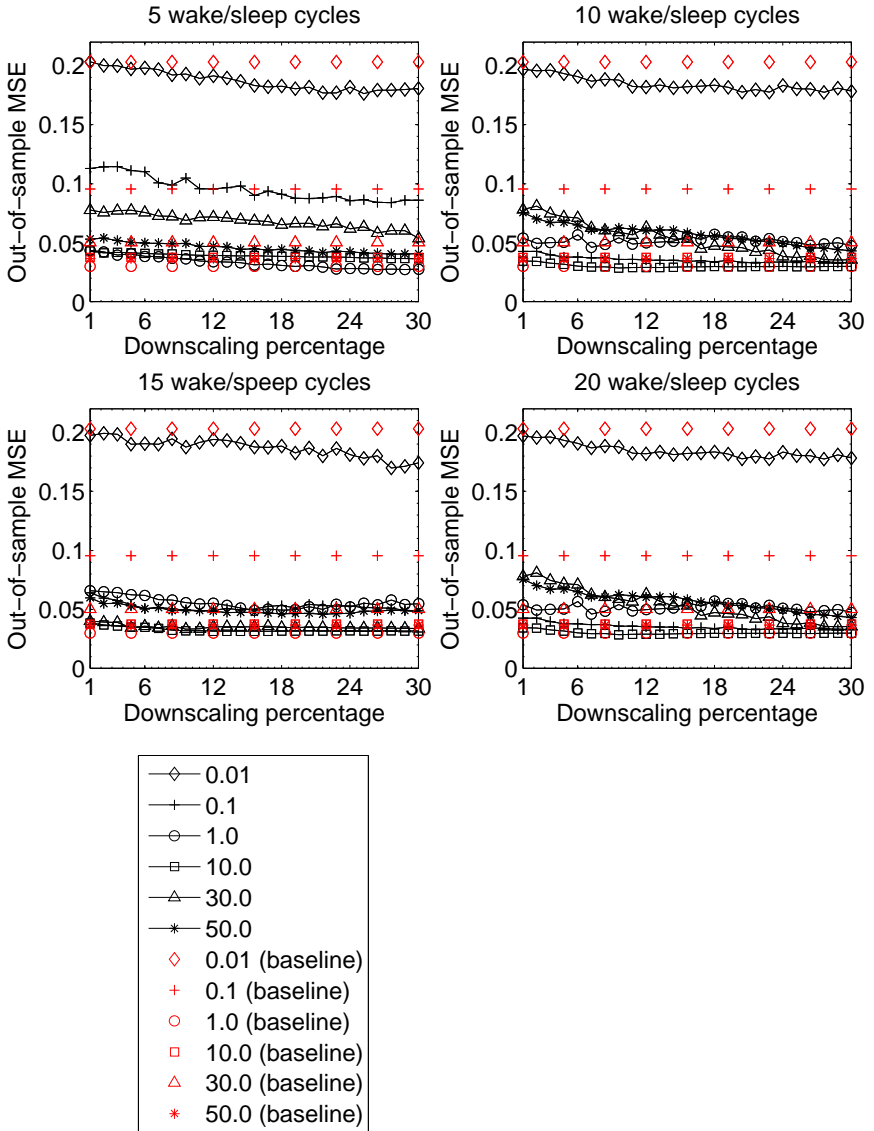


Figure 3. Out-of-sample results for problem F_3 with six different noise levels.

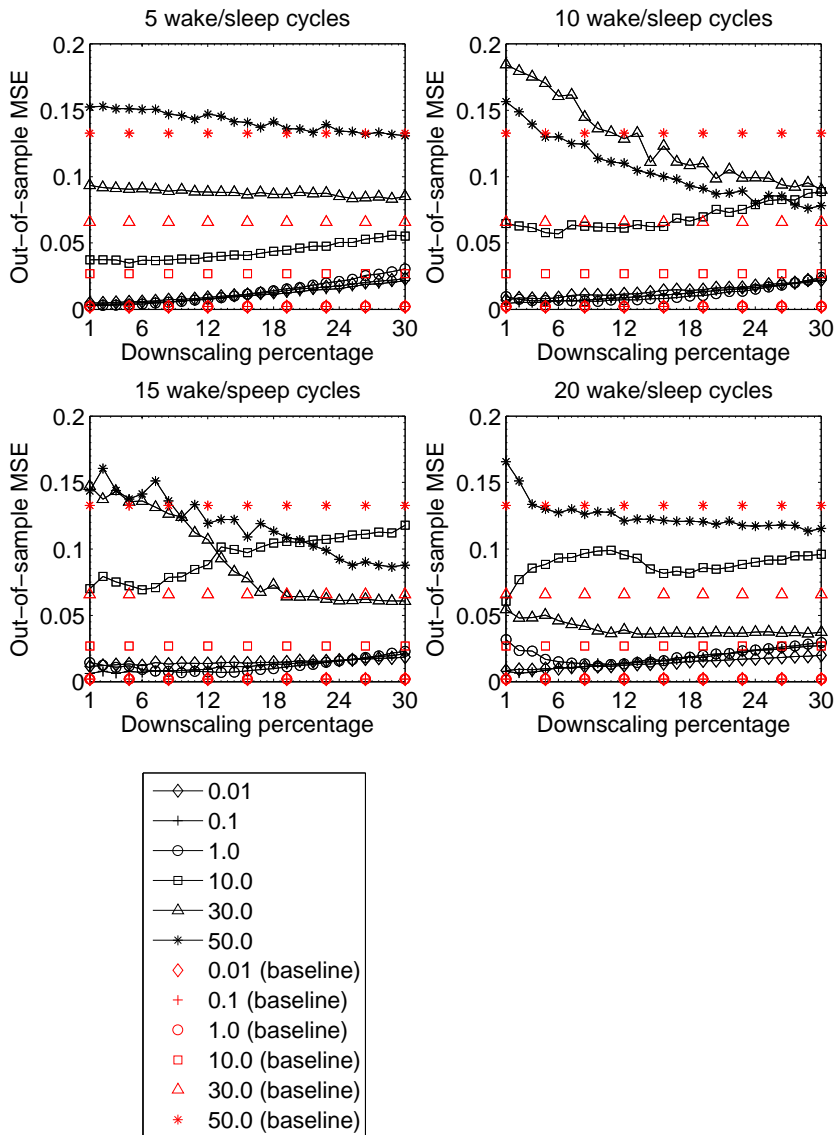


Figure 4. Out-of-sample results for problem F_4 with six different noise levels.

Table 2. Out of sample MSE: mean values and standard errors. Function 1.

Down	Noise 0.01	Noise 0.1	Noise 1.0	Noise 10.0	Noise 30.0	Noise 50.0
Standard MLP						
n/a	0.060 (0.0005)	0.077 (0.0011)	0.151 (0.0005)	0.195 (0.0012)	0.164 (0.0013)	0.172 (0.0009)
5 wake/sleep cycles						
1%	0.063 (0.0007)	0.068 (0.0016)	0.166 (0.0011)	0.194 (0.0028)	0.245 (0.0023)	0.169 (0.0036)
15%	0.062 (0.0005)	0.069 (0.0008)	0.162 (0.0013)	0.158 (0.0018)	0.252 (0.0022)	0.173 (0.0018)
30%	0.067 (0.0003)	0.074 (0.0009)	0.159 (0.0016)	0.158 (0.0009)	0.239 (0.0030)	0.160 (0.0010)
10 wake/sleep cycles						
1%	0.061 (0.0027)	0.077 (0.0018)	0.123 (0.0029)	0.128 (0.0029)	0.218 (0.0036)	0.160 (0.0049)
15%	0.084 (0.0038)	0.061 (0.0003)	0.127 (0.0023)	0.122 (0.0008)	0.201 (0.0020)	0.186 (0.0030)
30%	0.105 (0.0015)	0.069 (0.0003)	0.134 (0.0018)	0.135 (0.0012)	0.214 (0.0019)	0.181 (0.0017)
15 wake/sleep cycles						
1%	0.056 (0.0021)	0.064 (0.0014)	0.159 (0.0035)	0.121 (0.0032)	0.228 (0.0019)	0.154 (0.0045)
15%	0.078 (0.0032)	0.066 (0.0009)	0.139 (0.0005)	0.111 (0.0009)	0.232 (0.0019)	0.215 (0.0004)
30%	0.099 (0.0004)	0.077 (0.0008)	0.141 (0.0004)	0.111 (0.0004)	0.206 (0.0009)	0.198 (0.0003)
20 wake/sleep cycles						
1%	0.052 (0.0015)	0.078 (0.0008)	0.074 (0.0017)	0.182 (0.0044)	0.239 (0.0077)	0.167 (0.0022)
15%	0.074 (0.0004)	0.077 (0.0008)	0.072 (0.0004)	0.195 (0.0016)	0.243 (0.0027)	0.212 (0.0005)
30%	0.086 (0.0004)	0.101 (0.0005)	0.081 (0.0002)	0.188 (0.0005)	0.234 (0.0009)	0.206 (0.0002)

Table 3. Out of sample MSE: mean values and standard errors. Function 2.

Down	Noise 0.01	Noise 0.1	Noise 1.0	Noise 10.0	Noise 30.0	Noise 50.0
Standard MLP						
n/a	0.034 (0.0007)	0.107 (0.0019)	0.167 (0.0027)	0.174 (0.0057)	0.204 (0.0073)	0.234 (0.0041)
5 wake/sleep cycles						
1%	0.188 (0.0053)	0.141 (0.0041)	0.203 (0.0127)	0.202 (0.0117)	0.156 (0.0042)	0.212 (0.0126)
15%	0.087 (0.0040)	0.074 (0.0013)	0.186 (0.0072)	0.169 (0.0074)	0.149 (0.0026)	0.225 (0.0077)
30%	0.031 (0.0009)	0.064 (0.0022)	0.197 (0.0040)	0.149 (0.0036)	0.147 (0.0019)	0.226 (0.0058)
10 wake/sleep cycles						
1%	0.124 (0.0086)	0.103 (0.0043)	0.263 (0.0090)	0.375 (0.0141)	0.082 (0.0047)	0.289 (0.0115)
15%	0.025 (0.0008)	0.077 (0.0017)	0.234 (0.0030)	0.238 (0.0037)	0.068 (0.0027)	0.206 (0.0060)
30%	0.035 (0.0007)	0.083 (0.0005)	0.255 (0.0016)	0.169 (0.0014)	0.065 (0.0015)	0.223 (0.0017)
15 wake/sleep cycles						
1%	0.048 (0.0033)	0.107 (0.0053)	0.296 (0.0050)	0.175 (0.0087)	0.084 (0.0021)	0.202 (0.0095)
15%	0.038 (0.0012)	0.075 (0.0007)	0.269 (0.0004)	0.066 (0.0012)	0.067 (0.0026)	0.172 (0.0031)
30%	0.043 (0.0007)	0.087 (0.0006)	0.266 (0.0005)	0.086 (0.0005)	0.061 (0.0028)	0.218 (0.0008)
20 wake/sleep cycles						
1%	0.037 (0.0009)	0.083 (0.0026)	0.124 (0.0054)	0.149 (0.0062)	0.307 (0.0107)	0.286 (0.0064)
15%	0.042 (0.0004)	0.102 (0.0007)	0.150 (0.0007)	0.108 (0.0005)	0.236 (0.0024)	0.249 (0.0006)
30%	0.054 (0.0002)	0.117 (0.0009)	0.187 (0.0001)	0.122 (0.0003)	0.178 (0.0009)	0.229 (0.0005)

Table 4. Out of sample MSE: mean values and standard errors. Function 3.

Down	Noise 0.01	Noise 0.1	Noise 1.0	Noise 10.0	Noise 30.0	Noise 50.0
Standard MLP						
n/a	0.203 (0.0002)	0.095 (0.0003)	0.030 (0.0001)	0.037 (0.0001)	0.050 (0.0007)	0.036 (0.0001)
5 wake/sleep cycles						
1%	0.203 (0.0010)	0.113 (0.0037)	0.043 (0.0007)	0.043 (0.0005)	0.077 (0.0019)	0.053 (0.0010)
15%	0.182 (0.0026)	0.091 (0.0027)	0.031 (0.0004)	0.038 (0.0002)	0.065 (0.0022)	0.044 (0.0004)
30%	0.175 (0.0034)	0.083 (0.0015)	0.027 (0.0002)	0.035 (0.0002)	0.051 (0.0020)	0.041 (0.0005)
10 wake/sleep cycles						
1%	0.197 (0.0013)	0.042 (0.0009)	0.054 (0.0023)	0.034 (0.0008)	0.078 (0.0030)	0.075 (0.0026)
15%	0.183 (0.0019)	0.034 (0.0004)	0.057 (0.0018)	0.030 (0.0001)	0.047 (0.0025)	0.056 (0.0008)
30%	0.176 (0.0025)	0.034 (0.0007)	0.045 (0.0010)	0.030 (0.0002)	0.032 (0.0001)	0.040 (0.0004)
15 wake/sleep cycles						
1%	0.197 (0.0021)	0.064 (0.0015)	0.066 (0.0023)	0.038 (0.0013)	0.040 (0.0018)	0.060 (0.0019)
15%	0.188 (0.0032)	0.053 (0.0012)	0.049 (0.0011)	0.031 (0.0001)	0.035 (0.0004)	0.046 (0.0002)
30%	0.164 (0.0028)	0.045 (0.0003)	0.052 (0.0016)	0.032 (0.0001)	0.034 (0.0001)	0.051 (0.0004)
20 wake/sleep cycles						
1%	0.219 (0.0035)	0.051 (0.0013)	0.152 (0.0034)	0.063 (0.0022)	0.073 (0.0055)	0.063 (0.0029)
15%	0.228 (0.0006)	0.043 (0.0002)	0.075 (0.0004)	0.031 (0.0001)	0.029 (0.0001)	0.042 (0.0003)
30%	0.208 (0.0004)	0.034 (0.0000)	0.063 (0.0003)	0.030 (0.0000)	0.029 (0.0001)	0.039 (0.0002)

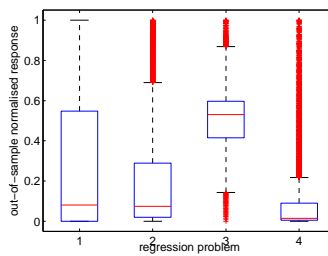
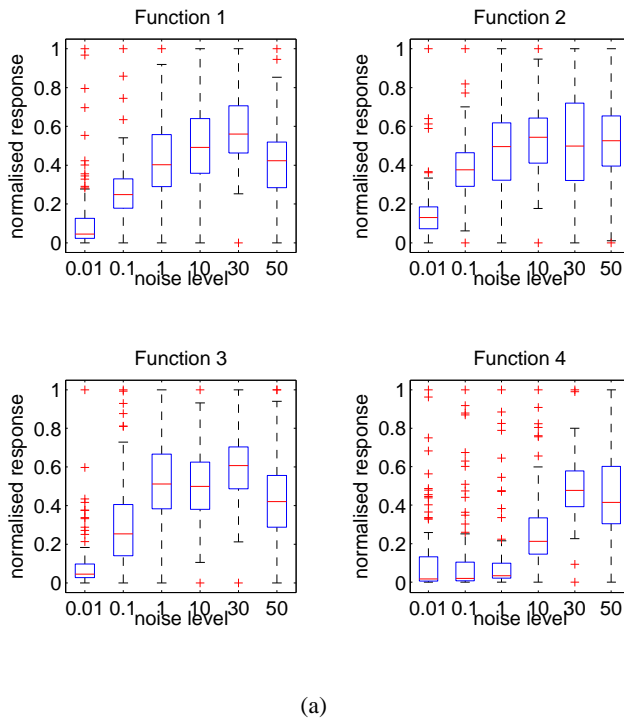


Figure 5. (a) Histogram of in-sample normalised values of the response variable for different regression problems of Table 1. (b) Histogram of out-of-sample normalised values of the response variable for different regression problems of Table 1.

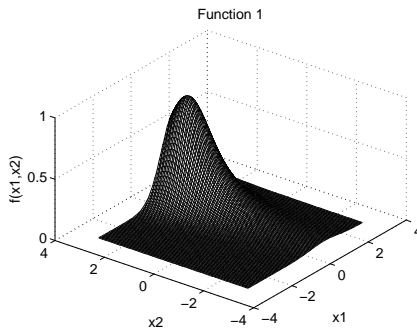


Figure 6. Plot of regression problem 1 of Table 1.

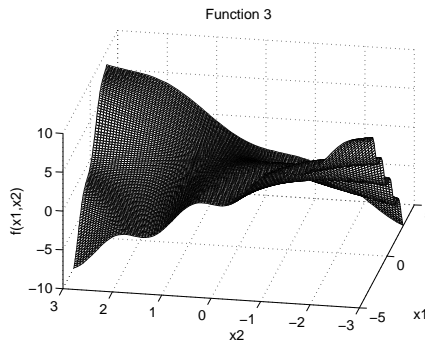


Figure 7. Plot of regression problem 3 of Table 1.

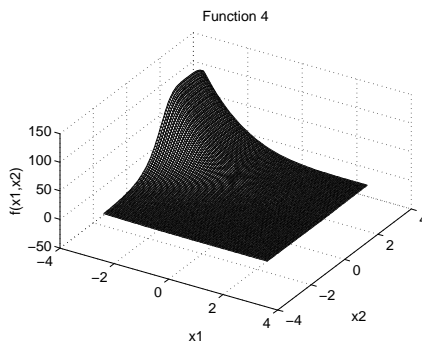


Figure 8. Plot of regression problem 4 of Table 1.

Table 5. Out of sample MSE: mean values and standard errors. Function 4.

Down	Noise 0.01	Noise 0.1	Noise 1.0	Noise 10.0	Noise 30.0	Noise 50.0
Standard MLP						
n/a	0.002 (0.0000)	0.002 (0.0001)	0.002 (0.0001)	0.027 (0.0001)	0.066 (0.0003)	0.133 (0.0002)
5 wake/sleep cycles						
1%	0.005 (0.0001)	0.004 (0.0001)	0.003 (0.0001)	0.037 (0.0011)	0.093 (0.0006)	0.152 (0.0027)
15%	0.013 (0.0001)	0.011 (0.0003)	0.014 (0.0002)	0.044 (0.0009)	0.087 (0.0010)	0.141 (0.0021)
30%	0.028 (0.0005)	0.027 (0.0006)	0.041 (0.0007)	0.065 (0.0018)	0.087 (0.0017)	0.124 (0.0025)
10 wake/sleep cycles						
1%	0.007 (0.0005)	0.004 (0.0002)	0.010 (0.0011)	0.065 (0.0028)	0.184 (0.0040)	0.156 (0.0038)
15%	0.014 (0.0004)	0.012 (0.0002)	0.010 (0.0003)	0.066 (0.0018)	0.108 (0.0033)	0.093 (0.0016)
30%	0.024 (0.0003)	0.026 (0.0003)	0.032 (0.0007)	0.095 (0.0018)	0.085 (0.0020)	0.083 (0.0028)
15 wake/sleep cycles						
1%	0.012 (0.0008)	0.006 (0.0008)	0.014 (0.0021)	0.070 (0.0026)	0.147 (0.0041)	0.144 (0.0057)
15%	0.015 (0.0003)	0.013 (0.0009)	0.010 (0.0002)	0.104 (0.0019)	0.073 (0.0026)	0.113 (0.0044)
30%	0.021 (0.0001)	0.024 (0.0002)	0.030 (0.0004)	0.122 (0.0012)	0.062 (0.0008)	0.090 (0.0015)
20 wake/sleep cycles						
1%	0.008 (0.0003)	0.008 (0.0004)	0.032 (0.0023)	0.061 (0.0027)	0.054 (0.0034)	0.166 (0.0065)
15%	0.015 (0.0003)	0.019 (0.0007)	0.018 (0.0002)	0.082 (0.0013)	0.036 (0.0003)	0.121 (0.0012)
30%	0.022 (0.0001)	0.030 (0.0001)	0.034 (0.0001)	0.098 (0.0008)	0.036 (0.0001)	0.115 (0.0010)

5. Conclusion

In prediction problems, fitting the training data too closely can be counterproductive. Reducing the expected loss on the training data beyond some point causes the population-expected loss to stop decreasing, and often start to increase. Regularisation methods in MLPs, like weight decay, prevent such overfitting by constraining the magnitude of the adaptive weights during the learning phase. In the chapter we showed that simulating a simple weight downscaling mechanism during a sleep phase can, similarly to weight decay, exert a positive effect on generalisation in the case of noisy datasets.

Controlling the parameters defined as the *downscaling percentage* and the *number of wake/sleep cycles* regulates the degree to which the expected loss on the training data is minimised. Each of the two parameters controls the degree-of-fit and thus values for each of these parameters interact. Decreasing the value of downscaling percentage, increases the best value for the wake/sleep cycles. Ideally, one should estimate optimal values for both by minimising a model selection criterion jointly with respect to the values of the two parameters. There are also computational considerations; increasing the value of sleep/wake cycles produces a proportionate increase in the computation. Its value should be made as large as is computationally feasible. The value of downscaling percentage should then be adjusted using cross-validation.

A final observation concerns the nature of the learning process as realised via a number of sleep/wake cycles. Unlike fitting the weights of the network during a number of epochs with a fixed learning rate, the sleep/wake approach instead *learns more slowly*. In general, it has been repeatedly advocated in the statistical machine learning literature that learning methods that learn slowly tend to generalise well.

Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number 08/SRC/FM1389.

References

- [1] Axmacher, N., Draguhn, A., Elger, C., Fell, J. (2009). Memory processes during sleep: beyond the standard consolidation theory. *Cellular*

-
- and Molecular Life Sciences*, 66:2285–2297.
- [2] Cirelli, C. (2009). The genetic and molecular regulation of sleep: from fruit flies to humans. *Nature Reviews Neuroscience*, 10:549–560.
- [3] Diekelmann, S., Born, J. (2010). The memory function of sleep. *Nature Reviews Neuroscience*, 11:114–126.
- [4] Greenspan, R., Tononi, G., Cirelli, C., Shaw, P. (2001). Sleep and the fruit fly. *Trends in Neurosciences*, 24(3):142–145.
- [5] Hardt, O., Nader, K., Nadel, L. (2013). Decay happens: the role of active forgetting in memory. *Trends in Cognitive Sciences*, 17(3):111–120.
- [6] Hinton, G., Dayan, P., Frey, B., Neal, R. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.
- [7] James, W. (1890). *Principles of Psychology*, Holt, New York.
- [8] Karnin, E. (1990). A Simple Procedure for Pruning Back-Propagation Trained Neural Networks. *IEEE Transactions on Neural Networks*, 1(2):239–242.
- [9] Krogh, A., Hertz, J. (1995). A Simple Weight Decay Can Improve Generalization. In: *Advances in Neural Information Processing Systems 4*, pp. 950–957, Morgan Kaufmann, San Mateo.
- [10] LeCun, Y., Denker, J., Solla, S., Howard, R., Jackel, L. (1989). Optimal brain damage. In: *Advances in Neural Information Processing Systems (NIPS)*, vol. 2, pp. 598–605, Morgan Kaufmann, San Mateo.
- [11] Lima, S., Rattenborg, N., Lesku, J., Amlaner, C. (2005). Sleeping under the risk of predation. *Animal Behaviour*, 70:723–736.
- [12] Majumdar A, Cesario WC, White-Grindley E, Jiang H, Ren F, Khan M, Li, L, Choi E, Kannan K, Guo F, Unruh J, Slaughter B, Si K. (2012) Critical role of amyloid-like oligomers of *Drosophila* Orb2 in the persistence of memory. *Cell* 148: 515529. doi: 10.1016/j.cell.2012.01.004.
- [13] Marr, D. (1971). Simple memory: a theory for archicortex, *Philosophical Transactions of the Royal Society of London, Series B*, 262:23–81.

- [14] McGaugh, J. (2000). Memory—a Century of Consolidation. *Science*, 287(5451): 248–251.
- [15] Müller, G., Pilzecker, A. (1900). Experimentelle Beiträge zur Lehre vom Gedächtniss. *Zeitschrift für Psychologie*. Ergänzungsband.
- [16] Nadel, L. and Moscovitch, M. (1997). Memory consolidation, retrograde amnesia and the hippocampal complex, *Current Opinion in Neurobiology*, 7:217–227.
- [17] Plaut, D., Nowlan, S., Hinton, G. (1986). Experiments on Learning by Back Propagation. *Technical Report Computer Science Department (CMU-CS-86-126)*, Carnegie-Mellon University, Pittsburgh.
- [18] Rattenborg, N., Amlaner, C., Lima, S. (2000). Behavioral, neurophysiological and evolutionary perspectives on unihemispheric sleep. *Neuroscience and Biobehavioral Reviews*, 24:817–842.
- [19] Rattenborg, N., Martinez-Gonzalez, D., Lesku, J. (2009). Avian sleep homeostasis: Convergent evolution of complex brains, cognition and sleep functions in mammals and birds. *Neuroscience and Biobehavioral Reviews*, 33:253–270.
- [20] Rudy, J. (2014). *The Neurobiology of learning and memory* (2nd ed), Sinauer Associates Inc., Sunderland Massachusetts.
- [21] Schulz, H. (2008). Rethinking sleep analysis. Comment on the AASM (American Academy of Sleep Medicine) Manual for the Scoring of Sleep and Associated Events. *Journal of Clinical Sleep Medicine*, 4(2): 99–103.
- [22] Sejnowski, T., Destexhe, A. (2000). *Why do we sleep?* *Brain Research*, 886:208–223.
- [23] Siegel, J. (2008). Do all animals sleep? *Trends in Neuroscience*, 31(4):208–213.
- [24] Siegel, J. (2009). Sleep viewed as a state of adaptive inactivity. *Nature Reviews Neuroscience*, 10:747–753.
- [25] Sietsma, J., Dow, R. (1989). Neural net pruning—why and how. In: *Proceedings of 1988 IEEE International Conference on Neural Networks*, 1:325–333, IEEE Press.

-
- [26] Sietsma, J., Dow, R. (1991). Creating artificial neural networks that generalise. *Neural Networks*, 4(1):67–79.
- [27] Squire, L. (1987). *Memory and the Brain*, Oxford University Press.
- [28] Stickgold, R. (1998). Sleep: off-line memory reprocessing. *Trends in Cognitive Science*, 2(12):484–492.
- [29] Tononi, G., Cirelli, C. (2003). Sleep and synaptic homeostasis: a hypothesis. *Brain Res Bull.* 62(2):143–150.
- [30] Tononi, G., Cirelli, C. (2006). Sleep function and synaptic homeostasis. *Sleep Med Rev.* 10(1):49–62.
- [31] Tononi, G., Cirelli, C. (2013). Perchance to Prune. *Scientific American*, 309:34–39.
- [32] Teyler, T. and DiScenna, P. (1986). The hippocampal memory indexing theory, *Behavioral Neuroscience*, 100:147–152.
- [33] Weigend, A., Rumelhart, D., Huberman, B. (1991) Generalization by weight-elimination with application to forecasting. In: R. P. Lippmann, J. Moody, & D. S. Touretzky (eds.), *Advances in Neural Information Processing Systems* 3, pp. 875–882, San Mateo, CA, Morgan. Kaufmann.
- [34] White-Grindley E, Li L, Mohammad Khan R, Ren F, Saraf A, Florens L, Si K. (2014). Contribution of Orb2A Stability in Regulated Amyloid-Like Oligomerization of Drosophila Orb2. *PLoS Biol* 12(2): e1001786. doi:10.1371/journal.pbio.1001786.
- [35] Yang, G., Lai, C. S., W., Cichon, J., Ma, L., Li, W., Gan, W-B. (2014). Sleep promotes branch-specific formation of dendritic spines after learning. *Science*, 344(6188):1173–1078.

Chapter 3

PLANT PROPAGATION-INSPIRED ALGORITHMS

A. Brabazon^{1,*}, *S. McGarraghy*^{1,†} and *A. Agapitos*^{2,‡}

¹Complex Adaptive Systems Laboratory and School of Business

University College Dublin, Dublin, Ireland,

²Complex Adaptive Systems Laboratory and

School of Computer Science and Informatics

University College Dublin, Dublin, Ireland

Abstract

Plants represent some 99% of the eukaryotic biomass of the planet and have been highly successful in colonising many habitats with differing resource potential. The success of plants in "earning a living" suggests that they have evolved robust resource capture mechanisms and reproductive strategies. In spite of the preponderance of plant life, surprisingly little inspiration has been drawn from plant activities for the design of optimisation algorithms.

In this chapter we focus on one important aspect of plant activities, namely seed and plant dispersal. Mechanisms for seed and plant dispersal have evolved over time in order to create effective ways to disperse seeds into locations in which they can germinate and become established. These

*E-mail address: anthony.brabazon@ucd.ie

†E-mail address: sean.mcgarrahy@ucd.ie@ucd.ie

‡E-mail address: alexandros.agapitos@ucd.ie

mechanisms are highly varied, ranging from morphological characteristics of seeds which can assist their aerial or animal-mediated dispersion, to co-evolved characteristics which "reward" animals or insects who disperse a plant's seeds. At a conceptual level, dispersal can be considered as a "search process", wherein the seed or plant is searching for "good" locations and therefore, inspiration from dispersal activities of plants can plausibly serve as the design inspiration for optimisation algorithms.

Initially, we provide an overview of relevant background on the seed dispersal process from drawing on the ecology literature. Then we describe a number of existing optimisation algorithms which draw inspiration from these processes, and finally we outline opportunities for future research.

1. Introduction

The key imperative of a plant's life is to maximise its number of viable offspring [11]. Many species of plants reproduce by producing seeds and then dispersing these in the landscape. The seeds are in essence embryonic plants, enclosed in a protective coat, usually with some stored food in order to provide energy for the germination process. The technical term for the dispersed unit is a *diaspore* and this may consist of a seed, spore or fruit containing seeds, plus any additional tissue which assists in dispersal. In this paper we employ the term seed in a broad sense to encompass all of these cases.

If the seeds find a suitable location, they germinate and in turn reproduce themselves. Hence, the process of seed dispersal plays a critical role in ensuring the long-term success of a plant species and is the predominant process by which plants can 'move around' a landscape [18].

1.1. Dispersal Mechanisms

Plants make use of multiple dispersal mechanisms, including:

1. wind dispersal,
2. animal dispersal,
3. water dispersal, and
4. ballistic dispersal.

Hence, dispersal mechanisms can be classed as abiotic (wind, water or gravity) or biotic (insect or animal dispersal). Many plants use more than one dispersal mechanism and dispersal can take place in stages. For example, wind dispersed seeds can subsequently be redispersed by ants or seed hoarding rodents.

Morphological adaptations in plants and seeds have arisen over time in order to increase the efficiency of seed dispersal. In the case of wind dispersal, seeds which have characteristics such as small size, wings, hairs etc. fall more slowly, essentially by lowering their wing loading (ratio of mass to surface area), and this promotes wider seed dispersal. Species with these adaptations are very common, comprising some 10-30% of all plants, and up to 70% of the flora in temperate plant communities [18]. Wind dispersed plants are common in dry habitats such as deserts [9]. An interesting example of this is provided by tumble weeds where the plant shoot dies and detaches from the root system. The seeds attached to the upper part of the plant are then dispersed as it is blown around the landscape. Some curious adaptations have emerged in order to promote the effectiveness of wind dispersal mechanisms whereby a plant manipulates its environment in order to 'generate' a local wind current in order to assist dispersal. One example is provided by the spores of ascomycete fungi where by synchronising the ejection of thousands of spores, the fungi create a flow of air that carries their spores further than they would otherwise disperse [37]. Another example is provided by oyster and shiitake mushrooms which release water vapour before releasing their spores which in turn cools the surrounding air creating convection currents thereby helping to disperse their spores [36].

Some curious adaptations have emerged in order to promote the effectiveness of wind dispersal mechanisms whereby a plant manipulates its environment in order to 'generate' a local wind current in order to assist dispersal. One example is provided by the spores of ascomycete fungi where by synchronising the ejection of thousands of spores, the fungi create a flow of air that carries their spores further than they would otherwise disperse [37]. Another example is provided by oyster and shiitake mushrooms which release water vapour before releasing their spores which in turn cools the surrounding air creating convection currents thereby helping to disperse their spores [36].

Adaptations for animal dispersal include the offering of 'rewards' for dispersion, such as fleshy, nutritious, fruits which attract the attention of frugivores (fruit eaters) who consume the fruit. The seeds contained in the fruit pass through the digestive tract of the animal and are eventually excreted back into the environment. This means of seed dispersal is common with some 50-75%

of tree species in tropical forests producing fleshy fruits adapted for animal consumption [9]. A similar figure is quoted by [30] who notes that 75% of tropical tree species display adaptations for biotic seed dispersal. Other (non-reward) adaptations for animal dispersal include clinging structures such as hooks or resin whereby seeds stick to fur or feathers of animals and are accordingly dispersed as the animal moves around the environment (this mechanism led to the discovery of Velcro in 1948, inspired by the observation of seed burrs sticking to the hair of a dog [20]). Many types of animals are seed dispersers including various species of mammals, birds, bees, fish and reptiles [10, 30]. One example of such dispersal is provided by ants. It is estimated that more than 10,000 plant species have evolved mechanisms to assist dispersal of their seeds by ants [31]. Typically the ants are attracted using by an elaiosomes, or fleshy structure, attached to the seed which is rich in lipids and proteins. The elaiosome and attached seed is taken to the nest to feed larvae and the seed is then discarded and later germinates. Animals and insects can also play a role as secondary dispersers. For example, ants and dung beetles can transport seeds which have fallen from plants.

Apart from wind and animal dispersal, seeds can also be dispersed by water, for example via buoyant coconuts. Some plant species have evolved ballistic fruits that open explosively and can toss seeds several metres from the parent plant. In this chapter we employ the term seed in a broad sense to encompass all of these cases.

1.2. Why Do Plants Disperse Their Seeds?

An obvious question given the wide range of strategies adopted by plants to disperse their seeds is what evolutionary advantages accrue to plants from their investment in dispersal structures? Such investments only make sense if dispersing seeds leads to a higher rate of seed survival and a higher rate of subsequent establishment. Three hypotheses are usually proposed to support the adaptive nature of seed dispersal [9], namely the:

1. escape hypothesis, the
2. colonisation hypothesis, and the
3. directed-dispersal hypothesis.

The core of the *escape hypothesis* is the claim that seeds which are dispersed further from their parent have higher rates of survival and reproductive success. In other words, if seeds were only dispersed in close proximity to their parent, their rates of mortality would be higher, due to density-dependent mortality factors such as insect / rodent predators which would be attracted to clusters of ‘target plants’, susceptibility to pathogen attack, and resource competition from other seedlings. Another factor which could promote dispersal is ‘shade escape’ as a non-dispersed seed would end up competing directly with their parent for light and other resources. In a study of 34 tree species, [1] found that seeds from species requiring light-gaps for early seedling survival had slower rates of descent, enhancing their chances of escape from the light shadow of their parent.

The *colonisation hypothesis* notes that habitats and environments change over time, and a currently resource poor environment may subsequently become more abundant. Hence, seeds which reach this environment, perhaps remaining dormant initially, will be well-placed to germinate and colonise the area if conditions later improve. This hypothesis underscores the fact that seed dispersal can be temporal as well as spatial, as some seeds can remain in a dormant condition for considerable periods awaiting better conditions. Dormancy capability is valuable, as it can notably increase the reproductive success of the parent plant [33].

The *directed dispersal hypothesis* [9] argues that plants can adapt their diaspores and / or their morphology in order to enhance their chances of dispersing seeds into locations which provide good conditions for seed establishment and growth. For example, plants can adapt their morphology in order to utilise differing seed dispersing agents. Non-random dispersal into resource rich environmental patches presents an obvious evolutionary advantage over random seed dispersal methods [26, 32].

1.3. Design Trade-Offs

Plants can exert some control over their seed dispersal patterns as morphological factors such as plant height, fruit / seed size and design, and ease of abscission (release of fruit/seed) are all adaptable over time.

Taking plant height, a taller plant can produce a wider seed shadow via wind dispersal than a low-sized plant. Of course, a greater degree of tissue investment is required to grow a taller plant, leaving less energy for seed production, potentially creating a design trade-off.

In the case of seed design, plants can select different levels of investment in their seeds, with some plants adopting a ‘low investment’ model, where the plant invests little in individual seeds but produces a large number of them, with other plant species adopting a ‘high investment’ model, producing fewer, larger, seeds. A larger seed can contain greater energy reserves thereby enhancing the probability of germination but larger seeds are usually harder to disperse than smaller ones, requiring larger animals, stronger winds or more powerful propulsion mechanisms [34]. Hence, larger seed size will impact on the design of the plant’s dispersal mechanisms.

The level of investment in fruit production (for fruiting plants) can also be adapted as production of richer, more attractive, fruits will enhance biotic seed dispersal but at the expense of leaving less energy for other plant requirements.

In essence, when ‘selecting’ a dispersal mechanism, two costs are being balanced, the cost of seed mortality (arising when seeds produced by a plant fail to subsequently germinate), and the allocation costs (i.e., the costs of that dispersal mechanism. In summary, plants can employ a wide variety of seed dispersal techniques, each requiring different levels of resource investment, and each requiring differing plant morphologies which embed specific trade-offs.

1.4. Structure of Paper

The remainder of this chapter is organised as follows. Section 2 provides some background on aspects of the seed dispersal process Section 3 outlines a number of optimisation algorithms whose design has drawn inspiration from the plant propagation process. Conclusions and opportunities for future work are discussed in Section 5

2. Background

As the seed dispersal pattern of plants is important both for individual and species-level survival, a significant research effort has been expended in order to gain insight into the dispersal patterns for various plant species. Levey et al (2008) [13] notes that the ‘Holy grail of seed dispersal is to accurately predict the probability distribution of seed density from a particular configuration of parents and then relate those distributions to seedling demography’ (p. 604).

The spatial distribution of seed dispersal from an individual plant, or cluster of plants, is known as a *seed shadow*. More formally, these seed shadows can

be represented by a probability distribution, relating the probability that an individual seed is dispersed a given distance from its maternal plant. Spatial dispersal patterns can be considered either in one dimensional terms, focussing on dispersal distance, or in two dimensions by also considering the directionality of dispersion. Both dispersal distance and direction for an individual plant will be impacted by the nature of the plant's dispersal mechanism and by location-specific factors.

Ballists and ant-dispersed seeds tend to travel the shortest distances (up to a few metres typically), with wind-dispersal and animal dispersal producing greater dispersal distances in terms of both mode and maxima. Directional dispersal can be influenced by several factors, the most obvious of which is prevailing wind direction in the case of wind-dispersed seeds [34]. The directionality of animal dispersed seeds will be influenced by the topology of the local environment as this will impact on animal movement patterns.

A practical issue that arises in attempting to capture empirical data on seed dispersal is that long-range dispersal events tend to be under-reported as it becomes difficult to accurately attribute seeds to specific parent plants as seeds disperse over increasing distances. For example, extreme distance dispersal events, such as may occur when seeds get stuck to the feathers or feet of birds are unlikely to be captured in empirical studies. The problem of capturing good data on long-dispersal events is noted by many studies, with [5] pithily stating that 'for [dispersal] distances exceeding a few hundred metres we essentially know nothing'. However, there have been some attempts to construct general frameworks of long-distance dispersal [7] in order to facilitate the construction and testing of the biogeographical consequences of long-distance dispersal. Understanding long-distance dispersal of seeds is of critical importance in gaining insight into the spread of plant populations (including invasive species), and in explaining the diversity and dynamics of ecological communities [4].

Another perspective on seed dispersion is that it can be considered as taking place across time as well as spatially [7]. An obvious example is the case of long-distance dispersal whereby a seed or spore may be dispersed by rafting on ocean flotsam, and take many days to reach its final destination. More generally, seed germination and spore revival may be long delayed awaiting suitable environmental conditions and thus we can distinguish between seeds germinating from a *seed bank* (seeds dispersed in the past which have lain in the soil) and *seed rain* (recently deposited new seeds arising from current dispersal).

2.1. Modelling Seed Dispersal

Two main approaches have been taken to modelling of seed dispersal patterns, a conceptual approach which attempt to build a model from the underlying physical mechanisms of dispersal, and an empirical approach which seeks to reverse fit a mathematical model to real-world data.

In seeking to build a model of seed dispersal, [13] notes that an important distinction must be made between cases where the seeds are dispersed *abiotically*, for example by wind, and cases where seeds are dispersed *biotically*, for example, by animals or insects. In the former case, the focus is on parameterising a mechanistic seed dispersal model, accounting for plant height, characteristics of the seed structure, wind conditions etc. In the latter case, the situation is more complex, and it is necessary to consider factors governing animal movement, animal physiology, and animal behaviour. Initially thought to be infrequent, reports of such directed dispersal by animals are increasing, as more detailed studies of the food caching behaviours of animals are undertaken [35].

2.1.1. Modelling Wind Dispersal

The earliest studies which attempted to construct a model of wind borne dispersal of seeds used a ballistic formulation, considering seeds to be non-powered projectiles [8]:

$$x = \frac{Hu}{F} \quad (1)$$

where x is the predicted horizontal distance from maternal parent to the deposition site, H is seed release height above the ground, F is a constant descent velocity, and u is the horizontal wind velocity averaged between H and the ground. The basic ballistic model assumes that the dispersed seed reaches terminal velocity (the falling velocity of a seed in still air) immediately after release, and that horizontal wind velocity is constant during the descent phase.

Although this model is a simplification of reality, it highlights that there will be a variation in the deposition distance depending on the wind speeds in the downwind, crosswind and vertical directions, the terminal velocity of the seed, and its release height. For example, a low terminal velocity, such as would arise with a lightweight or an aerodynamic seed structure, will enhance dispersal distance as there is more chance of an uplift eddy with consequent horizontal displacement during the lengthier ‘descent’ process. The model also illustrates

that the detachment mechanism from a plant is important as this determines the minimum level of wind speed which will act on the seed when it is detached from the plant.

A shortcoming of these models is that they produce seed dispersal estimates which have far lower maximum dispersal distances than are seen in the real world. A more realistic model can be obtained if variable windspeeds are incorporated, with turbulent fluctuations in the vertical velocity component. Simulations using these models produces dispersal distributions which are more realistic, producing maximum seed dispersal distances that are two to three orders of magnitude bigger than those produced by simple ballistic models. These distributions can be approximated by a power law dispersal kernel [18].

At a macro level, it may be possible to model long distance wind dispersal as storms, trade winds and high-altitude jet streams are at least partly predictable on longer time scales in terms of direction, time of year, and typical wind speeds [7].

2.1.2. Modelling Animal Dispersal

As animals are important seed dispersal vectors, knowledge of animal movement patterns and animal physiology could contribute to our understanding of seed dispersal distribution. Recent years have seen the development of the new multi-disciplinary field of *movement ecology* [29]. This field is concerned with empirical and theoretical study into the movement of animals, plants or microorganisms. Areas of interest include movement phenomena surrounding foraging and seasonal migration.

The simplest models of animal foraging movement ignore cognition and sensory inputs, corresponding to a case where resources are randomly dispersed and cognition and sensory capabilities are either non-existent or alternatively, too limited to effectively aid the search process. In this case, foraging movement can be modelled as being a random walk. The best-known random walk models assume Brownian motion and it was long thought that this could be used to approximate the diffusion of biological organisms. In turn, due to the Central Limit Theorem whereby the distribution of the sum of i.i.d. random variables with finite variance converges to a Gaussian, this would produce a normal distribution for multi-step foraging expeditions [29].

However, the assumption of Brownian motion ignores important aspects of real-world foraging including the ‘directional persistence’ typically exhibited

by organisms. Animals rarely undertake 180 degree turns and revisit a just-sampled site. Animals also do not blindly move around the environment but rather stop when a resource is found, nor do they tend to persist in searching a ‘patch’ in the environment which has been unfruitful in the recent past.

Movements of animals might therefore be expected to display ‘fat tails’ having a greater number of very short and very long ‘jumps’ than would be expected under a Brownian motion assumption. When tested using empirical data from foraging organisms, the results indicate that, particularly in cases where resources are sparsely and randomly distributed, the foraging movements of many organisms are described as a Lévy flight, giving rise to the *Lévy flight foraging hypothesis* [28]. A Lévy flight is a random walk in which the step-lengths (jump sizes) have a power law distribution.

We may also consider a slightly more complex foraging model where resources are randomly distributed in the environment and the forager is allowed to have sensory perception, such as the ability to ‘see’ or ‘smell’ food resources and move accordingly. In this case, the animal behaves as follows [27]:

- i. if there is a resource located within a direct vision distance r_v then the searcher detects it with certain probability and moves on a straight line to the detected resource;
- ii. if there is no detected resource within distance r_v then the searcher chooses a direction at random and a distance l_j from a probability distribution and moves incrementally to the new point constantly looking for resources within a distance r_v along the way;
- iii. if it does not detect any resources, it stops after traversing distance l_j and chooses a new direction and distance l_{j+1} , otherwise it moves to the resource;

where the probability distribution for move distances is a Lévy distribution, as follows:

$$P(l_j) \sim l_j^{-\mu} \quad (2)$$

Analysis in [27] suggests that in the absence of a priori knowledge of the distribution of food resources, the optimal strategy for a forager is to choose $\mu \approx 2$. The study notes that several empirical studies of foraging behaviour across a range of organisms (micro organisms, insects, birds, mammals) have been found to follow a Lévy distribution of flight lengths or times with $\mu \approx 2$.

Although the above analysis ignores a number of important issues concerning real-world foraging movement such as personal and social learning, environments in which resources are patchy, and local environment topology, it provides some support for a claim that the foraging movement patterns of animals will produce a leptokurtic pattern of seed dispersal.

An additional physiological factor in animal-mediated dispersal is the length of time the seed is carried by the animal before dispersal. Some animals such as birds will typically excrete ingested seeds within a few hours of consumption, in other cases, the digestion passage time may be considerably longer, 3-17 days in the case of some species of tortoises [10]. Seed morality may also vary depending on the animal that ingests them, although in the case of tortoises, less than 5% of seeds were found to be damaged whilst in transit through the digestive tract [10].

Animal-mediated seed dispersal is a complex animal-plant interaction which can take multiple forms, including cases where seeds commence germination whilst in the digestive tract of an animal. Other examples include the caching of seeds by animals in areas suitable for seed establishment and survival. Initially, thought to be infrequent, reports of such directed dispersal by animals are increasing, as more detailed studies of the food caching behaviours of animals are undertaken [35]. Examples include cases where animals cache seeds in areas of suitable soil conditions for seed growth, and caching of seeds at an optimal depth for their survival. Such synergistic interactions are plausible, as an ecology in which both plants and animals thrive is beneficial to both.

The social environment of animals also impacts on seed dispersal. Some mammals and birds live in groups, and hence defecate collectively at their feeding and resting sites. In turn, this will result in more localised dispersal of seeds than would occur if the seed consumers were solitary.

Due to the number of relevant factors, and our imperfect understanding of animal behaviours, it is clear that developing a comprehensive model of animal movements, which could then feed into a model of animal-mediated seed dispersal, is a non-trivial task. However, we can expect to see continued attempts to develop such models as the field of movement ecology develops.

2.2. Modelling

2.2.1. Empirical Modelling

An alternative approach to the modelling of seed dispersal patterns is to concentrate on empirical data rather than attempting to construct an explanatory model using underlying physical mechanisms. Empirical examination of the relationship between the number of seeds dispersed and distance from parent plant, indicates a leptokurtic distribution, displaying a higher peak and a heavier tail than a Gaussian distribution, with seed numbers decreasing monotonically with distance from the parent plant [9, 19, 34]. In attempting to reverse engineer a seed distribution function from observed seed count data, the aim is to uncover a probability density function $p(x)$ which gives the probability that a dispersed seed arrives at a distance x away from the source plant. This defines a dispersal kernel which maps seed density to distance (one dimensional case), or seed density by unit area to distance (two dimensional case) [18]. Typical kernels seen in the literature are Gaussian, negative exponential, and the inverse power function. A negative exponential model will have the general form [3]:

$$S_D = a_1 \cdot \exp(-b_1 \cdot D) \quad (3)$$

where S_D is the density of seeds at distance D from the source and a_1 and b_1 are constants indicating the density of seeds falling at the source and the slope of the decline in seed density with distance. In contrast, an inverse power model produces longer, fatter, tails:

$$S_D = a_2 \cdot (D)^{-b_2} \quad (4)$$

Based on sample of 73 herbaceous species and 75 tree/shrub samples [33] indicates that a negative exponential distribution provides a reasonable fit to the data, noting that the fit was better for the part of the curve around the mode with the tail area being less well-explained by this distribution. The study also noted that many empirical investigations stop collecting data on seed dispersal long before the end of the right tail and hence, their results need to be read with caution.

Differing studies have examined the seed dispersal patterns of numerous plant species and these have produced varying suggestions as to whether a negative exponential or an inverse power model (negative power distribution) produces a better fit to the collected data [26]. The relative scarcity of data

on long-range dispersal of seeds can make it difficult to distinguish between alternative model specifications.

In an attempt to better explain the tail of dispersal curves, some authors including [3] have suggested the use of a mixed model formulation with two kernel components: with a fat-tailed kernel for long distance dispersal, and negative-exponential component for short-distance dispersal:

$$S_D = a_3 \cdot \exp(-b_3 \cdot D) + (c_3 \cdot D)^{-p_3} \quad (5)$$

As distance from the parent plant increases, the first component goes to zero and the second component then estimates the tail.

One interesting question is whether the ‘typical’ tail shape of the dispersal curve is qualitatively impacted by the mode of dispersal, in other words, do certain dispersal mechanisms produce a significantly different tail to the dispersal curve? Based on a study employing 68 different datasets, [34] indicates that there is no clear link between tail shape and dispersal mode, suggesting that there is relatively little selection pressure for tail behaviour.

A further complicating factor is that the seeds of most plants are dispersed by multiple mechanisms. Hence, their seed shadows are comprised of a mixture of dispersal models [17], hence, the calibration of a seed distribution pattern to a single model is likely to be errorful. An additional feature is that dispersal agents may engage in secondary dispersal, i.e., from initial dispersal sites, thereby increasing the seed shadow.

2.3. Plant-inspired Algorithms

Until recently, little attention was paid to the potential utility of plant metaphors for the design of computational algorithms. The last few years have seen increased interest in this area, with the development of a number of plant-inspired algorithms. Broadly speaking, these fall into three categories, namely algorithms inspired by:

- i. plant propagation behaviour,
- ii. light-foraging behaviour (branching algorithms), and
- iii. purported swarm behaviour of plant root networks.

In this paper we restrict attention to the first of these.

3. Plant Propagation Algorithms

Plants have a repertoire of processes by which they propagate themselves including seed dispersal and root propagation. Effective propagation plays an important role in ensuring the survival of plant species, and in turn this depends on the ability of the plant to propagate itself into resource-rich areas. Hence, this process can metaphorically provide inspiration for the design of robust optimisation algorithms and also for the design of engineering systems [20].

Three algorithms which have been inspired by these processes, the *Invasive Weed Algorithm* [15], the *Paddy Field Algorithm* [21] and the *Strawberry Plant Algorithm* [24] are discussed below.

3.1. Invasive Weed Optimisation Algorithm

The *invasive weed optimisation algorithm* (IWO) (pseudocode provided in Algorithm 2), based on the colonisation behaviour of weeds, was proposed by Mehrabian and Lucas in 2006 [15]. The inspiration for the algorithm arose from the observation that weeds, or more generally, any plant, can effectively colonise a territory unless their growth is carefully controlled. Two aspects of this colonising behaviour are that weeds thrive in fertile soil and reproduce more effectively than their peers in less-fertile soil, and the dispersal of seeds during plant reproduction is stochastic.

Algorithm 1: Invasive Weed Algorithm [15]

```
Generate  $p_{initial}$  seeds and disperse them randomly in the search space;  
Determine the best solution in the current colony and store this location;  
repeat  
    Each plant in the population produces a quantity of seeds depending on the quality  
    of its location;  
    Disperse these new seeds spatially in the search space giving rise to new plants;  
    If maximum number of plants ( $p_{max} > p_{initial}$ ) has been exceeded, reduce the  
    population size to  $p_{max}$  by eliminating the weakest (least fit) plants. This simulates  
    competition for resources;  
    Assess the fitness of new plant locations and, if necessary, update the best location  
    found so far;  
until until terminating condition;  
Output the best location found;
```

Algorithm 2: Invasive Weed Algorithm

Generate p_{initial} seeds and disperse them randomly in the search space;
 Determine the best solution in the current colony and store this location;
repeat
 Each plant in the population produces a quantity of seeds depending on the quality of its location;
 Disperse these new seeds spatially in the search space giving rise to new plants;
 If maximum number of plants $p_{\text{max}} > p_{\text{initial}}$ has been exceeded, reduce the population size to p_{max} by eliminating the weakest (least fit) plants. This simulates competition for resources;
 Assess the fitness of new plant locations and, if necessary, update the best location found so far;
until *terminating condition*;
 Output the best location found;

The three key components of the algorithm are seeding (reproduction), seed dispersal and competition between plants. Mehrabian and Lucas operationalised these mechanisms in the following way in the IWO algorithm.

3.1.1. Seed Production

Each plant produces multiple seeds, based on its fitness relative to that of the other plants in the current colony of weeds. A linear scaling system is used whereby all plants are guaranteed to produce a minimum number of seeds (min_{seeds}), and no plant can produce more than a maximum number of seeds (max_{seeds}). The number of seeds produced by an individual plant is calculated using the following:

$$s(x) = \frac{f(x) - f_{\min}}{f_{\max} - f_{\min}} * (s_{\max} - s_{\min}) + s_{\min} \quad (6)$$

where f_{\max} and f_{\min} are the maximum and minimum fitnesses in the current population and $f(x)$ is the fitness of the plant x .

3.1.2. Seed Dispersal

While the IWO algorithm employs the notions of fitness and reproduction, unlike the GA, the IWO does not use genetic operators in the creation of populational diversity. Exploration of the search space is obtained via a simulated

seed dispersal mechanism. The seeds associated with each plant are dispersed by generating a random displacement vector and applying this to the location of their parent plant. The displacement vector has n elements corresponding to the n dimensions of the search space, and is obtained by generating n normally distributed random numbers, with a mean of zero and a standard deviation calculated using the following:

$$\sigma_{iter} = \left(\frac{iter_{max} - iter}{iter_{max}} \right)^n (\sigma_{max} - \sigma_{min}) + \sigma_{min} \quad (7)$$

where $iter$ is the current algorithm iteration number, $iter_{max}$ is the maximum number of iterations, σ_{max} and σ_{min} are maximum and minimum allowable values for the standard deviation, n is a non-linear modulation index, and σ_{iter} is the standard deviation used in the current iteration in calculating the seed displacements.

The effect of this formulation is to encourage random seed dispersal around the location of the parent plant, with decreasing variance over time. This results in greater seed dispersal in earlier iterations of the algorithm, promoting exploration of the search space. Later, the balance is tilted towards exploitation as the value of σ_{iter} is reduced. The incorporation of the non-linear modulation index in (7) also tilts the balance from exploration to exploitation as the algorithm runs.

Depending on the scaling of the search space, the same value of σ_{iter} could be applied when randomly drawing each element of the displacement vector. Alternatively, differing values of $\sigma_{initial}$ and σ_{final} could be set for each dimension if required.

3.1.3. Competition for Resources

Competition between plants is simulated by placing a population size limit on the colony (p_{max}). The plant colony starts with a population of size $p_{initial}$. The population increases as new plants grow in subsequent generations. Once the p_{max} population limit is reached, parent plants compete with their children for survival. The parent and child plants are ranked by fitness, with only p_{max} plants surviving into the next generation. This mechanism ensures that the best solution found to date cannot be lost between iterations (elitism).

3.1.4. Performance of the Algorithm

The IWO is a conceptually simple, numerical, non-gradient based, optimisation algorithm. As yet due to its novelty, there has been limited investigation of its effectiveness, scalability and efficiency. Mehrabian and Lucas [15] report GA and PSO competitive results from the IWO algorithm with settings of 10-20 weeds, maximum and minimum numbers of seeds per plant of 2 and 0 respectively, and a non-linear modulation index value of 3. Competitive results for the IWO algorithm are also reported by [2, 16] and [38].

The algorithm requires that several problem-specific parameters are set by the modeller including, the maximum and minimum number of seeds that a plant can produce, the values for σ_{max} , σ_{min} and $iter_{max}$, and the initial and the maximum population size. However, the determination of good values for these parameters is not necessarily a trivial task, particularly in poorly understood problem environments.

Recent work has extended the application of IWO into clustering where each individual seed consists of a string of up to n real-valued vectors of dimension d , corresponding to the n cluster centre coordinates (in d dimensional space) [14]. Apart from the IWO algorithm, a number of other algorithms which draw inspiration from seed-dispersal behaviour have been proposed, including the *Paddy Field Algorithm* [21].

3.2. Paddy Field Algorithm

The *paddy field algorithm* was first proposed by Premaratne, Samarabandu and Sidu (2009) [21]. This algorithm draws inspiration from aspects of the plant reproduction cycle, concentrating on the processes of pollination and seed dispersal.

Let the vector $x = (x_1, x_2, \dots, x_n)$ correspond to a location in an n dimensional space and $y = f(x)$ is the ‘fitness’ or ‘quality’ of that location. Each seed i therefore, has a corresponding location x_i and a corresponding fitness. The paddy field algorithm manipulates a population of these ‘seeds’ in an attempt to find a good solution to the optimisation problem of interest. The algorithm consists of five stages, sowing, selection, seeding, pollination, and dispersion [21]. Each of these are described below.

3.2.1. Sowing

An initial population of (p) seeds are (sown) at random locations in the search space.

3.2.2. Selection

The seeds are assumed to grow into plants, and each of these plants has an associated fitness value (y) determined by the output of the underlying objective function when evaluated at the plant's location. The plants are ranked by fitness, and the best n plants are then selected to produce seeds.

3.2.3. Seeding

Each plant produces a number of seeds in proportion to its fitness. The fittest plant produces s_{max} seeds and the other plants produces varying amounts of seeds, calculated using:

$$s = s_{max} \frac{y - y_t}{y_{max} - y_t}$$

The term y_{max} is the fitness of the best plant in the current population, and y_t is the fitness of the lowest ranked plant selected in the previous step. Although the algorithm describes this step as 'seeding', it can more correctly be considered as the process of growth of flower structures in order to enable pollination.

3.2.4. Pollination

Only a portion of the seeds become viable and to determine this portion, a simulated pollination process is applied whereby the probability that a seed is pollinated depends on the local density of plants around the seed's parent plant. The higher the density, the greater the chance of pollination. A hypersphere of radius a is defined, and two plants are considered to be neighbours if the distance between them is less than a . The pollination factor U_j of plant j (with $0 \leq U_j \leq 1$) is then calculated using:

$$U_j = \exp(v_j/v_{max} - 1)$$

where v_j is the number of neighbours of the plant j and v_{max} is the number of neighbours of the plant with the largest number of neighbours in the population.

3.2.5. Dispersion

The pollinated seeds are then dispersed from the location of their parent plant such that the location of the new plant (grown from the dispersed seed) is determined using $N(x_j, \sigma)$ where x_j is the location of the parent plant and σ is a user-selected parameter.

The above five steps are iterated until a termination condition is reached. In summary, the fittest plants give rise to the greatest number of seeds, and search is intensified around the better regions of the landscape uncovered thus far. Variants on the PFA include [12].

Algorithm 3: Paddy Field Algorithm [21]

```

Generate an initial population of  $p$  plants each located randomly in the search space;
Choose value for  $maxiter$  and  $n$  (see below);
Set generation counter  $iter = 1$ ;
repeat
    Calculate fitness of each plant ( $y_i$ ) and store in vector  $N$ 
    ( $N_i = fitness(y_i : i = 1, \dots, p)$ );
    Sort  $N : (N_i : i = 1, \dots, p)$  into descending order (assuming the objective is to
    maximise fitness);
    for  $i = 1 : n$  (top  $n$  plants) do
        Generate seeds for each selected plant;
        Implement pollination step;
        Disperse pollinated seeds;
    end
    Replace old population with new plants;
     $iter = iter + 1$ ;
until  $iter = maxiter$ ;
Output the best location found;

```

3.3. Strawberry Plant Algorithm

Although many plants propagate using seeds, some employ a system of ‘runners’, or horizontal stems which grow outwards from the base of the plant. At variable distances from the parent plant, if suitable soil conditions are found, new roots will grow from the runner and in turn produce an offspring clone of the parent plant. An example of this behaviour is provided by modern strawberry plants which can propagate via seeds and by runners. This has inspired the development of an optimisation algorithm based on this phenomenon [24].

The algorithm is based on the following ideas:

- healthy plants in good resource locations generate more runners,
- plants in good resource locations tend to send short runners in order to exploit local resources,
- plants in poorer resource locations tend to send longer runners to search for better conditions, and
- as the generation of longer runners requires more resource investment, plants generating these will create relatively few of them.

The algorithm therefore seeks to balance exploration with exploitation, with increasing local exploration over time as plants concentrate in the locations with best conditions for growth. Salhi and Fraga [24] report competitive results from this algorithm when applied to a number of real-valued benchmark optimisation problems. Algorithm 4 presents an adapted version of the algorithm based on [24].

4. Applications

Despite the relative recency of the introduction of plant propagation-inspired algorithms, there have been a number of applications to a range of diverse real-world problems, showing promise compared to existing approaches. We mention a selection of applications here: these applications range from recommender systems [38] to engineering problems [39, 40]. [39] apply the Invasive Weed Optimisation algorithm to the problem of optimising radio antenna structures. They find that the Invasive Weed Optimisation is competitive with the Particle Swarm Optimisation (PSO) algorithm, in accuracy, speed of convergence and simplicity. [41] apply a modified (discrete) invasive weed optimization algorithm to optimize DNA encoding sequences. Experimental results show that the proposed method is effective and convenient for the design and selection of effective DNA sequences *in silico* for controllable DNA computing.

[42] use a discrete invasive weed optimization (DIWO) algorithm for cooperative multiple task assignment of unmanned aerial vehicles (UAVs) and compare the solutions with those of genetic algorithms (GAs). Their results show that DIWO has better performance than GAs in both optimality of the solutions and computation time.

Algorithm 4: Strawberry Propagation Algorithm (adapted from [24])

```

Generate an initial population of  $m$  plants  $p_i : i = 1, \dots, m$  each located randomly in
the search space;
Choose values for  $maxgen$  and  $y$  (see below);
Set generation counter  $gen = 1$ ;
repeat
    Calculate fitness of each plant and store in vector  $N$ 
    ( $N_i = fitness(p_i : i = 1, \dots, m)$ );
    Sort  $N : (N_i : i = 1, \dots, m)$  into descending order (assuming the objective is to
    maximise fitness);
    for  $i = 1 : (m/10)$  (top 10% of plants) do
        Generate  $(y/i)$  short runners for each plant ( $y$  is a user-defined parameter
        which defines the intensity of local search around each of the fitter plants);
        if any of the new locations has higher fitness than that of the parent plant then
            move the parent plant to the new location with the highest fitness
            ( $r_i \rightarrow p_i$ );
        else
            Discard the new locations and the parent plant stays at its current location;
        end
    end
    for  $i = (m/10) + 1 : m$  (indices for remaining plants) do
        Generate one long runner for each plant not in the top 10% and select the
        location of the end-point  $r_i$  for that runner randomly in the search space;
        if the new location has higher fitness than that of the parent plant then
            move the parent plant to the new location ( $r_i \rightarrow p_i$ );
        else
            Discard the new location and the parent plant stays at its current location;
        end
    end
until  $gen = maxgen$ ;
Output the best location found;

```

[40] examine the performance of their extended Strawberry Propagation Algorithm on a range of constrained engineering optimisation problems on continuous domains, including design of welded beam, pressure vessel, spring and speed reducer. Their results are that the Strawberry Propagation Algorithm found either near best known solutions or optimal ones to all problems. They compare the Strawberry Propagation Algorithm results to results obtained with other approaches such as GAs, Fogel's Evolutionary Programming, PSO, variations of the Harmony Search Algorithm and Integer Programming, and find that the Strawberry Propagation Algorithm is superior in the majority of cases.

5. Conclusion

At a conceptual level, plant dispersal can be considered as a search process, wherein the seed or plant is searching for good locations and therefore, inspiration from dispersal activities of plants can plausibly serve as the design inspiration for optimisation algorithms. In this chapter we focussed on different processes of plant dispersal, and described a number of existing optimisation algorithms which draw inspiration from these. These were the *invasive weed optimisation* algorithm, the *paddy field* algorithm, and the *strawberry plant* algorithm.

In this work, we have noted and justified an array of plant behaviours which are exhibited in the natural world. With some exceptions, little inspiration has been taken from these mechanisms, as yet, for the design of computational algorithms. Most of the algorithms developed thus far are relatively recent in design and further work is required in order to assess their utility and to assess more fully whether they represent truly novel problem-solving mechanisms or whether they are qualitatively similar to existing natural computing algorithms. Work to date appears to indicate that they are at least competitive on the problems to which they have been applied. However, there is clearly rich potential for future work.

We wish to stimulate interest in this exciting, and under-explored area of natural computing. Of great importance here are the investigation of additional strategies for overcoming local optimality in complex solution spaces, and performing a robust search of a solution space. Additional research should study the degree to which neighbourhoods are exploited under different parameter settings governing the operation of each algorithm.

References

- [1] Augspurger, C. (1986). Morphology and Dispersal Potential of Wind-dispersed Diaspores of Neotropical Trees, *American Journal of Botany*, 73(3):353-363.
- [2] Basak A., Pal S., Das S., Abraham A. and Snasel V. (2010). A Modified Invasive Weed Optimization Algorithm for Time-Modulated Linear Antenna Array Synthesis, in *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2010)*, pp. 372-379, IEEE Press.

-
- [3] Bullock, J. and Clarke, R. (2000). Long distance seed dispersal by wind: measuring and modelling the tail of the curve, *Oecologia*, 124:506-521.
- [4] Cain, M., Milligan, B. and Strand, A. (2000). Long-Distance Seed Dispersal in Plant Populations, *American Journal of Botany*, 87(9):1217-1227.
- [5] Contreras Sanchez, J., Green, D. and Quesada, M. (2011). A Field Test of Inverse Modeling of Seed Dispersal, *American Journal of Botany*, 98(4):698-703.
- [6] Dressaire, E., Santoso, J., Yamada, L. and Roper, M. (2013). Control of fluidic environments by mushrooms, Paper presented at *66th Annual Meeting of the American Physical Society Division of Fluid Dynamics*, November 24-26, 2013; Pittsburgh, Pennsylvania.
- [7] Gillespie, R., Baldwin, B., Waters, J., Fraser, C., Nikula, R. and Roderick, G. (2011). Long-distance dispersal: a framework for hypothesis testing, *Trends in Ecology and Evolution*, 27(1):47-56.
- [8] Green, D. and Johnson, E. (1989). A Model of Wind Dispersal of Winged or Plumed Seeds, *Ecology*, 70(2):339-347.
- [9] Howe, H. and Smallwood, J. (1982). Ecology of Seed Dispersal, *Annual Review of Ecology and Systematics*, 13:201-228.
- [10] Jerzolimski, A., Beatriz Ribeiro, M. and Martins, M. (2009). Are Tortoises Important Seed Dispersers in Amazonian Forests? *Oecologia*, 161(3):517-528.
- [11] Koller, D. (2011). *The Restless Plant*, Harvard University Press, Cambridge, MA.
- [12] Kong, X., Chen, Y-L., Xie, W. and Wu, X. (2012). A Novel Paddy Field Algorithm Based on Pattern Search Method, in *Proceedings of the IEEE International Conference on Information and Automation*, pp. 686-690, IEEE Press.
- [13] Levey, D., Tewksbury, J. and Bolker, B. (2008). Modelling long-distance seed dispersal in heterogeneous landscapes, *Journal of Ecology*, 96:599-608.

- [14] Liu, R., Wang, X. and Li, Y. (2012). Multi-objective Invasive Weed Optimization Algorithm for Clustering, in *Proceedings of 2012 IEEE World Congress on Computational Intelligence (WCCI 2012)*, pp. 1556-1563, IEEE Press.
- [15] Mehrabian, A. and Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization, *Ecological Informatics*, 1:355-366.
- [16] Mehrabian, A. and Yousefi-Koma, A. (2007). Optimal positioning of piezoelectric actuators on a smart fin using bio-inspired algorithms, *Aerospace Science and Technology*, 11:174-182.
- [17] Nathan, R. and Muller-Landau, H. (2000). Spatial patterns of seed dispersal, their determinants and consequences for recruitment, *Tree*, 15(7):278-285.
- [18] Nathan, R., Katul, G., Bohrer, G., Kuparinen, A., Soons, M., Thompson, S., Trakhtenbrot, A. and Horn, H. (2011). Mechanistic models of seed dispersal by wind, *Theoretical Ecology*, 4:113-132.
- [19] Niklas, K. and Spatz, H. (2012). *Plant Physics*, University of Chicago Press.
- [20] Pandolfi, C. and Izzo, D. (2013). Biomimetics on seed dispersal: survey and insights for space exploration, *Bioinspiration and Biomimetics*, 8(2):025003, doi:10.1088/1748-3182/8/2/025003.
- [21] Premaratne, U., Samarabandu, J. and Sidhu, T. (2009). A New Biologically Inspired Optimization Algorithm, in *Proceedings of Fourth International Conference on Industrial and Information Systems (ICIIS 2009)*, pp. 279-284, IEEE Press.
- [22] Rad, H. and Lucas, C. (2007). A Recommender System based on Invasive Weed Optimization Algorithm, in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 4297-4304, IEEE Press.
- [23] Roper, M., Seminara, A., Bandi, M., Cobb, A., Dillard, H. and Pringle, A. (2010). Dispersal of fungal spores on a cooperatively generated wind, *Proc Natl Acad Sci*, 107(41):17474-9.

-
- [24] Salhi, A. and Fraga, E. (2011). Nature-Inspired Optimisation Approaches and the New Plant Propagation Algorithm, in *Proceedings of 2011 International Conference on Numerical Analysis and Optimization (ICeMATH 2011)*, pp. K2-1:K2-8.
- [25] Schupp, E. (1993). Quantity, Quality and the Effectiveness of Seed Dispersal by Animals, *Vegetatio*, 107/108:15–29.
- [26] Venable, D. and Brown, J. (1993). The Population-dynamic Functions of Seed Dispersal, *Vegetatio*, 107/108:31-55.
- [27] Viswanathan, G., Afanasyev, V., Buldyrev, S., Havlin, S., da Luz, M., Raposo, E. and Stanley, E. (2001). Lévy flights search patterns of biological organisms, *Physica A*, 295:85-88.
- [28] Viswanathan, G., Raposo, E. and da Luz, M. (2008). Lévy flights and superdiffusion in the context of biological encounters and random searches, *Physics of Life Reviews*, 5(3): 133–150
- [29] Viswanathan, G., da Luz, M., Raposo, E. and Stanley, E. (2011). *The Physics of Foraging: An Introduction to Random Searches and Biological Encounters*, Cambridge University Press.
- [30] Wallace, H. and Trueman, S. (1995). Dispersal of *Eucalyptus torelliana* seeds by the resin-collecting stingless bee, *Trigona carbonaria*, *Oecologia*, 104:12-16.
- [31] Warren, R., Giladi, I. and Bradford, M. (2014). Competition as a mechanism structuring mutualisms, *Journal of Ecology*, 102:486-495.
- [32] Wenny, D. (2001). Advantages of seed dispersal: A re-evaluation of directed dispersal, *Evolutionary Ecology Research*, 3:51–74.
- [33] Willson, M. (1993). Dispersal Mode, Seed Shadows, and Colonization Patterns, *Vegetatio*, 107/108: 260–280.
- [34] Willson, M. and Traveset, A. (2000). The Ecology of Seed Dispersal, in *The Ecology of Regeneration in Plant Communities* 2nd edition (ed. M. Fenner), pp. 85–110, CAB International, Oxford, UK.

- [35] Yi, X., Liu, G., Steele, M., Shen, Z. and Liu, C. (2013). Directed seed dispersal by a scatter-hoarding rodent: the effects of soil water content, *Animal Behaviour*, 86:851-857.
- [36] Dressaire, E., Santoso, J., Yamada, L. and Roper, M. (2013) 'Control of fluidic environments by mushrooms', Paper presented at *66th Annual Meeting of the American Physical Society Division of Fluid Dynamics*, November 24–26, 2013, Pittsburgh, Pennsylvania.
- [37] Roper, M., Seminara, A., Bandi, M., Cobb, A., Dillard, H. and Pringle, A. (2010). Dispersal of fungal spores on a cooperatively generated wind, *Proc Natl Acad Sci*, 107(41):17474–17479.
- [38] Rad, H. and Lucas, C. (2007) 'A Recommender System based on Invasive Weed Optimization Algorithm', in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2007)*, IEEE Press, pp. 4297–4304.
- [39] Mallahzadeh, A. R., Oraizi, H. and Davoodi-Rad, Z. (2008) 'Application of the invasive weed optimization technique for antenna configurations', *Progress In Electromagnetics Research*, 79:137–150.
- [40] Suleiman, M., Salhi, A., Selamoglu, B. I. and Kirikchi, O. B. (2014) 'A Plant Propagation Algorithm for Constrained Engineering Optimisation Problems', *Mathematical Problems in Engineering*, vol 2014. doi:10.1155/2014/627416
- [41] Zhang, X., Wang, Y., Cui, G., Niu, Y. and Xua, J. (2009) 'Application of a novel IWO to the design of encoding sequences for DNA computing', *Computers & Mathematics with Applications*, 57(11–12):2001–2008.
- [42] Ghalenoiei, M.R.; Hajimirsadeghi, H.; Lucas, C., (2009) 'Discrete invasive weed optimization algorithm: application to cooperative multiple task assignment of UAVs', Paper presented at *Decision and Control, 2009* held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. *Proceedings of the 48th IEEE Conference*, 1665–1670, 15–18.

Chapter 4

TOPOGRAPHICAL CLEARING DIFFERENTIAL EVOLUTION APPLIED TO REAL-WORLD MULTIMODAL OPTIMIZATION PROBLEMS

Wagner F. Sacco^{1,*}, *Ana Carolina Rios-Coelho*^{1,†}
and Nélio Henderson^{2,‡}

¹Universidade Federal do Oeste do Pará, Santarém, PA, Brazil

²Instituto Politécnico, Universidade do Estado do Rio de Janeiro,
Nova Friburgo, RJ, Brazil

Abstract

Many real-world optimization problems are multimodal, requiring techniques that overcome local optima, which can be done using niching methods. In order to do so, we describe a niching method based on the clearing paradigm, Topographical Clearing, which employs a topographical heuristic introduced in the early nineties, as part of a global optimization method. This niching method is applied to differential evolution, but it can be used in other evolutionary or swarm-based methods, such as the genetic algorithm and particle swarm optimization. The algorithm, called TopoClearing-DE, is favorably compared against the canonical version of differential evolution in real-world optimization problems. As the problems attacked are quite challenging, the results show that Topographical

*E-mail address: wagner.sacco@ufopa.edu.br

†E-mail address: ana.coelho@ufopa.edu.br

‡E-mail address: nelio@iprj.uerj.br

Clearing can be applied to populational optimization methods in order to solve problems with multiple solutions.

Keywords: Niching Methods, Clearing, Topographical Heuristic, Differential Evolution, Multimodal Problems

1. Introduction

Some optimization problems in engineering are highly multimodal, remaining a great challenge for most methods, as they have large search spaces with multiple local and even global optima. In this chapter, we address four challenging real-world problems taken from the literature, three continuous and an NP-hard combinatorial optimization problem [19].

In these multimodal problems, the search space should be thoroughly explored so that the optimization algorithm does not converge to a local optimum. To overcome this difficulty, many solutions have been proposed. Let's mention, for example, some techniques that were applied to nuclear-engineering problems: a parallel genetic algorithm [31], a niching method [24] applied to genetic algorithms [45], a hybrid algorithm that alternates exploration and exploitation of the search space [41], and a new mutation scheme [43] applied to differential evolution (DE) [49].

Niching methods are techniques designed to maintain populational diversity in evolutionary or swarm-based methods, so that multiple optima are determined in multimodal problems. These optima may consist in more than one global optimum and some local minima, or in a single global optimum and many global minima. Most niching methods are based on one of the following schemes:

1. Fitness sharing [15], which modifies the search landscape by reducing the payoff in densely populated regions [46].
2. Crowding [11], where a new individual replaces its most similar element in the population.
3. Clearing [32], where the best members of the population, the so-called dominants, receive the entire payoff.

The three main niching methods have been applied to the differential evolution algorithm, which we use in this work. See, for example, [51, 61, 34]. For a

brief survey, see [10]. For a more detailed exposition, the reader should refer to Rönkkönen's thesis [38].

Sareni and Krähenbühl [46] tested these three niching schemes applied to the genetic algorithm, concluding that clearing is the best, provided that the niching radius σ that delimits each dominant's territory is correctly estimated. This is the drawback of this method, especially in real-world problems, where the search space is generally unknown beforehand.

In order to overcome this limitation, Sacco et al. [45] proposed a variant of clearing where the individuals are clustered using Fuzzy Clustering Means (FCM) [7] and each cluster has a dominant individual. However, FCM requires the number of clusters as input and is rather complicated.

With the same motivation, Qu et al. [34] proposed an ensemble of clearing differential evolution algorithms, where the initial population is divided into three equal subpopulations P_1 , P_2 , and P_3 , which receive radii $\sigma_{P_1} = 0.005 \times SR$, $\sigma_{P_2} = 0.01 \times SR$, and $\sigma_{P_3} = 0.05 \times SR$, where SR is the problem's search range. These subpopulations exchange information during the selection phase. This scheme increases clearing's efficiency, but is still dependent of σ .

In this chapter, we employ a method which was recently introduced by Sacco et al. [44]. It is based on the clearing paradigm which is simpler than the schemes introduced in [45] and [34]. It uses a clustering heuristic based on the topographical information on the objective function, which was part of an optimization algorithm proposed by Törn and Viitanen [54], the Topographical Algorithm (TA). In this method, we employ the topographical heuristic with the purpose of determining the dominant individual in a neighborhood. Originally, Törn and Viitanen [54] used this mechanism to determine minima from a set of sampled points, so that they were initial solutions for a local optimization algorithm. We apply this clearing variant, called topographical clearing, to differential evolution, which outperformed the more popular genetic algorithm and particle swarm optimization in extensive experiments [60]. However, this method can be applied to any evolutionary or swarm-based technique.

The remainder of this chapter is described as follows. The description of DE is presented in Section 2. The novel niching method is described in Section 3, as well as its application to DE. The computational experiments and their discussions are in Section 4. Finally, the conclusions are made in Section 5.

2. The Differential Evolution Algorithm

In this section, we describe the canonical version of differential evolution, as introduced by Storn and Price [49]. DE is applied to the minimization of an objective function $f(\mathbf{x})$, where \mathbf{x} is a continuous variable vector with domain $[\mathbf{low}, \mathbf{up}] \subset \mathbb{R}^n$.

Let's describe DE in a pseudo-code style, so that the novice can easily grasp its concept. The algorithm is outlined in Fig. 1 and its operators are described in Figs. 2, 3, 4, and 5.

Input parameters, which remain constant along the optimization process, are population size NP and, to be explained below, crossover rate CR and scaling factor F . First of all, an initial random population is generated by function “initialize”, as described in Fig. 2. Note that each initial solution or individual must meet the boundary constraints. After that, inside a loop, the evolutive process starts until a stopping criterion is satisfied.

The first operation inside the loop is mutation, described by function “mutate”, Fig. 3. In mutation, a trial solution is generated for each individual i as follows:

$$\hat{\mathbf{x}}_i = \mathbf{x}_{p(1)} + F(\mathbf{x}_{p(2)} - \mathbf{x}_{p(3)}), \quad (1)$$

where $p(1)$, $p(2)$, and $p(3)$ are random indexes mutually different from each other and different from index i , and F is a scaling factor in the range $[0, 2]$. The solution correspondent to the first random index, $\mathbf{x}_{p(1)}$, is known as the base vector. This vector is altered by the addition of the weighted difference of the two other solutions with indexes $p(2)$ and $p(3)$. The operation is repeated as long as trial solution $\hat{\mathbf{x}}_i$ is outside the domain.

After mutation, population goes through crossover, as in Fig. 4. In this operation, component j of offspring \mathbf{y}_i is found from its parents \mathbf{x}_i and $\hat{\mathbf{x}}_i$ according to the rule

$$y_i^j = \begin{cases} \hat{x}_i^j, & \text{if } R^j \leq CR \text{ or } j = I_i \\ x_i^j, & \text{otherwise} \end{cases}, \quad (2)$$

where I_i is a random integer in range $[0, n]$, R^j is a random in $[0, 1]$, and crossover rate CR , also in $[0, 1]$, controls the fraction of parameter values that are copied from the trial solution $\hat{\mathbf{x}}_i$. Note that alternative $j = I_i$ assures that at least one component will receive a mutated value.

Finally, there is the selection process, Fig. 5, which defines the population of next generation as follows:

$$\mathbf{x}_i^{NIter+1} = \begin{cases} \mathbf{y}_i^{NIter}, & \text{if } f(\mathbf{y}_i^{NIter}) \leq f(\mathbf{x}_i^{NIter}) \\ \mathbf{x}_i^{NIter}, & \text{otherwise} \end{cases} . \quad (3)$$

The trial solution will only replace its counterpart in the current population if it's equal or better than the latter. As pointed out in [21], in DE's selection scheme, a trial vector is not compared against all the individuals in the current population, but only against its counterpart.

Note that it's in function "select" (Fig. 5), that the best solution found so far and its fitness value are stored.

As termination criterion, one may use the number of generations ($NIter$ in our pseudocode), the number of objective-function evaluations, or, as in [20], $|f_{max} - f_{min}| < \varepsilon$, where f_{max} and f_{min} are the maximum and minimum function values within a generation.

3. Topographical Clearing

3.1. Clearing

As mentioned in the Introduction, in clearing the best members of the population, the so-called dominants, receive the entire payoff. This procedure is applied after evaluating the fitness of the individuals and before applying the selection operator [32]. The clearing radius σ defines a range inside all but the κ individuals having the best fitnesses are cleared [38], i.e., have their objective function values zeroed for a maximization problem or receive a large value for a minimization problem. The population members distant more than from a dominant individual are not affected. Figure 6, based on [32], shows the function that performs the original clearing method. Function **SortFitness(P)** sorts the population **P** in decreasing order of fitness, so that the first elements of the list are the dominants, if it is a maximization problem, or in increasing order, for minimization problems.

3.2. The Topographical Algorithm

Between the early seventies and mid-nineties, a global optimization paradigm based on clustering was studied by some researchers, mainly in Europe. The

seminal article by Becker and Lago [6] was followed by, among others, Törn [58], Timmer [52], Törn and Viitanen [54, 55], and Ali and Storey [2]. Ali [1], and Levi and Haas [23] present fine reviews on the clustering methods. According to Törn and Žilinskas [57], the motivation for exploring clustering methods is based on the following:

1. It is possible to obtain a sample of points in the search space consisting of concentration of points in the neighborhood of local minimizers of the objective function f .
2. The points in the sample can be clustered giving clusters identifying the neighborhoods of local minimizers and thus permitting local optimization methods to be applied.

The original TA is non-iterative and based on the exploration of the search space [2]. It consists of three steps [55]:

1. A uniform random sampling of N points in the search space.
2. The construction of the topograph, which is a graph with directed arcs connecting the accepted sampled points on a k -nearest neighbors basis, where the direction of the arc is towards a point with a larger function value. The minima of the graph are the points better than their neighbors, i.e., the nodes with no incoming arcs.
3. The topograph minima are starting points for a local optimization algorithm. The best point obtained from all the executions using each minimum as the initial solution is the result of the algorithm.

Originally, Törn and Viitanen [54, 55] obtained the initial solutions from step 1 sampling points in a unit hypercube, until N points with their nearest neighbors farther than a threshold distance δ were obtained. Then, these points were denormalized. But these authors add that any other method that produces a very uniform covering can be used. In fact, they used the more efficient quasi-random sampling in an iterative version of TA [56]. In their tests, Törn and Viitanen [55] used mostly $N = 100$ or $N = 200$.

Step 2, the construction of the topograph, is the heart of the method. First of all, a $N \times N$ symmetric distance matrix is computed. Following that, a

$N \times k$ matrix called kNN -matrix is constructed containing, for each point, the indexes of its k -nearest neighbors sorted by distance. Next, this matrix, which is an undirected topograph, is transformed into a directed topograph indicating if the reference is to a point with larger or smaller objective function value by giving the reference a plus or minus sign, respectively [1]. The signs represent the directed arcs in the graph, a positive sign representing the “arrow head” of the arc, and the negative sign the “start” of the arc [55]. Finally, the points that correspond to rows with only positive signs are the topograph minima.

Let us illustrate how the topographical heuristic works by a simple illustrative example, adapted from [1]. Suppose we want to minimize the function

$$f(x, y) = x^2 + y^2, \tag{4}$$

and that six points were sampled and their function values calculated: $f(P_1) = f(2, 5) = 29$, $f(P_2) = f(1, 2) = 5$, $f(P_3) = f(3, 4) = 25$, $f(P_4) = f(0, 1) = 1$, $f(P_5) = f(5, 0) = 25$, and $f(P_6) = f(4, 2) = 20$.

First, the symmetric squared distance matrix \mathbf{D} is constructed, where, for example, the element $d_{1,3}$ corresponds to the distance between P_1 and P_3 :

$$\mathbf{D} = \begin{bmatrix} 0 & 10 & 2 & 20 & 34 & 13 \\ 10 & 0 & 8 & 2 & 20 & 9 \\ 2 & 8 & 0 & 18 & 20 & 5 \\ 20 & 2 & 18 & 0 & 26 & 17 \\ 34 & 20 & 20 & 26 & 0 & 5 \\ 13 & 9 & 5 & 17 & 5 & 0 \end{bmatrix}. \tag{5}$$

Following that, the kNN -matrix is formed by each points k -nearest neighbors. Using $k = 3$, the nearest neighbors of P_1 (the first row of \mathbf{D}) are the points with indexes 3, 2, and 6, respectively. These elements will constitute the first row of the matrix. The process goes on until the following matrix is obtained:

$$\mathbf{kNN} = \begin{bmatrix} 3 & 2 & 6 \\ 4 & 3 & 6 \\ 1 & 6 & 2 \\ 2 & 6 & 3 \\ 6 & 2 & 3 \\ 3 & 5 & 2 \end{bmatrix}. \tag{6}$$

This matrix represents an undirected graph. Computationally, it is obtained sorting each row of \mathbf{D} and taking the first k elements' indexes. The elements of

the main diagonal of \mathbf{D} receive a very large value (e.g., 10^8) before sorting, so that they are not included in the kNN -matrix.

Now, the elements of kNN will receive a plus or minus sign according to their functional values in relation to the value of the point represented by the row index. The second row, for example, corresponds to P_2 , whose function value is equal to 5, which is more than $f(P_4) = 1$ (P_4 is element knn_{21}), but less than $f(P_3) = 25$ and $f(P_1) = 29$ (elements knn_{22} and knn_{23} , respectively). Therefore, knn_{21} will receive a minus sign and the other two elements a plus sign. The signed matrix becomes

$$\mathbf{kNN} = \begin{bmatrix} -3 & -2 & -6 \\ -4 & +3 & +6 \\ +1 & -6 & -2 \\ +2 & +6 & +3 \\ -6 & -2 & +3 \\ -3 & +5 & -2 \end{bmatrix}. \quad (7)$$

As the only point that corresponds to a row with only positive signs is $P_4 = (0, 1)$, this will be the starting point for a local optimization algorithm. When implementing the topographical heuristic, the signs can be attributed in the process of construction of kNN .

In step 3, Törn and Viitanen [55] say that any local optimization method can be used. They employed a gradient-based algorithm, as their tests were performed on algebraic test functions.

3.3. The Novel Niching Method

As mentioned in section 1, the new method based on the clearing paradigm is much simpler than those available in the literature. The topographical heuristic is applied to the population and the topograph minima are determined and flagged (function topograph in Figure 7). These minima receive the value $flag_i = 1$, and the others remain with $flag_i = 0$ that was previously assigned for all the individuals. Then, clearing (function *topoclearing* in the same figure) is applied as follows: the non-flagged individuals are cleared, receiving a large function value (as we are working with minimization problems), while the topograph minima are not punished, maintaining their original values. This scheme is described by Figure 8. Note that our new method does not require

parameters σ and κ , neither the sorting of individuals by fitness value, which is a computationally expensive procedure.

Finally, the selection is performed, as in Figure 9. The decision on whether to replace or not the current solution by the new one is made based on the cleared fitness values.

4. Numerical Comparisons

4.1. The Practical Problems

4.1.1. Chemical Equilibrium Problem

This nonlinear system, introduced by Meintjes and Morgan [28], has been widely employed in the literature, see [17, 25, 59, 16, 39], among others.

It concerns the combustion of propane (C_3H_8) in air (O_2 and N_2) to form ten products. This chemical reaction generates a system of ten equations in ten unknowns, which can be reduced to a system of five equations in five unknowns [28]. We solve this system formulating it as an optimization problem. To see how this formulation is made, the interested reader should see Appendix A.

The system is given by

$$\left\{ \begin{array}{l} f_1 = x_1x_2 + x_1 - 3x_5 \\ f_2 = 2x_1x_2 + x_1 + x_2x_3^2 + R_8x_2 - Rx_5 + 2R_{10}x_2^2 + R_7x_2x_3 + R_9x_2x_4 \\ f_3 = 2x_2x_3^2 + 2R_5x_3^2 - 8x_5 + R_6x_3 + R_7x_2x_3 \\ f_4 = R_9x_2x_4 + 2x_4^2 - 4Rx_5 \\ f_5 = x_1(x_2 + 1) + R_{10}x_2^2 + x_2x_3^2 + R_8x_2 + R_5x_3^2 + x_4^2 - 1 + R_6x_3 \\ \quad + R_7x_2x_3 + R_9x_2x_4 \end{array} \right. \quad (8)$$

where

$$\left\{ \begin{array}{l} R = 10 \\ R_5 = 0.193 \\ R_6 = 0.002597/\sqrt{40} \\ R_7 = 0.003448/\sqrt{40} \\ R_8 = 0.00001799/40 \\ R_9 = 0.0002155/\sqrt{40} \\ R_{10} = 0.00003846/40 \end{array} \right.$$

Variables x_i are surrogates for atomic combinations, which means that only positive values make physical sense. Among the four real solutions reported by

Meintjes and Morgan [28], only one has all-positive components (3.114102×10^{-3} , 3.459792×10^1 , 6.504177×10^{-2} , 8.593780×10^{-1} , 3.695185×10^{-2}) [28]. Hence, if the search domain is taken from the positive side, as we did using the interval $[0, 100]^5$, this will be the only solution.

4.1.2. Catalytic Reactor Model

Generally, parameter estimation problems are solved using least-squares, assuming that the variables are not subject to measurement error. In this problem, however, it is assumed that there are measurement errors in all variables. In order to solve it, it is necessary to use the error-in-variables approach [53]. Using this model, the objective-function has the form [14]:

$$\min_{\theta, \tilde{x}_i} \sum_{i=1}^m \sum_{j=1}^n \frac{(\tilde{x}_{ij} - x_{ij})^2}{\sigma_j^2} \quad (9)$$

subject to

$$f(\theta, \tilde{x}_i) = \mathbf{0}, \quad i = 1, \dots, m. \quad (10)$$

In the equations above, $\mathbf{x}_i = (x_{i1}, \dots, x_{in})^T$ represents measurements of the variables from $i = 1, \dots, m$ experiments, $\tilde{\mathbf{x}}_i = (\tilde{x}_{i1}, \dots, \tilde{x}_{in})^T$ are the unknown actual values, and σ_j is the standard deviation associated with the measurement of variable j [14]. Therefore, the error-in-variables approach involves not only the parameters θ , but also the true values \tilde{x}_i , increasing the dimensionality of the optimization problem.

This parameter estimation problem was introduced in [37], to model gas-phase catalytic hydrogenation of phenol on a palladium catalyst in pseudo-differential reactor [53]. Variables x_1 , x_2 and x_3 represent the partial pressures of phenol and hydrogen, and the initial reaction rate [53]. The model is described by the following equation [53]:

$$x_3 = \frac{\theta_1 \theta_2^2 \theta_3 x_1 x_2^2}{(1 + \theta_1 x_1 + \theta_2 x_2)^3}, \quad (11)$$

where θ_1 , θ_2 , and θ_3 are the parameters to be estimated. Standard deviations of 0.0075, 0.0075 and 2.5 are specified for x_1 , x_2 and x_3 , respectively [53]. Table B.1 (Appendix B) presents the data and the fitted values. The optimal parameters of this 59-variable problem are equal to $\theta_1 = 7.39696 \text{ atm}^{-1}$, $\theta_2 = 0.63782 \text{ atm}^{-1}$, and $\theta_3 = 1769.71 \text{ mol/kg h}$, corresponding to an objective

value of 30.3072 [14]. For variables \tilde{x}_i , we use the same search region as in [14]: $[x_{1i} - 3\sigma_1, x_{1i} + 3\sigma_1]$ and $[x_{2i} - 3\sigma_2, x_{2i} + 3\sigma_2]$, for $i = 1, \dots, 28$. For the parameters to be estimated, we adopt a wider range than these authors, $\theta_1, \theta_2 \in [0, 10]$ and $\theta_3 \in [1000, 2000]$. We must add that this problem has many local optima [14].

4.1.3. Turbine Balancing Problem

This is a combinatorial optimization problem [30]. The turbine balancing problem is very relevant, being a real challenge for optimization methods, as it is NP-hard [33]. It was originally proposed by [29] as a combinatorial optimization problem, being also formulated as a quadratic assignment problem [22]. Since then, it has been attacked by other researchers, using both formulations and different kinds of turbines [48, 4, 33, 8].

In this work, we solve the case presented in [29]. The problem consists in balancing the runners of a Francis hydraulic turbine. Ref. [48] gives a precise description of the problem to be solved:

A hydraulic turbine runner consists essentially of a cylinder with blades attached to its circumference. The turbine rotates as water flows across the blades. During the manufacturing process the individual blades must be welded into place, equally spaced around the cylinder. The problem encountered during this phase is the static balancing of the completed runner. Because of the complexity of the manufacturing process, the final weights of the blades may differ substantially. The result is an unbalanced runner. Since the runner can rotate at very high revolutions during use, it is crucial that the unbalance be as small as possible, otherwise the bearings on which the runners rotate will wear out very quickly.

We must add that, according to [29], the variations in final weight mentioned above can be as great as $\pm 5\%$.

Let us formulate the problem, following [29]. The runner is modeled as n equally-spaced weights on a circle of zero mass and radius r equal to the common distance from the blade centers-of-mass to the runner axis. The blade positions are labeled counterclockwise, starting at position $1 = (r, 0)$ in an $x - y$ coordinate system, receiving indexes $k = 1, 2, \dots, n$. Let P_t be a configuration of blades where $P_t(j) = k$ assigns blade k to position j . First, we define the

following variables:

M^k = mass of blade k ;

M_j^k = mass of blade k when in position j ;

$\theta_j = (2\pi/n)(j - 1)$ = angle between position j and position 1, $j = 1, \dots, n$;

M = total mass of blades = $\sum_{k=1}^n M^k$.

Then, each permutation P_t determines a center of mass (\bar{x}, \bar{y}) given by:

$$\bar{x}(P_t) = \frac{1}{M} \sum_{j=1}^n M_j^k r \cos \theta_j, \quad (12)$$

$$\bar{y}(P_t) = \frac{1}{M} \sum_{j=1}^n M_j^k r \sin \theta_j, \quad (13)$$

Finally, Eq. (12) and Eq. (14) define deviation \bar{D} ,

$$\bar{D}(P_t) = \sqrt{[\bar{x}(P_t)]^2 + [\bar{y}(P_t)]^2}, \quad (14)$$

which is the objective function to be minimized. $\bar{D}(P_t) = 0$ means that a perfect static balance has been reached [29].

As suggested by [29], we scale the problem making $r = 1$. We use $n = 14$ blades, as a typical runner has between 14 and 18 blades [29], and this value of n is one of the most difficult to optimize [22]. Regarding the values of M_k , we follow [22], generating n numbers according to a normal distribution with a mean of 100 and a standard deviation of $5/3$, so that most M_k s fall within $\pm 5\%$ of the mean. We generated these numbers using a Gaussian Random Number Generator available at the Random.org website [35].

4.1.4. Nuclear Reactor Core Design Optimization Problem

This is a highly multimodal problem [45], which has been attacked with many methods (see, for example, [12, 13, 44, 43]). Consider a cylindrical 3-enrichment-zone nuclear reactor, with a typical cell composed by moderator (light water), cladding and fuel. The design parameters that may be varied in the optimization process, as well as their variation ranges, are shown in Table 1. The materials are represented by discrete variables.

The objective of the optimization problem is to minimize the average flux or power peaking factor, f_p , of the proposed reactor, allowing the reactor to be sub-critical or super-critical ($k_{eff} = 1.0 \pm 1\%$), for a given average flux ϕ_0 .

Table 1. Parameters range

Parameter	Symbol	Range
Fuel Radius (cm)	R_f	0.508 to 1.270
Cladding Thickness (cm)	Δ_c	0.025 to 0.254
Moderator Thickness (cm)	Δ_m	0.025 to 0.762
Enrichment of Zone 1 (%)	E_1	2.0 to 5.0
Enrichment of Zone 2 (%)	E_2	2.0 to 5.0
Enrichment of Zone 3 (%)	E_3	2.0 to 5.0
Fuel Material	M_f	{U-Metal or UO ₂ }
Cladding Material	M_c	{Zircaloy-2, Aluminum or Stainless Steel-304}

Let $\mathbf{D} = \{R_f, \Delta_c, R_e, E_1, E_2, E_3\}$ be the vector of design variables. Then, the optimization problem can be written as follows:

Minimize $f_p(\mathbf{D})$ s.t.

$$\phi(\mathbf{D}) = \phi_0; \quad (15)$$

$$0.99 \leq k_{eff}(\mathbf{D}) \leq 1.01; \quad (16)$$

$$\frac{dk_{eff}}{dV_m} > 0; \quad (17)$$

$$\mathbf{D}_i^l \leq \mathbf{D}_i \leq \mathbf{D}_i^u, i = 1, 2, \dots, 6; \quad (18)$$

$$M_f = \{\text{U-Metal or UO}_2\}; \quad (19)$$

$$M_c = \{\text{Zircaloy-2, Al or SS-304}\}, \quad (20)$$

where V_m is the moderator volume, and the superscripts l and u indicate respectively the lower and upper bounds (of the feasible range) for each design variable.

4.2. Implementation and Setup

Our tests were performed on an Intel[®] Core[™] i7 PC with 12 Gb RAM running Ubuntu 14.04 LTS. TopoClearing-DE was implemented in C++ and compiled with GNU g++ version 4.8.2. For the stochastic part of this algorithm, we used the pseudorandom number generating algorithm developed by Matsumoto and

Nishimura [26], the Mersenne Twister, which is available for download at one of its creator's website [27].

For the nuclear core optimization problem, our source code was connected to the HAMMER reactor physics code [50], which calculates the objective function value for each solution proposed by the optimization algorithm.

Regarding the turbine balancing problem, as DE was conceived as a continuous optimization algorithm [49], first, we need to adapt it for combinatorial optimization. In order to do so, we employ a representation technique named random keys [5]. This mechanism, originally designed for the genetic algorithm, allows us to treat discrete problems as if they were continuous. The solution is translated into a discrete sequence only in the moment of the objective function evaluation. Let us show how it works with a simplified example: a six-city TSP. DE works with six continuous variables, all in the range $[0, 1]$. Let us suppose we have a solution \mathbf{S}_1 , given by

$$\mathbf{S}_1 = (0.18, 0.73, 0.42, 0.87, 0.01, 0.23). \quad (21)$$

Each one of these variables receive an integer index, in subscripts, corresponding to their order of appearance:

$$\mathbf{S}_1 = (0.18_1, 0.73_2, 0.42_3, 0.87_4, 0.01_5, 0.23_6). \quad (22)$$

Then, these real numbers (the so-called random keys) are sorted:

$$\mathbf{S}_{1_{\text{sorted}}} = (0.01_5, 0.18_1, 0.23_6, 0.42_3, 0.73_2, 0.87_4). \quad (23)$$

The subscripts represent a valid sequence \mathbf{T}_1 :

$$\mathbf{T}_1 = (5, 1, 6, 3, 2, 4). \quad (24)$$

Note that, even in an extreme case with repeated real numbers, a valid sequence is produced:

$$\mathbf{S}_2 = (0.93, 0.27, 0.93, 0.45, 0.11, 0.93), \quad (25)$$

$$\mathbf{S}_2 = (0.93_1, 0.27_2, 0.93_3, 0.45_4, 0.11_5, 0.93_6), \quad (26)$$

$$\mathbf{S}_{2_{\text{sorted}}} = (0.11_5, 0.27_2, 0.45_4, 0.93_1, 0.93_3, 0.93_6), \quad (27)$$

$$\mathbf{T}_2 = (5, 2, 4, 1, 3, 6). \quad (28)$$

We used the following parameters in our tests: population size $NP = 100$ and 500 , crossover rate $CR = 0.9$, and scaling factor $F = 0.5$, which are values that have been widely employed in the literature [60, 3, 39], among others. The same one-hundred random seeds (one per execution) were used for canonical DE and its variant. For the topographical heuristic inside the niching method, we used $k = 10$ as Törn and Viitanen [54, 55], and also tested $k = 20$.

Regarding the nuclear For the nuclear reactor core design, the algorithms were set up to stop at 100,000 objective function evaluations, so that the results were obtained with the same maximum computational effort as previous results [41, 40, 42, 44, 43].

As the other optimization problems attacked in this work have known global minima, DE was run using the same termination criterion as in [47, 17, 9, 18, 36], which is ideal for an algorithm's performance assessment:

$$|f(\mathbf{x}^*) - f(\mathbf{x})| \leq \varepsilon_1 |f(\mathbf{x}^*)| + \varepsilon_2, \quad (29)$$

where $f(\mathbf{x}^*)$ is the global optimum, $f(\mathbf{x})$ is the current best, coefficient $\varepsilon_1 = 10^{-4}$ corresponds to the relative error and $\varepsilon_2 = 10^{-6}$ corresponds to the absolute error [47].

For these problems, we set a maximum number of generations equal to 100,000 for all population sizes as a stopping criterion, in case the condition given by Eq. (29) is not achieved.

4.3. Computational Results

4.3.1. Chemical Equilibrium Problem

Table 2 compares the results obtained by TopoClearing-DE with $k = 10$ and $k = 20$ against those achieved by the conventional DE. The population size is denoted by PS. We performed one-hundred executions of each algorithm with the same independent random seeds for all of them, so that the experiments are unbiased. SR is the success rate for each algorithm and/or setup. Regarding the number of fitness evaluations (NFE), we display the minimum, maximum, and average NFEs taking into account only the successful runs.

Note that TopoClearing-DE obtains a success rate of 100/100 even for a population of one-hundred individuals. The topographical heuristic with $k = 20$ requires more function evaluations than with $k = 10$.

Table 2. Results for the chemical equilibrium problem

		DE		TopoClearing-DE, $k = 10$		TopoClearing-DE, $k = 20$	
		100	500	100	500	100	500
PS		2/100	100/100	100/100	100/100	100/100	100/100
SR	Min.	15,197	396,801	778,501	3,136,575	1,497,250	5,612,756
NFE	Max.	81,186	514,717	1,049,478	3,567,895	2,036,221	6,844,980
	Avg.	48,191	455,152	910,942	3,352,001	1,669,972	6,132,049

The only drawback of the method described in this chapter is the higher computational cost, but this can be explained due to a certain tendency of the canonical DE to converge prematurely to local optima [10].

4.3.2. Catalytic Reactor Model

Table 3 displays the results for the catalytic reactor model.

Table 3. Results for the catalytic reactor model

		DE		TopoClearing-DE, $k = 10$		TopoClearing-DE, $k = 20$	
		100	500	100	500	100	500
PS		7/100	100/100	100/100	100/100	100/100	100/100
SR	Min.	3,867,260	509,125	766,005	10,000,000	1,418,602	10,000,000
NFE	Max.	9,725,345	631,204	1,257,134	12,273,204	2,741,023	23,504,887
	Avg.	7,886,874	579,446	1,043,178	11,514,262	2,060,437	21,917,686

Once more, only TopoClearing-DE achieves 100% success with a small population.

4.3.3. Turbine Balancing Problem

Table 4 compares the results obtained by TopoClearing-DE with $k = 10$ and $k = 20$ against those achieved by the conventional DE.

Table 4. Results for the turbine balancing problem

		DE		TopoClearing-DE, $k = 10$		TopoClearing-DE, $k = 20$	
		100	500	100	500	100	500
PS		10/100	85/100	29/100	86/100	31/100	90/100
SR	Min.	87,310	78,140	27,692	222,135	102,613	147,673
NFE	Max.	8,280,554	46,394,433	9,904,947	49,093,657	9,268,706	48,357,305
	Avg.	3,296,290	14,107,599	4,315,295	17,610,059	4,137,400	18,601,053

Note that TopoClearing DE with both values of k outperformed the canonical DE for all population sizes, with a slightly better performance for $k = 20$.

Also note that the results obtained with one-hundred individuals show that Topographical Clearing generated a high diversity even with a small population for such a complex problem.

4.3.4. Nuclear Reactor Core Design Optimization Problem

Table 5 shows the results obtained by ten independent executions of each variant in terms of fitness (i.e., objective-function value in evolutionary computation terminology) and NFEs to reach the optimum.

Table 5. Results for the nuclear core design problem

Experiment	DE		TopoClearing-DE, $k = 10$		TopoClearing-DE, $k = 20$	
	Fitness	NFE	Fitness	NFE	Fitness	NFE
#1	1.2765	67,967	1.2765	48,094	1.2766	32,619
#2	1.2767	24,832	1.2765	99,375	1.2765	44,119
#3	1.2765	46,173	1.2763	58,372	1.2763	39,562
#4	1.2763	49,570	1.2763	40,603	1.2765	76,368
#5	1.2765	77,814	1.2766	41,591	1.2763	82,141
#6	1.2767	22,468	1.2766	34,185	1.2765	50,525
#7	1.2767	28,716	1.2765	71,802	1.2763	42,476
#8	1.2766	28,410	1.2765	42,306	1.2765	64,447
#9	1.2767	25,012	1.2766	29,601	1.2764	99,607
#10	1.2767	25,271	1.2765	58,720	1.2766	33,268
Average	1.2766	39,623.3	1.2765	52,464.9	1.2765	56,513.2

Comparing the results, we can see that TopoClearing DE outperformed canonical DE, particularly with $k = 20$, where the best value of 1.2763 was reached 3/10 times. As in the previous problem and for the same reason, DE with the niching method requires more objective-function evaluations.

5. Conclusion

In this chapter, we present a niching method to overcome local optima of multimodal optimization problems, which are quite common in the real world [24]. This method is applied to challenging real-world optimization problems.

The results obtained here demonstrate the potential of topographical clearing, which is easy to implement and can be used in other evolutionary or swarm-based optimization methods besides differential evolution. Last but not least, it does not require the burden of estimating the radius (a very difficult task, especially for practical problems).

Acknowledgments

W.F.S. and N.H. gratefully acknowledge the financial support provided by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Ministry of Science, Technology and Innovation, Brazil). The authors also would like to thank Prof. Montaz Ali for providing a copy of his doctoral thesis. The research by N.H. has been carried out within the framework of project PROCIENCIA-UERJ financed by FAPERJ.

Appendix A. Nonlinear Systems Formulated as Optimization Problems

Let us consider the problem of computing solutions of nonlinear systems with simple bound constraints. We can express this problem as

$$\begin{cases} f_1(\mathbf{x}) = 0 \\ f_2(\mathbf{x}) = 0 \\ \vdots \\ f_N(\mathbf{x}) = 0 \end{cases} \quad \text{s.t. } \mathbf{x} \in [\mathbf{a}, \mathbf{b}] \subseteq \mathbb{R}^n, \quad (\text{A.1})$$

where $\mathbf{x} = (x_1, \dots, x_N)^T \in \mathbb{R}^n$, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ and $[\mathbf{a}, \mathbf{b}] \equiv [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_N, b_N]$, with $a_i < b_i$, for all $i = 1, \dots, N$. Note that vectors $\mathbf{a} = (a_1, a_2, \dots, a_N)$ and $\mathbf{b} = (b_1, b_2, \dots, b_N)$ are specified as the lower and upper bounds of the variables, and set $[\mathbf{a}, \mathbf{b}]$ is a box in \mathbb{R}^n , where there exist one or more roots of the nonlinear system. Let us suppose that function $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, for any $i = 1, \dots, N$, can be nondifferentiable or even discontinuous, but it must be bounded in $[\mathbf{a}, \mathbf{b}]$. If $F = (f_1(\mathbf{x}), \dots, f_N(\mathbf{x}))^T$, the problem described by Eq. (A.1) can be reformulated as the following optimization problem:

$$\text{Min } f(\mathbf{x}) \text{ s.t. } \mathbf{x} \in [\mathbf{a}, \mathbf{b}] \subseteq \mathbb{R}^n \quad (\text{A.2})$$

In Eq. (A.2), $f : [\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is a nonnegative and possibly multimodal merit function, given by

$$f(\mathbf{x}) = F^T(\mathbf{x})F(\mathbf{x}), \quad (\text{A.3})$$

Since the system represented by Eq. (A.1) has solution(s) in $[\mathbf{a}, \mathbf{b}]$, then, in terms of results, to solve this system is equivalent to find the global minimum(a) of the optimization problem given by Eq. (A.2).

Appendix B. Data and Fitted Variables for the Catalytic Reactor Model

Table B.1. Data and fitted values for the Catalytic Reactor Model [37]

x_1		x_2		x_3	
Data	Fitted	Data	Fitted	Data	Fitted
0.015	0.018	0.235	0.236	6.25	2.54
0.030	0.031	0.220	0.220	4.90	3.11
0.045	0.045	0.205	0.205	2.90	3.21
0.100	0.100	0.150	0.150	1.75	1.94
0.180	0.180	0.070	0.070	0.30	0.35
0.015	0.023	0.485	0.486	12.30	8.88
0.030	0.034	0.470	0.471	14.00	10.83
0.045	0.038	0.455	0.453	5.00	10.77
0.045	0.047	0.455	0.456	14.20	11.82
0.100	0.100	0.400	0.400	10.81	10.73
0.143	0.143	0.357	0.357	7.81	8.12
0.167	0.167	0.333	0.333	6.41	6.72
0.250	0.250	0.250	0.250	3.90	3.06
0.333	0.333	0.167	0.167	3.60	1.09
0.030	0.023	0.720	0.720	13.00	14.74
0.045	0.044	0.705	0.705	20.00	20.72
0.100	0.100	0.650	0.649	19.81	22.45
0.180	0.180	0.570	0.570	15.10	15.88
0.240	0.241	0.510	0.510	8.90	11.10
0.300	0.300	0.450	0.450	7.50	7.51
0.360	0.360	0.390	0.389	2.00	4.85
0.026	0.015	0.974	0.974	13.00	14.56
0.050	0.048	0.950	0.950	30.00	30.67
0.100	0.100	0.900	0.901	37.50	34.89
0.250	0.248	0.750	0.752	25.00	20.52
0.150	0.150	0.850	0.850	31.50	30.98
0.333	0.334	0.667	0.666	10.00	13.34
0.500	0.500	0.500	0.500	4.00	5.26

References

- [1] M. M. Ali. *Some modified stochastic global optimization algorithms with applications*. Loughborough University of Technology, 1994.
- [2] M. M. Ali and C. Storey. Topographical multilevel single linkage. *Journal of Global Optimization*, 5(4):349–358, 1994.

- [3] M. M. Ali and A. Torn. Population set-based global optimization algorithms: some modifications and numerical studies. *Computers and Operations Research*, 31(10):1703–1725, 2004.
- [4] S. V. Amiouny, J. J. Bartholdi III, and J. H. Vande Vate. Heuristics for balancing turbine fans. *Operations Research*, 48(4):591–602, 2000.
- [5] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160, 1994.
- [6] R. W. Becker and G. Lago. A global optimization algorithm. In *Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*, pages 3–12, 1970.
- [7] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. 1981.
- [8] W. Choi and R. H. Storer. Heuristic algorithms for a turbine-blade-balancing problem. *Computers and Operations Research*, 31(8):1245–1258, 2004.
- [9] T. Csendes, L. Pal, J. O. H. Sendin, and J. R. Banga. The global optimization method revisited. *Optimization Letters*, 2(4):445–454, 2008.
- [10] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on*, 15(1):4–31, 2011.
- [11] K. A. De Jong. *Analysis of the behavior of a class of genetic adaptive systems*. University of Michigan, 1975.
- [12] C. M. do Nascimento Abreu Pereira, R. Schirru, and A. S. Martinez. Basic investigations related to genetic algorithms in core designs. *Annals of Nuclear Energy*, 26(3):173–193, 1999.
- [13] R. P. Domingos, R. Schirru, and C. M. Pereira. Particle swarm optimization in reactor core design. *Nuclear science and engineering*, 152(2):197–203, 2006.
- [14] C. Gau and M. Stadtherr. Deterministic global optimization for errors-in-variables parameter estimation. *AIChE Journal*, 48:1192–1197, 2002.

-
- [15] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 41–49. L. Erlbaum Associates Inc., 1987.
- [16] C. Grosan and A. Abraham. A new approach for solving nonlinear equations systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(3):698–714, 2008.
- [17] M. J. Hirsch, C. Meneses, P. M. Pardalos, and M. G. Resende. Global optimization by continuous grasp. *Optimization Letters*, 1(2):201–212, 2007.
- [18] M. J. Hirsch, P. M. Pardalos, and M. G. Resende. Solving systems of nonlinear equations with continuous grasp. *Nonlinear Analysis: Real World Applications*, 10(4):2000–2006, 2009.
- [19] D. S. Johnson and M. Garey. *Computers and Intractability: A guide to the theory of NP-completeness*. FreemanandCo, San Francisco, 1979.
- [20] P. Kaelo and M. Ali. A numerical study of some modified differential evolution algorithms. *European journal of operational research*, 169(3):1176–1184, 2006.
- [21] J. Lampinen and I. Zelinka. Mixed variable non-linear optimization by differential evolution. *Proceedings of Nostradamus*, 99(2):7–8, 1999.
- [22] G. Laporte and H. Mercure. Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35(3):378–381, 1988.
- [23] A. Levi and S. Haas. *Appendix a-global optimization algorithms, optimal device design*. pages 262–276, 2010.
- [24] S. W. Mahfoud. *Niching methods for genetic algorithms*, volume 51. 1995.
- [25] C. D. Maranas and C. A. Floudas. Finding all solutions of nonlinearly constrained systems of equations. *Journal of Global Optimization*, 7(2):143–182, 1995.

- [26] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.
- [27] M. u. r. .-.-. Matsumoto. *Mersenne twister home page*, 2011.
- [28] K. Meintjes and A. P. Morgan. Chemical equilibrium systems as numerical test problems. *ACM Transactions on Mathematical Software (TOMS)*, 16(2):143–151, 1990.
- [29] J. Mosevich. Balancing hydraulic turbine runners – a discrete combinatorial optimization problem. *European Journal of Operational Research*, 26(2):202–204, 1986.
- [30] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publications, 1998.
- [31] C. M. Pereira and C. M. Lapa. Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy*, 30(5):555–565, 2003.
- [32] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 798–803. IEEE, 1996.
- [33] L. S. Pitsoulis, P. M. Pardalos, and D. W. Hearn. Approximate solutions to the turbine balancing problem. *European Journal of Operational Research*, 130(1):147–155, 2001.
- [34] B.-Y. Qu, P. N. Suganthan, and J.-J. Liang. *Differential evolution with neighborhood mutation for multimodal optimization*. 2012.
- [35] Random.org. *Gaussian random number generator*. Website, 2014. <http://www.random.org/gaussian-distributions/> (retrieved 09-09-2014).
- [36] A. Rios-Coelho, W. Sacco, and N. Henderson. A metropolis algorithm combined with hooke-jeeves local search method applied to global optimization. *Applied Mathematics and Computation*, 217(2):843–853, 2010.

-
- [37] V. Rod and V. Hancil. Iterative estimation of model parameters when measurements of all variables are subject to error. *Computers and Chemical Engineering*, 4(2):33–38, 1980.
- [38] J. Rönkkönen. Continuous multimodal global optimization with differential evolution-based methods. *Lappeenranta University of Technology*, 2009.
- [39] W. Sacco and N. Henderson. Finding all solutions of nonlinear systems using a hybrid metaheuristic with fuzzy clustering means. *Applied Soft Computing*, 11(8):5424–5432, 2011.
- [40] W. Sacco, N. Henderson, A. Rios-Coelho, M. Ali, and C. Pereira. Differential evolution algorithms applied to nuclear reactor core design. *Annals of Nuclear Energy*, 36(8):1093–1099, 2009.
- [41] W. Sacco, A. Rios-Coelho, N. Henderson, and M. Ali. The particle collision algorithm. In *International Workshop on Stochastic and Applied Global Optimization (SAGO 2008)-Book of Abstracts*. University of Witwatersrand, Johannesburg, 2008.
- [42] W. F. Sacco, A. A. de Moura Meneses, and N. Henderson. Some studies on differential evolution variants for application to nuclear reactor core design. *Progress in Nuclear Energy*, 63:49–56, 2013.
- [43] W. F. Sacco and N. Henderson. Differential evolution with topographical mutation applied to nuclear reactor core design. *Progress in Nuclear Energy*, 70:140–148, 2014b.
- [44] W. F. Sacco, N. Henderson, and A. C. Rios-Coelho. Topographical global optimization applied to nuclear reactor core design: Some preliminary results. *Annals of Nuclear Energy*, 65:166–173, 2014a.
- [45] W. F. Sacco, M. D. Machado, C. M. Pereira, and R. Schirru. The fuzzy clearing approach for a niching genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy*, 31(1):55–69, 2004.
- [46] B. Sareni and L. Krahenbuhl. Fitness sharing and niching methods revisited. *Evolutionary Computation, IEEE Transactions on*, 2(3):97–106, 1998.

- [47] P. Siarry, G. Berthiau, F. Durdin, and J. Haussy. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. *ACM Transactions on Mathematical Software (TOMS)*, 23(2):209–228, 1997.
- [48] M. Sinclair. Comparison of the performance of modern heuristics for combinatorial optimization on real data. *Computers and Operations Research*, 20(7):687–695, 1993.
- [49] R. Storn and K. Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
- [50] J. E. Suich and H. Honeck. *The HAMMER System: Heterogeneous Analysis by Multigroup Methods of Exponentials and Reactors*. Du Pont de Nemours (EI) and Co., Aiken, SC Savannah River Lab., 1967.
- [51] R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1382–1389. IEEE, 2004.
- [52] G. T. Timmer. *Global optimization: A stochastic approach*. 1984.
- [53] I.-B. Tjoa and L. Biegler. Reduced successive quadratic programming strategy for errors-in-variables estimation. *Computers and chemical engineering*, 16(6):523–533, 1992.
- [54] A. Torn and S. Viitanen. Topographical global optimization. *Recent advances in global optimization*, pages 384–398, 1992.
- [55] A. Torn and S. Viitanen. Topographical global optimization using pre-sampled points. *Journal of Global Optimization*, 5(3):267–276, 1994.
- [56] A. Torn and S. Viitanen. Iterative topographical global optimization. *Non-convex Optimization and Its Applications*, 7:353–364, 1996.
- [57] A. Torn and A. Zilinskas. *Global optimization*. Springer-Verlag New York, Inc., 1989.
- [58] A. A. Torn. *A search-clustering approach to global optimization*. Abo Swedish University School of Economics, 1977.

-
- [59] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM Journal on Numerical Analysis*, 34(2):797–827, 1997.
- [60] J. Vesterstrom and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1980–1987. IEEE, 2004.
- [61] G. Y. Yang, Z. Y. Dong, and K. P. Wong. A modified differential evolution algorithm with fitness sharing for power system planning. *Power Systems, IEEE Transactions on*, 23(2):514–522, 2008.

Chapter 5

ROBOTICS, EVOLUTION AND INTERACTIVITY IN SONIC ART INSTALLATIONS

Artemis Moroni^{1,} and Jônatas Manzolli²*

¹Robotics and Computer Vision Division /

Center for Information Technology Renato Archer – DRVC/CTI

²Interdisciplinary Nucleus for Sound Studies;

Music Department / University of Campinas – NICS; IA/UNICAMP

Abstract

Focusing on the interactivity that a robotic interface establishes between the virtual and the real world, some sensory systems and mobile robotic platforms were developed for the AURAL project, a robotic evolutionary environment for sound production. From the AURAL perspective, human and robots are agents of a complex system and the sonification is the emergent propriety produced by their interaction and behavior. One way to characterize types of interactions is by looking at ways in which systems can be coupled together to interact. The representation of the interaction between a person and a dynamic system as a simple feedback loop faces the role of information looping through both a person and a system. Two different sonification paradigms were applied in AURAL environment. In the first case, the sonification is generated by an evolutionary mapping of the robot trajectories into sound events. In the second case, the sound production is the result of a generative process. As such the sonification here is not seen as an isolated aspect of AURAL, but as a representation of the synergetic capacity of the agents to collaborate and produce a complex product.

* E-mail address: Artemis.Moroni@cti.gov.br

A comparison between the results obtained with both approaches is presented. The structure/novelty tradeoff has been approached.

Introduction

New ground is currently being broken in the areas of robotics, musical composition and interactive narratives. With the advent of new interactive and sensing technologies, computer-based music systems have evolved from sequencers to algorithmic composers and complex interactive systems that sense their environment and can automatically generate music. Consequently, the frontiers between composers, computers and autonomous creative systems have become more and more blurred, while the concepts of musical composition and creativity are being put into a new perspective. The use of synthetic interactive music systems allows for the direct exploration of a sentient approach to music composition. Venturing into the controlling of motion and sound, as well as robotics, the transformation of everyday items, the mixing of realities to straddle the physical and virtual worlds, other kinds of exploration are being investigated taking into account the principles of emergence, embodiment and feedback. Mixed reality systems are being integrated with nowadays activities in areas such as robotics, game technologies and artistic installation.

Concerning to computer-aided composition, based on a rich history of classical music theory and teaching, one of the first goals was to help the composer during the creative process. Probably the most widespread computer-aided composition paradigm is still that of a music sequencer. This model is somehow a continuation of the traditional composition based on the writing of musical scores. Within the sequencer paradigm, the user/composer creates an entire piece by entering notes, durations or audio samples on an electronic score. Due to its digital nature, this score can later be subjected to various digital manipulations. Within this paradigm, the computer is “passive”, the human being is in control of the entire compositional processes and uses the computer as a tool to lay down ideas and speedup specific tasks (copying, pasting or transposing parts).

In contrast with the standard sequencer approach, computer-based algorithmic composition relies on mathematical formalisms that allow the computer to automatically generate musical material, usually without external output. The composer does not specify directly all the parameters of the musical material, but a set of simple rules or input parameters, which will be taken into account by the algorithm to generate musical material. In the latter paradigm, the computer carries out most of the detailed work and the composer controls a limited set of

initial global parameters. Different approaches to algorithmic composition inspired by technical advances have been proposed and tested; the main ones are statistical methods, rule-based methods, neural networks and genetic algorithms (Papadopoulos and Wiggins, 1999; Nierhaus, 2009). With the advent of new programming languages, communication standards and sensing technologies, it has now become possible to design complex real-time music systems that can foster rich interactions between humans and machines (Rowe, 1993; Winkler, 2001; Zicarelli, 2002; Wright, 2005; Puckette, 1996). Interaction is understood here as “reciprocal action or influence” as defined in the Oxford New Dictionary of American English (Jewell et al., 2001). Nowadays, one may build sensate composition systems able to analyze external sensor inputs in real-time and use this information as an ingredient of the composition (Le Groux, 2011). These kinds of interactive systems are in accordance with the philosophy that a theory of mind, including one of creativity and aesthetics, will be critically dependent on its accomplishment as a real-world artifact because only in this way may such a theory of an open and interactive system as the mind be fully validated (Verschure and Manzolli, 2013; Boden, 1991).

The AURAL system, described in this chapter, was created by focusing on the interactivity that a robotic interface establishes between the virtual and the real world. An interactive evolutionary graphical interface applied to sound production, an omnidirectional vision system and mobile robots are integrated in an arena constructed so as to allow interactive control of real time sonification and robotic navigation. A similar architecture, with an artificial vision system and mobile robots, but with a different sonification paradigm based on generative systems is applied in AURAL₂. In both versions, humans and robots are agents of a complex system and the sonification is the emergent propriety that is produced by their interaction and behavior. This exploration is also related with the concept of self-organization in complex systems.

This chapter is organized in the following way. Section 1 describes the evolutionary compositional interface of the AURAL environment, its components, features and AURAL as an art installation. Section 2 describes AURAL₂ and its generative process. Section 3 presents the automation and interactivity, the parts of an art system, comparing them with dynamic systems under the general systems theory. Section 4 compares both AURAL and AURAL₂ environments, concerning to the characteristics of the sound results. Finally, the conclusions are presented.

1. JaVOX, an Evolutionary Composition System

In the AURAL environment, the behavior of mobile robots in an arena is applied as a compositional strategy and the sonification is generated by means of a mapping of the trajectories of the robots into sound events (Moroni & Manzolli, 2010). Starting with VOX POPULI (Moroni et al., 2000), an evolutionary composition system, another environment, JaVOX, evolved. Like its predecessor VOX POPULI, JaVOX is based on three musical aspects: melody, harmony and voice range. The specifications of these criteria define the fitness of a group to the applied selection function. This function returns the “better individual”, or “better chord”, according to the measured aspects. The selected group is treated as a set of MIDI notes and played. The system allows the user to modify the fitness function by using four controls, one for the melodic criterion; another for the duration of the genetic cycle and music rhythm; a third one for the set of octave range to be considered, and the last one for the time segment for each selected orchestra. All these controls are available for real time performance allowing the user to play and interact with the musical evolution, but the controls may also be automatically modified during a performance, depending on the behavior of the robots.

1.1. Population as MIDI Data

For our purposes, an auditory event may be described using four parameters: pitch, timbre, loudness and duration. *Pitch* can be defined as the auditory propriety of a note that is conditioned by its frequency relative to the other notes. The range of musical pitch has been defined as the range within which the interval of an octave can be perceived. This has been found to correspond roughly to the range of the piano. From this continuum of frequencies, a set of discrete frequencies is selected so that the frequencies bear a definite interval relationship to one another. So, pitch in the musical sense corresponds to a frequency that is selected from a predefined repertoire. In this scheme, two discrete frequencies are chosen in the interval of an octave such that the ratio between any two adjacent frequencies is $\sqrt[12]{2}$. This interval ratio in music terminology is termed a semitone in a temperament system or chromatic scale. *Loudness* is that aspect of an auditory event related to its intensity. *Duration* is characterized by the period of time in which the event is perceivable. *Timbre* is a complex feature of the sound domain, but in this chapter it is taken as being the individuality of sound acquired by the addition of harmonics to the fundamental pitch. Here it is specifically

defined as the characteristic of a given musical instrument and the mode of playing it.

Using the foregoing notions, we define here a melody as a fixed temporal ordering of auditory events. So, a melody in conventional occidental notation resembles a system of cartesian coordinates. The pitch and duration are carefully marked; timbre is decided by the instrument for which it is written and loudness is crudely marked (Vidyamurthy, 1992). However, this is a very subjective issue; the judgment of harmony does not seem to have a natural basis, but seems to be a common response people acquired in a certain cultural context. Therefore, opinions on the subject may vary widely depending on social and cultural backgrounds.

1.2. The Evolutionary Sound Process

In JaVOX, we used the MIDI protocol to code a musical genotype. In this evolutionary sound system, the individuals of the population are defined as a set of four notes. These notes are randomly generated in the interval [0, 127] where every value represents a MIDI event. In each generation, 30 individuals are created.

Two cycles are integrated in the evolutionary sound process. The *reproduction cycle* is the evolving process that generates a set of four notes using genetic operators and selecting individuals. In the *MIDI cycle* the interface looks for notes to be played. When a set is selected, the program places it in a critical area which is continually verified by the MIDI interface. These notes are played until the next set is selected. Figure 1 depicts the reproduction cycle and the MIDI cycle.

The musical fitness for each pitch set, described in (Moroni et al., 2002) is a conjunction of three partial fitness functions: melody, harmony and vocal range, each returning a numerical value.

$$\text{Musical Fitness} = \text{Melodic Fitness} + \text{Harmonic Fitness} + \text{Vocal Range Fitness}$$

An analogy may be made of each individual with a chord of four voices (or a chord played by four instruments). The chord with the highest fitness is selected and played as a new MIDI event. At each generation of the process a new sonority is created by applying the fitness criteria regarding the melodic line (*mel*), voice range (*oct*), duration of the evolutionary cycle (*bio*) and music meter (*rhy*). Based on the order of musical interval consonance, the notion of approximating a

sequence of notes to its harmonically compatible note, or a tonal center (*mel*), is used. The selected notes are sent to the MIDI board and can be heard as sound events in real time. The duration of the evolutionary cycle (*bio*) and music meter (*rhy*) is taken into account. This sequence produces a sound resembling a chord cadence or fast block counterpoints.

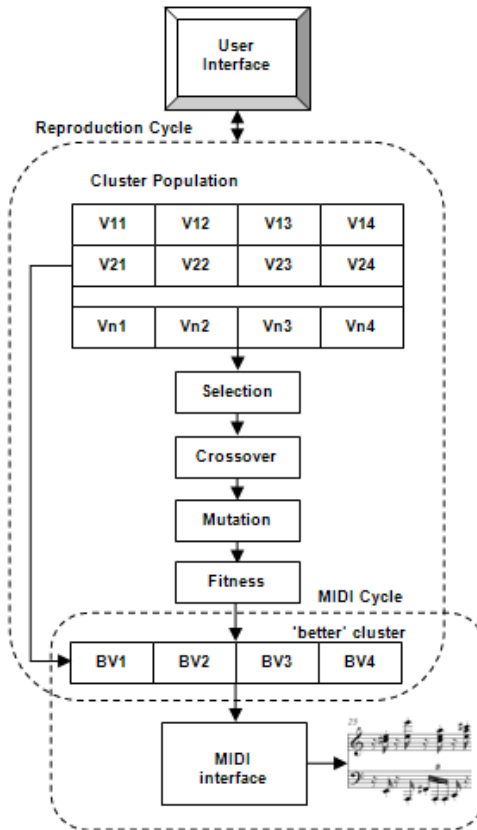


Figure 1. The reproduction cycle and the MIDI cycle in the evolutionary process for sound production.

1.3. The Structure/Novelty Tradeoff

An issue of central importance in the construction of any evolutionary system is the structure/novelty tradeoff. When filling the void using chaotic materials necessary for invention, the more such materials are introduced, the more the

structure and knowledge will be added to the system and structure will be present in the system's output behavior. That is, more highly structured systems can produce more highly constrained output. Some of these concepts were firstly stated by one of the pioneers of cybernetics, Ashby (1956). By applying these ideas to algorithmic composition systems means that more knowledge and structure allows the creation of new pieces that are more tightly matched to the desired musical genre. However, the flipside of more structure is less new material. The highly constrained output will be less likely to stray beyond a genre's limitations or it may be surprising. Thus, the highly structured composition system will be less general, able to reach less 'music space' with its output (Todd & Werner, 1999).

In the AURAL, this tradeoff is treated by creating an interplay between sound, real-world artifacts, user and behavioral information, through the interaction among the evolutionary sound process, the artificial vision system and the mobile robots. The sound interface has a *Graphic Area*, the heart of the system, wherein the user may draw curves to be sent as trajectories to the robots. This area is associated with a conceptual sound space with two axis, the "red" one, or melodic, and the "blue" one, or rhythmic. The paths travelled by the robots in the arena are observed by the artificial vision system and sent, as sequences of points, to the sonification module. The red curves, sent as trajectories to the robots and the blue curves associated with the paths travelled guide the evolutionary sound process across different regions in the sound space.

Figure 2 shows JaVOX interface on the left and the curves from which the parameters are extracted for fitness evaluation. In the *Graphic Area*, three curves are shown: a) the trajectory the user draws, b) the path followed by a master robot and c) the path followed by another robot. Curves a) and b) are shown in detail on the right.

The fitness criteria, based on the ordering of musical interval of consonances (see previous section), introduces in the process some structure and knowledge in the process. At the same time, depending on the distance between the couple of robots (until four), the performance controls are activated. The *Performance Control* area offers other possibilities to control the sound production. For each of the four MIDI voices there are three controls: *solo*, *sequence* and *block*. They work as delay lines in which MIDI notes from previous generations are played again as solo, melodic patterns or chords. The relative position of the robots is used to select the solo, the sequence or block mode for each voice in real time. Table 1 shows the five simple rules associating the distance between the robots and the processes (the solo, sequence and block) of the Performance Control.

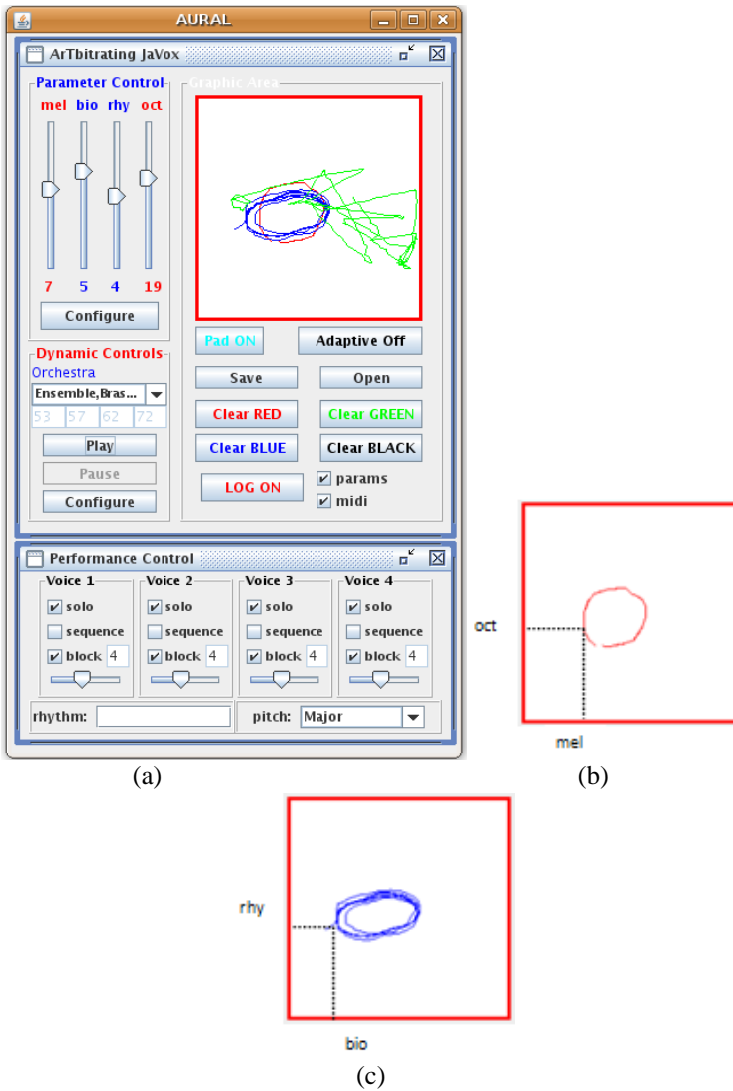


Figure 2. On the left (a), the JaVOX interface shows the different control areas: the Parameter Control; the Graphic Area and the Performance Control. The other graphics show the details of the curves in the Graphic Area. In the top (b) are the parameters extracted from the trajectory that was sent to a master robot. In the bottom (c) is the path followed by the robot, as well as the extracted parameters.

Table 1. Rules relating the distance between the robots and performance controls

Rule	Distance (m)	Solo	Sequence	Block
1	>0.5	X		
2	$0.4 < D < 0.5$		X	
3	$0.2 < D < 0.4$		X	X
4	$D < 0.2$			X

Other interface features enable the user to modify the number of notes of the Performance Control, as well as the rhythm, the pitch and the orchestra controls affecting the musical performance. The user interaction may be interpreted as attempts to improve the outcome by opening the possibility of the system to learn with it. It shows how robots physically fulfill the arena with a textural representation of the generated music. If the “*solo*” is activated, the resultant music structure will consist of independent voice lines. If the “*sequence*” is activated, the resultant music structure will consist of pitch pattern sequences. If the “*block*” is activated, the resultant music structure will consist of chord structures. In short, Table 1 expresses relationships between the motion behavior of robots and the music structures making the emerging composition depend on real world constraints.

During an AURAL performance, all the interactive paths can be recorded. It is possible to register all the automatic and interactive events, as well as the audio and MIDI files generated in real time. Some of them were used as a basic material for generating instrumental compositions. A piece titled “Robotic Variations” for piano, marimba and electronics (computer and robots) was made up of the obtained music structures. A dancer, three musicians (marimba, piano and computer) and four robots (see description below) performed the musical piece of the robot evolution at the AURAL installation.

1.4. AURAL as an Art Installation

AURAL was presented in an art gallery (Figure 3) where the visitors could appreciate the sound output and the interaction among the robots, as a kind of choreography. The visitors drew curves in the JaVOX interface, which were transmitted as trajectories to a master robot, the Nomad. While the robots (until 4) moved in the arena, virtually traveling along the conceptual sound space, people changed the orchestra, rhythm and pitch controls by investigating the sound

possibilities. Both a process of man-machine interaction and parallel exploration occurred.

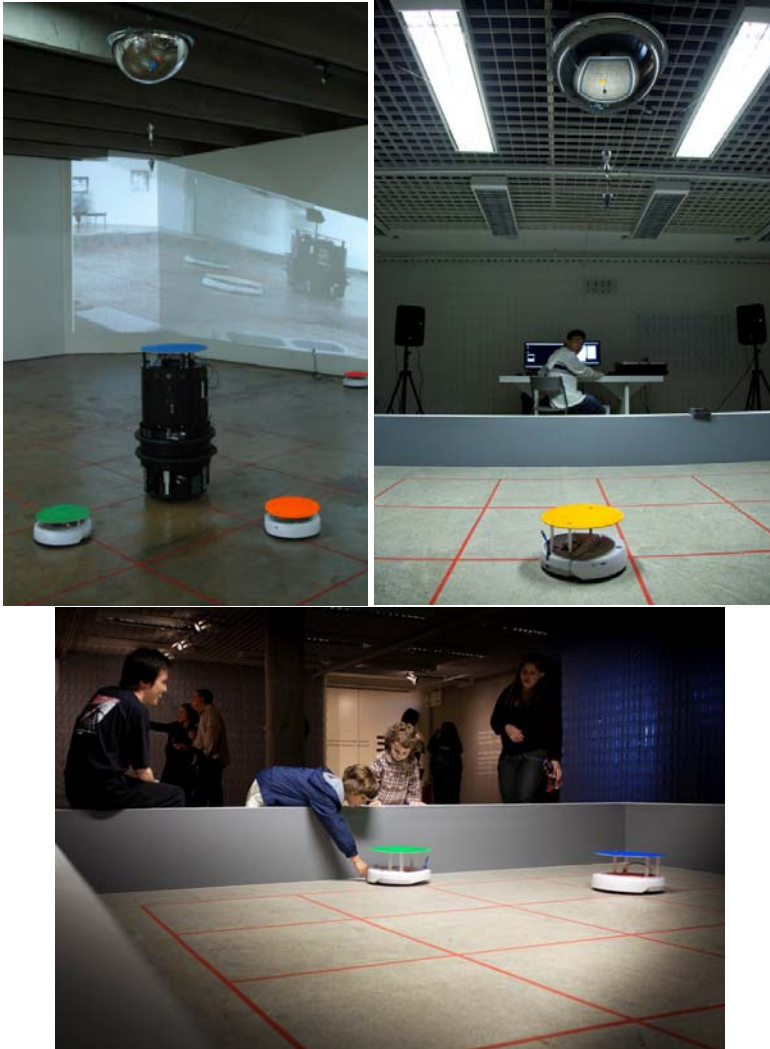


Figure 3. The first picture, on the left, shows AURAL installation at UNICAMP art gallery. The OmniEye, the artificial vision system, can be seen hanging on the ceiling. In the other pictures another setup of the AURAL at FILE festival, an international festival on electronic language, is depicted.

On the last day of the exhibition, a dancer, three musicians and the AURAL system itself, with four robots, performed an interactive concert called *Robotic Variations*. The same trajectories used to generate the material for the composition were used in the performance. An interactive scenery displayed real time processed images on the walls. The dancer was invited to interact with the robots in the arena, in a live performance. For the visual tracking, a strong color panel was fixed on the top of each robot. The dancer was invited to interact with the robots in the arena, in a live performance. For the visual tracking, a strong color panel was fixed on the top of each robot.



Figure 4. On the left, the dancer, the robots and the interactive scenery. On the right, the robots and the musicians. The third picture shows the dancer, the robots and the musicians during the rehearsals of the art performance.

Choreography was designed so that the robot with a red panel left the room and was replaced by the dancer using a red hat. Her position was tracked by the visual system through the red hat and interfered in the performance of the sound, incurring in another human-machine interaction cycle. Figure 4 shows some pictures of the musicians and of the dancer taken during the rehearsals of the performance. AURAL performance videos can be seen at (Moroni, 2012).

2. Generative Sonification

A similar architecture, with an artificial vision system and mobile robots, but with a different sonification paradigm, was applied in AURAL₂, a different version of the sound installation. If in the previous version the sound production is the result of an evolutionary process, in this second version the result is of a generative process interplaying with the human and real-world artifacts. The generative systems have many similarities with systems found in various areas of science; they may provide order and disorder, as well as a varying degree of complexity, making behavioral prediction difficult. However, such systems still contain a definite relation between cause and effect. The artist (or creator) generally provides basic rules, and then defines a process, random or semi-random, to work on these elements. The results continue to happen within the limits of the domain of the rules, but also may be subjected to subtle changes or even surprises. This paradigm of interaction between visitors and sound expression was also performed with the Roboser system in the “Ada: intelligent space” installation (Wasserman et al., 2003). Differently from ADA the visitors interaction is based on an individual basis while in ADA it was constructed as a collective behavior.

In the AURAL₂, sound fragments are inserted into a database, the memory of the system. The database is made up of four matrices, each one containing sound samples of different types: synthetic, game, environment and everyday sound fragments. Each cell in the matrix is associated with a cell in a virtual grid, projected on a winding format platform, or stage (3m x 3m wide, 0.3m high), depicted in Figure 5.

A hole inside the platform creates tracks that may be travelled on by only one robot or two robots. The robots have a border sensor, they stop when they detect the border. In the other regions of the platform, three or four robots can move around. This design causes conflicts among the robots when they try to escape from confined areas. The robots are tracked by a vision system which evaluates the position (x, y) of the robots on the stage; associates a cell in the matrix with that position and plays the sound fragment associated with it. The movement of the robots through the different regions of the stage triggers the sound of the associated cells, (re)creating soundscapes in the installation environment.

On a TV, the virtual grid is shown in several angles, as well as the cells activated by the robots (Figure 6). The visitors may interact with the system by talking, singing or screaming at a microphone, starting the intervention process: sound fragments are extracted from the interventions of the visitors and randomly inserted into the environment matrix; there is a possibility of the segments to be triggered and played again by the movement of the robots. A spectral analysis is

applied on the fragment that caused the intervention (Manzoli, 2011), and two visual effects may be perceived by the visitors. When there is more energy in the upper partials of the sound fragment, the following actions take place: the color of the cell associated with that fragment is changed to red on the TV, otherwise to blue. A rotation is applied on the grid. Finally, the sound fragment is inserted into the sound data base, i. e., the memory of the system, superposing a previous one, enhancing a recycling acoustic process. If no intervention occurs after a time interval (10 minutes), a sound matrix is randomly selected to supply the sound fragments and the intervention process is automatically performed by the system, which records a sound fragment from the environment and proceeds with the analysis.

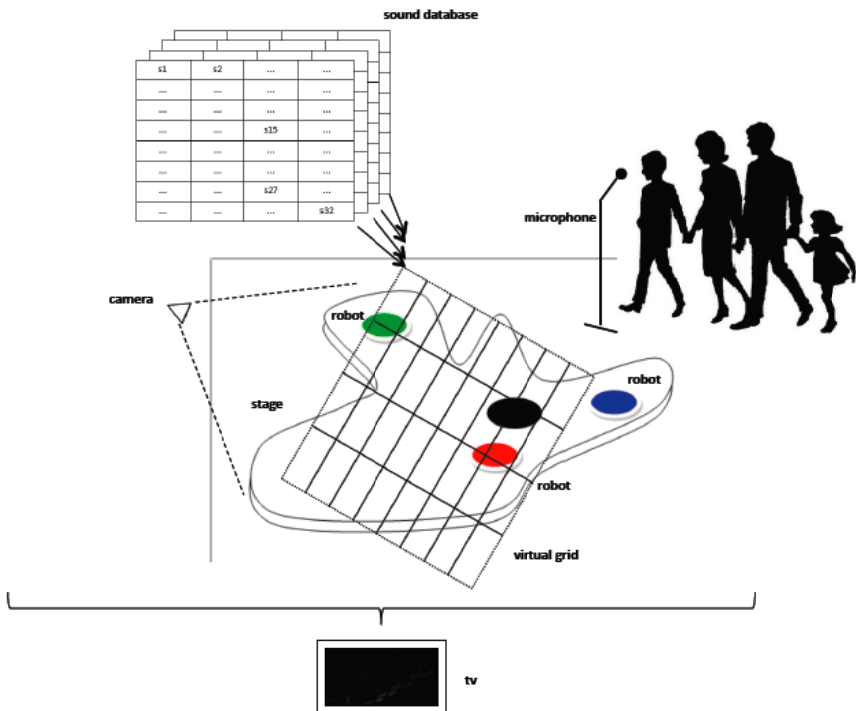


Figure 5. A virtual grid associates different sound databases with the platform: synthetic sounds, game sounds, everyday sounds and environment sounds. The movement of each robot - its location, is monitored by the vision system - triggers the sound associated with that place in the grid.



Figure 6. Above, AURAL₂ installation: the winding formatted stage, the robots and the television showing the active cells. On the bottom, a person interacting at the microphone.

3. Automation x Interactivity

One may see as an interesting aspect of the AURAL environments the possibility of different setups to explore distinct levels of interaction among humans and machines. One way of characterizing the types of interactions is by looking at ways in which systems can be coupled together to interact. Cornock and Edmonds (1973) early identified the ‘art system’ as consisting of the *artist*, the *participants*, the *artwork*, the *environment* in which these elements are placed, and the *dynamic processes* or *interactions* that result from the process (Candy & Edmonds, 2012).

Canonical models of computer-human interaction are based on an archetypal structure: the feedback loop. Representing interaction between a person and a

dynamic system as a simple feedback loop is a good first approximation, it forefronts the role of information looping through both the person and the system (Dubberly et al. 2009). Within dynamic systems, there is a distinction between those that only react, a linear or open-loop system, and those that interact, or closed-loop systems. Some closed-loop systems have a novel propriety—they can be self-regulating. But not all closed-loop systems are self-regulating. For example, the natural cycle of water is a loop. Rain falls from the atmosphere and is absorbed into the ground or runs into the sea. Water on the ground or in the sea evaporates into the atmosphere. But nowhere within the cycle is there a goal.

A self-regulating system has a goal. The goal defines a relationship between the system and its environment, which the system seeks to attain and maintain. This relationship is what the system regulates, what it seeks to keep constant in the face of external forces. A simple self-regulating system (one with only a single loop) cannot adjust its own goal; its goal can be adjusted only by something outside the system. Such single-loop systems are called “first order.”

Learning systems nest a first self-regulating system inside a second self-regulating system. The second system measures the effect of the first system on the environment and adjusts the first system’s goal according to how well its own second-order goal is being met. The second system sets the goal of the first, based on external action. We may call this learning—modification of goals based on the effect of actions. Learning systems are also called second-order systems.

Some learning systems nest multiple self-regulating systems at the first level. In pursuing its own goal, the second-order system may choose which first-order systems to activate. As the second-order system pursues its goal and tests options, it learns how its actions affect the environment.

A second-order system may in turn be nested within another self-regulating system. This process may continue for additional levels. For convenience, the term “second-order system” sometimes refers to any higher-order system, regardless of the number of levels, because from the perspective of the higher system, the lower systems are treated as if they were simply first-order systems.

3.1. Feedback Loops in the AURAL

When, in the AURAL, the user supplies parameters for fitness evaluation by drawing a curve (red) from which the coordinates of the points are taken as parameters, the curve – a linear system - provides input for a *learning* system, the evolutionary process. Medium solutions are expected in this case, since the fitness function changes quickly. The blue curve - output - supplies the *bio* parameter for

the reproduction cycle and the *rhy* parameter for the MIDI cycle (Figure 1). In the first case, the process is a *reinforcing* system. In this kind of system, the output of one self-regulating system is input for another. Reinforcing systems share similar goals. Redundancy is an important strategy in some cases. When the goals compete, we have *competing systems*.

The second case (the blue curve - output - supplies the *rhy* parameter for the MIDI cycle) is a *balancing* system: the output of one self-regulating system is input for another. Once the better individual of the population is selected (reproduction cycle) and placed in a critical area to be played (MIDI cycle) the process is a *conversing* system, when the output of a learning system becomes input for another.

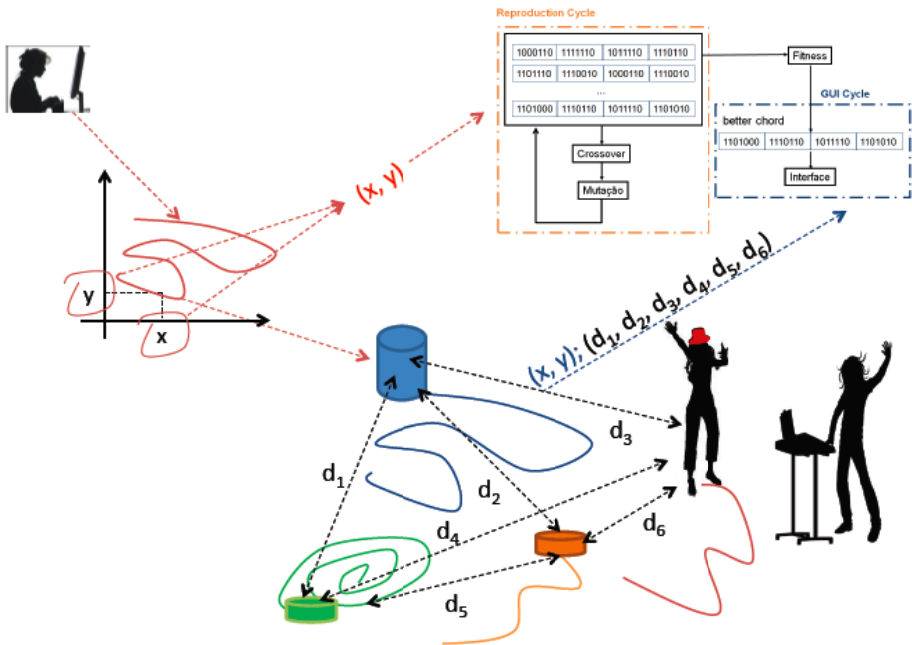


Figure 7. Flow of interaction in AURAL: on the left, above, the user/composer draws a curve and sends it as a trajectory to a master robot. The position of the robot (x, y) supplies parameters for the evolutionary process. The distance between the robots/dancer is evaluated by the vision system and performance modes are assigned and processed by the MIDI cycle. The musicians accompany the system in a performance.

Furthermore, depending on the distance between the pairs of robots, performance controls are activated (Figure 7). Moreover, the dancer using a red

hat is seen by the system as another robot, but with a different behavior. As an autonomous agent the dancer avoids collisions, while the robots handle collisions, they drive away from the obstacle.

When solo control is enabled, the sound events are sent directly to the MIDI board from the JaVOX evolutionary process, supplying a single sequence of MIDI events at every step of the genetic cycle. Therefore, the sound result depends only on the interaction between the red curve and the blue curve. In the second performance control (sequence), MIDI notes (voices) are played in quick event sequences. In this way the melodic character of the music is emphasized, generating a sound texture having a horizontal character. The third control (block) sends events to the MIDI board as quickly as possible, almost simultaneously, generating a superposition of notes or blocks. In this case the emphasis is in the verticality of the sound events, generating cluster textures.

The number of notes n cyclically sent to the MIDI board is described by a slider within the performance control of JaVOX. The notes are stored in a buffer memory containing the last n notes selected by the evolutionary cycle. This strategy introduces sound information of second order and brings about emergent and unexpected output, using data stored in the recent memory of the system.

Several cases occur in this interaction. Just to mention some of them, each agent, robot or dancer, can be considered a self-regulating system. The OmniEye observes their movement, evaluates the distance between the pairs of the agents and assigns the performance modes. As a system, the OmniEye measures the effect of the first system – the autonomous agents – and its output – the performance controls – becomes input for another system, the MIDI cycle. Again, in this case we have conversing systems. To achieve its goal, the MIDI cycle uses the notes stored in the buffer memory, which are the output of the evolutionary cycle, a learning system. The number of notes (n) is set by the user, characterizing a *regulating system*, where the output of one linear system provides the input to another.

3.2. Interactivity in AURAL₂

In the AURAL₂, sound fragments are triggered depending on the position of the robots on the stage. Like in the AURAL, each robot can be considered a self-regulating system. The vision system, videoGrid, tracks the movement of the robots on the stage, evaluates its position and assigns a sound fragment to be played. The sonifying process is activated, and may be considered a first order system. But, most important about AURAL₂, is that it is an *open system*. AURAL₂

is sensitive to both sounds of the environment and the interactions of the visitors at the microphone, storing sound fragments in the database when the result of the spectral analysis surpass a threshold. In this case, a rotation is applied on the grid which is presented on the TV monitor and the color of the cells change.

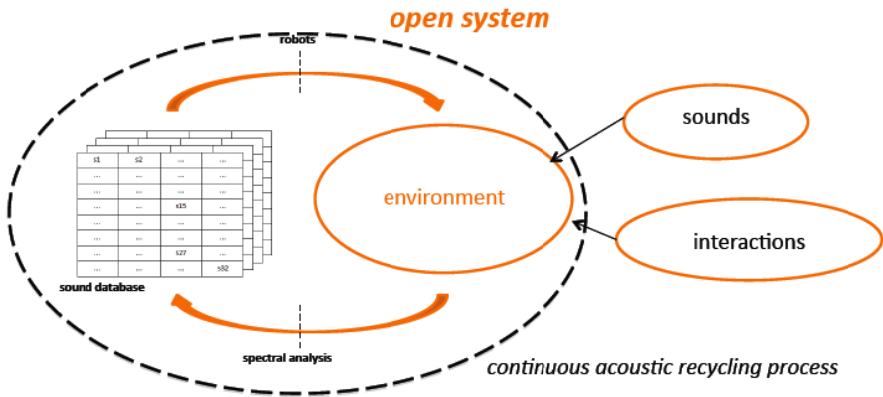


Figure 8. In AURAL₂, sound fragments are played depending on the position of the robots. User intervention at a microphone causes the storing of fragments in the sound database. These fragments may return modified to the environment and then return to the database.

The videoGrid acts as a regulating system. The sound fragments can be played again and, modified, they return to the sound database, in a continuous acoustic recycling process, once the cells where they are stored in the virtual grid are activated by the movement of the robots (Figure 8). In AURAL₂ the sonification is not the result of the complexity of the underlying system, but due to the complexity of the environment (Simon, 1981; Brooks, 1990).

4. Interactivity, Evolution and Structure

Several interactive processes were observed in the AURAL environments. In the interactive concert *Robotic Variations*, for example, the musicians played a music for which the movement of the robots on the arena was used as a composition strategy. A trajectory was sent to Nomad, the master robot, that tried to follow it, while other robots navigated in the arena in a pre-programmed autonomous mode. The same trajectory used to generate the material for the composition was used in the performance, but because of the evolutionary sonification process, even if the parameters of control are alike in every execution, the result is different in every run. The musicians knew the type of

music that would be generated, but they had to be able to adapt the performance. At the same time, the dancer, tracked by her red hat, was interacting with the robots, all interfering in the music that was being generated.

In each performance, the place of the robots, navigating in the arena in their autonomous mode, could be different; the dancer had to adapt herself to them. Important to remember is that all the process was triggered by a curve drawn by a human. The curve, as well as all the MIDI events generated in a test run, were recorded.

The sound material was adapted for human performance. The same curve was used in the rehearsals and in the final performance, with the musicians, the dancer and the robots. The dancer and the robots interfered in the sonification process, accomplished by the musicians, incurring in multiple feedback cycles.

Concerning to the structure/novelty tradeoff, the challenge faced by the designers of evolutionary composition systems is how to bring more structure and knowledge into the compositional loop, while trying to the user out of it. In this sense, JaVOX control interface offers some possibilities for the user. If simulated evolution techniques allow to obtain novelty, often complex novelty, the curves drawn in the interface permits to direct the novelty by guiding the evolutionary cycle through the desired regions of the conceptual sound space. The other controls: rhythm, pitch, performance and orchestra still allow to modify the sonority of the system in real time. The user/composer still is in the loop, but just directing the compositional loop.

On the other hand, in the AURAL₂ the structure is only suggested by the type of the sound samples (synthetic, game, environment and everyday sound fragments) stored in the database. The microphone acts as an invitation to the visitors for interaction. During the exhibition the people initially tried the installation by talking at the microphone. When the visitors heard segments of their speech mixed with other sounds, people started to explore the system by talking, singing, or even screaming, sometimes incurring in visual effects in the virtual grid displayed on the TV, by changing the color of the cells or the position of the grid. Filtered images of the robots and of the people were also displayed on the TV. When people become aware of the images, they started to move in front of the camera. The behavior of the people changed while they tried the environment.

More structure and knowledge built into the system means more reasonably structured musical output, but also more predictable output, which can be relaxed by introducing processes such as those linking the interaction of the robots with the performance controls. Less structure and knowledge in the system, like in AURAL₂, means more novel, unexpected output, but also more unstructured musical chaff. Producing computational models of such high-level behaviors,

embedded in robotic platforms, calls for novel research at the frontier between robotics, music and multimodal systems.

From the AURAL and AURAL₂ perspectives, humans and robots are agents of a complex system and the sonification is the emergent propriety that is produced by their interaction and behavior. As such the sonic result is not seen as an isolated aspect of these two systems but a representation of the synergetic capabilities of the agents to collaborate and produce a complex musical narrative. By interacting with the environment, which provides feedback via sensors, the systems generate different sonic structures. Evidences from situated robotics and neuroscience point to the fact that it is important to take into consideration the principles of parallelism, emergence, embodiment and feedback to foster the expressivity and creativity into machines (Verschure and Manzolli, 2013). Through the interaction between the real and then virtual worlds AURAL brings about emergent musical narratives critically dependent on their accomplishment as real-world artifacts, supplying a solution to be verified for the theory of an open and interactive system as the mind.

Conclusion

Through the interaction between the real and virtual worlds, AURAL produces emergent musical narratives, the so emanating composition depends on real world constraints. This kind of interactive system is based on the approach of the theory of mind including its creativity and aesthetics and it will be critically dependent on its accomplishments of real-world artifacts because only in this way may this theory of an open and interactive system be fully validated, since the computational power of a machine can be used to infer the implications of a program where the unassisted human mind is unable to do.

Researchers and developers remain optimistic about breaking down the barriers between humans and digital devices so that they can communicate more easily. Work that focuses attention on the physicality is especially important in an age when the interplay between real and virtual worlds is becoming so important. In time, sensing, computing and communication functions will become invisible and integrated into everyday objects and spaces. It remains unclear how far such developments can go. Meanwhile, artists and composers are enthusiastic about creating unprecedented interactive works that engage persons in innovative ways. The increasing presence of computing power in their artworks and compositions will continue to bring up questions about the extent to which artists might be involved in designing the intelligence, form and interactivity of works in the future.

Acknowledgments

We wish to thank the students Thiago Spina, Eddy Nakamura, Felipe Augusto, Helen Fornazier and Gustavo Solaira, who worked with Pioneer, Nomad, Roomba and iCreate robots. We also thank the students Lucas Soares, Igor Dias, Igor Martins and Flavio Kodama who worked in the development of OmniEye and JaVOX, and Marcelo Salvatori, Rafael Guimarães Ramos, Felipe Shida and Daniel Kantor, who worked in AURAL₂. We wish to thank Mariana Shellard for the video production. We thank the musicians Cesar Traldi, Chiquinho Costa, Adriano Monteiro and the dancer Tatiana Benone, who played in the performance at Unicamp/IA Art Gallery. We thank Adriana Giarola Kayama for her narration of AURAL videos. We thank the researchers Josué Ramos, Sidney Cunha and Hélio Azevedo for their valuable contributions. We also thank to the technical support of Douglas Figueiredo. We thank the Scientific Initiation Program of the National Research Council (PIBIC/CNPq), the Center for Technology Information Renato Archer and the Interdisciplinary Nucleus for Sound Studies of the State University of Campinas (NICS/UNICAMP) for making this research possible. This research work was part of the AURAL project, supported by the Foundation for the Research in São Paulo State (FAPESP) process 05/56186-9. Manzolli is supported by the Brazilian Agency CNPq.

References

- Ashby, W. R., 1956. An Introduction to Cybernetics, *Chapman & Hall*, London.
- Boden, M., 1991. Computer models of mind, Cambridge University Press.
- Brooks, R., 1990. Elephants don't play chess. *Robotics and autonomous systems* 6 (1-2), 3–15.
- Candy, L., Edmonds, E. Interaction in Art and Technology.
<http://crossings.tcd.ie/issues/2.1/Candy/>
- Cornock, S. and Edmonds, E., 1973. The Creative Process where the Artist is Amplified or Superseded by the Computer. *Leonardo* 6, 11–16.
- Dubberly, H., Haue, U., Pangaro, P., 2009. What is interaction? Are there different types? <http://www.dubberly.com/articles/what-is-interaction.html>
- Jewell, E., Abate, F., McKean, E., 2001. The New Oxford American Dictionary. Oxford University Press.
- Le Groux, S., 2011. Situated, perceptual, emotive and cognitive music systems: a psychologically grounded approach to interactive music composition, Dr. Thesis, Vershure, P. (advisor), Universitat Pompeu Fabra.

- Manzolli, J., 2011. pd descriptor: Spectral analysis on sound fragment. Internal Report, NICS/Unicamp.
- Moroni, A. 2012. AURAL: A Robotic Evolutionary Environment for Sound Production. <https://sites.google.com/site/auralroboticsonification/>
- Moroni, A., Manzolli, J., 2010. From Evolutionary Composition to Robotic Sonification. In: *EvoApplications 2010*, Istanbul. Applications of Evolutionary Computation. Berlin: Springer.
- Moroni, A., Manzolli, J., Von Zuben, F. J., Gudwin, R., 2000. VoxPopuli: An Interactive Evolutionary System for Algorithmic Music Composition. *Leonardo Music Journal* 10, pp. 49–54.
- Moroni, A., Manzolli, J., Von Zuben, F. J., Gudwin, R. 2002. Evolutionary Computation for Music Evolution. In: Bentley, P., Corne, D. *Creative Evolutionary Systems*, Morgan Kaufmann, San Francisco, pp. 205–221.
- Nierhaus, G., 2009. *Algorithmic composition: paradigms of automated music generation*. Springer Verlag Wien.
- Papadopoulos, G., Wiggins, G., 1999. AI methods for algorithmic composition: A survey, a critical view and future prospects. In: *AISB Symposium on Musical Creativity*, 124, 110–117.
- Puckette, M., 1996. Pure data: another integrated computer music environment. In: *Proceedings of the 2nd Intercollege Computer Music Concerts*. Tachikawa, Japan. 236.
- Rowe, R., 1993. *Interactive music systems: machine listening and composing*. MIT Press, Cambridge, MA, USA.
- Simon, H., 1981. *The science of the artificial*. Cambridge, USA, MA.
- Todd, P. M., Werner, G. M., 1999. Frankensteinian Methods for Evolutionary Music Composition. In: Griffith, N. & Todd, P. M. (eds.) *Musical Networks: Parallel Distributed Perception and Performance*, 313–340, Cambridge: The MIT Press.
- Verschure, P. F. M. J., Manzolli, J., 2013. Computational Modeling of Mind and Music. In *Language, Music, and the Brain. Strüngmann Forum Reports*, MIT Press, Vol. 10.
- Vidyamurthy, G. and Chakrapani, J., 1992. *Cognition of Tonal Centres: A Fuzzy Approach*, *Computer Music Journal*, 16:2.
- Wassermann, K. C., Eng, K., Verschure, P. F. M. J., Manzolli, J., 2003. Live Soundscape Composition Based on Synthetic Emotions. *IEEE Multimedia*, 82–90.

- Winkler, T., 2001. *Composing interactive music: techniques and ideas using Max*. The MIT Press.
- Wright, M., 2005. Open sound control: an enabling technology for musical networking. *Org. Sound* 10 (3), 193–200.
- Zicarelli, D., 2002. How I learned to love a program that does nothing. *Computer Music Journal* (26), 44–51.

Chapter 6

AN ANALYSIS OF EVOLUTIONARY-BASED SAMPLING METHODOLOGIES

*Yoel Tenne**

Department of Mechanical and Mechatronic Engineering,
Ariel University, Israel

Abstract

A common approach for solving simulation-driven engineering problems is by using metamodel-assisted optimization algorithms, namely, in which a metamodel approximates the computationally expensive simulation and provides predicted values at a lower computational cost. Such algorithms typically generate an initial sample of solutions which are then used to train a preliminary metamodel and to initiate optimization process. One approach for generating the initial sample is with the design of experiment methods which are statistically oriented, while the more recent search-driven sampling approach invokes a computational intelligence optimizer such as an evolutionary algorithm, and then uses the vectors it generated as the initial sample. Since the initial sample can strongly impact the effectiveness of the optimization process, this study presents an extensive comparison and analysis between the two approaches across a variety of settings. Results show that evolutionary-based sampling performed well when the size of the initial sample was large as this enabled a more extended and consequently a more effective evolutionary search.

*E-mail address: y.tenne@ariel.ac.il

When the initial sample was small the design of experiments methods typically performed better since they distributed the vectors more effectively in the search space.

PACS: 05.45-a, 52.35.Mw, 96.50.Fm

Keywords: evolutionary algorithms, sampling methods, expensive optimization problems, metamodelling

1. Introduction

These days computer simulations are used in engineering as a substitute for real-world experiments, with the goal of making the design process more efficient. These simulations, which must be properly validated with laboratory experiments, transform the design process into an optimization problem having three distinct features (Tenne and Goh, 2010):

- The simulation acts as the objective function since it assigns candidate designs their corresponding objective values. However, the simulation is often a legacy code or a commercial software whose inner workings are inaccessible to the user, and is therefore viewed as a *black-box function*, namely, which lacks an analytic expression. This precludes the of optimization methods which require an analytic function.
- Each simulation run requires extensive computer resources, and this severely restricts the overall number of candidate designs which can be evaluated during the design process.
- Both the real-world physics being modelled, and the numerical simulation process, may result in an objective function having a complicated nonconvex landscape which exacerbates the optimization difficulty.

These scenarios are commonly referred to in the literature as *expensive black-box optimization problems*, and a variety of algorithms have been proposed to address them (Tenne and Goh, 2010; Forrester and Keane, 2008; Queipo et al., 2005).

A common methodology which is applied to such scenarios is that of metamodel-assisted optimization, namely, in which a metamodel approximates the true expensive function (the simulation), and provides the optimizer with

predicted objective values at a lower computational cost. Such algorithms typically begin by generating an initial sample of vectors which represent candidate solutions, and these are then used to train a preliminary metamodel and to initiate the main optimization search. This implies that the initial sample can strongly impact the overall search effectiveness and motivates a closer analysis of this relation.

A common approach for generating the initial sample is by the design of experiments (DOE) methods, which are statistically-oriented and aim to generate a sample which is optimal in some sense. A more recent approach is that of search-based sampling (SBS) in which a direct search optimizer is invoked for a short duration and the vectors it evaluated then serve as the initial sample. In the literature, existing studies in the domain of metamodel-assisted optimization have focused mainly on DOE methods, for example, as discussed by Queipo et al. (2005); Chen et al. (2006); Sóbester et al. (2005); Forrester and Keane (2008); Wang and Shan (2007). Therefore, the focus and main contribution of this study is to: a) compare the DOE and the search-driven sampling (SDS) approaches in metamodel-assisted optimization, and b) analyse the impact of these sampling methods on the overall search effectiveness instead of the metamodel accuracy. An extensive set of numerical experiments is used to formulate guidelines as to how best apply such sampling methods.

The remainder of this chapter is organized as follows: Section 2 provides pertinent background information, Section 3 describes in detail the numerical experiments performed, and Section 4 provides an extensive performance analysis. Lastly, Section 5 concludes this chapter.

2. Background

This section provides background information on relevant optimization and sampling methods.

2.1. Expensive Optimization Problems and Computational Intelligence Algorithms

Computationally-expensive simulation-driven optimization problems are common across engineering and science domains, and Figure 1 shows their layout, where the simulation is treated as a black-box function. In this setup, candidate

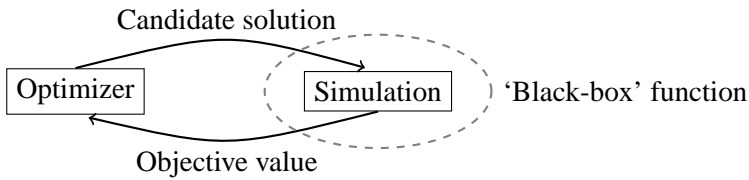


Figure 1. The layout of a computationally expensive simulation-driven optimization problem.

designs are parameterized as vectors of design variables and are provided as inputs to the simulation.

Meta-models (also termed *response surface* or a *surrogate model*) are often used to circumvent the high evaluation cost. They provide predicted objective values at a much lower computational cost when compared to the simulation code. Metamodels are typically interpolants which have been trained by using evaluated vectors, and examples include artificial neural networks, Kriging, polynomials, and radial basis functions (RBF) (Forrester and Keane, 2008; Queipo et al., 2005). Metamodel-assisted algorithm typically begin by sampling a set of vectors and using them to train a preliminary metamodel. An optimization search is then performed in which the optimum of the metamodel is sought, and vectors generated during this search are evaluated with the true expensive function and are used to update the metamodel. This process then repeats until a maximum number of analysis have been performed. Algorithm 1 gives a pseudocode of a baseline metamodel-assisted optimization algorithm.

After training a metamodel, two main classes of optimization algorithms can be used to search for an optimizer:

- Gradient-based optimizers: These optimizers typically operate on a single candidate solution which is refined at each iteration based on the local gradient. The resultant search is typically localized.
- Direct search optimizers: Such optimizers typically employ a number of candidate solutions concurrently, and manipulate them based only on the observed function values (gradients are not utilized). The resultant search tends to be more explorative when compared to gradient-based methods.

Algorithm 1: A baseline metamodel-assisted optimization algorithm

```
/* Initial sampling step                                     */
Generate an initial sample of vectors;
Evaluate the vectors with the true expensive function and store them in
memory;
/* Main optimization loop                                   */
while maximum number of analyses not reached do
    Train a metamodel by using the vectors stored in memory;
    Search for an optimum of the metamodel, based on an infill sampling
    criterion;
    Evaluate with the true expensive function one or more of the vectors
    generated during the search, and add them to the memory storage;
```

Since black-box functions often have a complicated nonconvex landscape, gradient-based optimizers can converge to a poor local optimum. This has motivated the use of direct-search optimizers in such problems since their explorative search behaviour often allows to them to locate a better final solution (Zahara and Kao, 2009; Babu and Rakesh, 2006). One such optimizer, which has been widely used in literature and which is also employed in this study, is the *evolutionary algorithm* (EA), which uses the following nature-inspired operators (de Jong, 2006): i) *selection*: the vectors (typically those with a better objective value) are selected as *parents*, ii) *recombination*: the parents vectors are combined to yield *offspring*, iii) *mutation*: some offspring vectors are perturbed. The offspring population is then evaluated, and the members with the best objective values serve as the next generation population. The process then repeats until a termination criterion is met, for example, if the maximum number of generations has been reached. Through these nature-inspired operators the EA population explores the function landscape and is able to locate good solutions even in challenging scenarios such as with nonconvex or nondifferentiable functions. Algorithm 5 gives a pseudocode of a baseline EA.

2.2. Sampling Methods

Sampling methods were originally aimed to assist in designing real-world laboratory experiments, and hence denoted as *design of experiments* (DOE) methods. The main assumptions under which they were developed were that the true

Algorithm 2: A baseline evolutionary algorithm (EA)

```

Initialize and evaluate a population of candidate solutions;
/* main loop                                     */
repeat
  Select a group of candidate solutions and designate them as parents;
  Recombine the parents to create offspring;
  Mutate some of the offspring;
  Evaluate the offspring;
  Select the candidate solutions which will comprise the population of
  the next generation;
until convergence or maximum search duration;

```

objective function is a low-order polynomial, and that the objective values being observed in the experiment always contain some random noise. In these settings the classical DOE methods were developed to minimize the noise impact and to improve the accuracy of estimating the polynomial coefficients. Examples of such methods include factorial designs (Fisher, 1926; Finney, 1945), central composite designs (Box and Wilson, 1951), and the response surface methodology (RSM) designs (Myers and Montgomery, 1995).

The above assumptions were invalid for computer experiments since now the observed function values were deterministic, namely, free of noise. Also, the true objective function was no longer restricted to be a low order polynomial, and more general approximations were being considered such as RBF, Kriging, and artificial neural networks (ANN). These issues have driven the development of modern DOE methods whose goal is to cover the search space in some optimal way, which consequently should improve the prediction accuracy of the resultant metamodel. Examples of such methods include Monte Carlo sampling (Metropolis and Ulam, 1949), Latin hypercube (LH) designs (McKay et al., 1979), orthogonal arrays (Owen, 1992), and maximin and minimax designs (Johnson et al., 1990), to name a few.

A recent and drastically different approach is that of search-driven sampling (SDS). Here an optimizer is invoked for a short duration and during its run it calls the objective function directly (no metamodels are involved in this phase). After this short run has been completed, the vectors evaluated then serve as the initial sample. Since the gradient of the black-box function is not known,

SDS setups have used a *direct search* optimizer, namely, which relies on the observed function values only (Jin et al., 2005; Quttineh and Holmström, 2009). Such optimizers typically operate on a set of vectors instead of a single one, and manipulate them based on the observed function values. In particular, EAs, which are known for their robustness and effectiveness, have been used as the direct search in the SDS approach, and examples include Büche et al. (2005) and Liang et al. (2000). To conclude this section, Table 1 compares the main aspects of the DOE and SDS approaches.

Table 1. Features of DOE and SDS sampling methods

	DOE	SDS
Procedure	Sampling vectors from a statistical distribution	Invoking an optimizer which directly evaluates the expensive function, and using the evaluated vectors as the initial sample
Sequence of sample generation	Entire sample a-priori	Incrementally during the search
Parallel evaluation	Entire sample a-priori	Only the set of vectors available at each iteration
Objective function affects sampling	No	Yes
Sample is space-filling	Yes	Partially
A metamodel is involved in the procedure	No	No ^[1]

^[1] : As an exception, Laurenceau and Sagaut (2008); Quttineh and Holmström (2009) used a Kriging metamodel.

3. Numerical Experiments: Design and Implementation

This section describes the numerical experiments which were used to evaluate the impact that different methods for generating the initial sample have on the search effectiveness.

3.1. Design of the Numerical Experiments

Each experiment consisted of a pre-optimization stage in which the initial sample was generated by one sampling method, followed by a full optimization search which was performed by a representative optimizer (which is described in Section 3.4).

The experiments were formulated based on the *designed experiments* framework of Myers and Montgomery (1995) such that they included two components:

- *Factors*: The main variables whose effect is being analyzed. In this study the sampling method type was defined as the factor. The different methods are described in Section 3.2.
- *Control variables*: These are settings of the experiments which are kept fixed while across different factor values. In this study, four control variables were defined: a) the size of the initial sample, b) the optimization budget, c) the function type, and d) the function dimension, which are described in Section 3.3. Also following the designed experiments framework, a 2^n factorial design of experiments was employed in which each control variable (except for *function type*) was given representative “low” and “high” settings, and which resulted in eight settings combinations, as described in Table 2. Together with the six test functions employed, this resulted in 48 different *optimization scenarios*, namely, unique combinations of control variables settings.

Each sampling method was employed in all 48 optimization scenarios, and to support a valid statistical analysis in each scenario 30 runs were repeated with each sampling method. In summary, the layout of the numerical experiments was:

Step 1) Generate an initial sample by either:

- Latin hypercube sampling (LHS).
- Monte Carlo sampling.
- EA-based search-driven sampling.

No metamodel was used in this step.

Step 2) Perform a full optimization search with a metamodel-assisted algorithm (the same optimization algorithm was used in all tests).

Table 2. Formulation of the numerical experiments

Component type	Component name	Assigned settings
Factor	Sampling method	LHS, Monte Carlo sampling, micro-EA SDS
Control variable	Optimization budget	200, 2000 evaluations of the true function
	Size of initial sample	10%, 25% of the optimization budget
	Function type	Ackley, Griewank, Rastrigin, Rosenbrock, Schwefel 2.13, Weierstrass
	Function dimension	10, 50

This setup results in 48 optimization scenarios: 2 sizes of the initial sample \times 2 optimization budgets \times 6 objective functions \times 2 function dimensions.

3.2. Sampling Methods

Three methods for generating the initial sample were employed, as follows:

- Latin hypercube (LH) sampling: A DOE method which ensures that the resultant sample is space-filling *and* covers the full range of the design variables (McKay et al., 1979). Briefly, for a sample of k vectors the range of each variable is split into k equal intervals, and one point is sampled at random in each interval. Next, a sample point is selected at random and without replacement for each variable, and these samples are combined to produce a vector. This procedure is repeated for k times to generate the complete sample. The method has been widely employed in literature, in problems ranging from the allocation of water resources (Mugunthan and Shoemaker, 2006) to the design of electronic circuits (You et al., 2009).
- Monte Carlo sampling: A DOE method, also termed *random sampling*, which generates each sample vector individually and without considering previously sampled vectors, which can result in some vectors being adjacent. Vectors are typically drawn from the uniform multivariate distribution. The method has been widely employed in literature, in problems ranging from the design of a satellite boom (El-Betalgy and Keane, 2001) to the design of an electric motor (Neri et al., 2008).
- Micro-EA SDS: In this approach an EA is invoked for a short duration,

and the vectors it evaluated then serve as the initial sample. The small size of the initial sample in expensive optimization problems implies that the population must be small, as otherwise the allotted function evaluations would rapidly be exhausted and the EA would terminate prematurely. Therefore, studies have used what is termed in the literature as a *micro-EA* (Krishnakumar, 1989), namely, an EA with a small population. For example, Liang et al. (2000) used a ‘(1+1) EA’ in which a single parent generated a single offspring and the better out of the two progressed to the next generation. As another example, Büche et al. (2005) used a covariance matrix adaptation evolutionary strategy (CMA-ES) optimizer in which 2 parents generated 10 offspring.

Following the literature, the numerical experiments included a micro-EA which employed a population of five members (Senecal, 2000). The micro-EA operates as described in Algorithm 5, and Table 5 gives its internal parameter settings.

Table 3. Internal parameters of the micro-EA used for generating the initial sample

Population size	5
Selection	Stochastic universal selection (SUS)
Recombination	Intermediate, $p = 0.7$
Mutation	Breeder Genetic Algorithm (BGA) mutation (Chipperfield et al., 1994), $p = 0.1$
Elitism	10%

p : The probability of applying the operator

3.3. Optimization Scenarios

As described in Section 3.1, 48 different optimization scenarios were used, which correspond to different combinations of control variables settings, as follows:

- i) Optimization budget: Affects the duration of the optimization search and the size of the initial sample. The experiments included a low setting of 200 function evaluations and a high setting of 2000 function evaluations.

- ii) *Relative size of the initial sample*: Affects the trade-off between the size of the initial sample and the duration of the main optimization search which succeeds it. The experiments included a low setting of 10% of the optimization budget and a high setting of 25%.
- iii) *Objective function type*: Affects the optimization difficulty due to the function features (nonconvexity, nondifferentiability). The experiments included the test functions set (Suganthan et al., 2005): Ackley, Griewank, Rastrigin, Rosenbrock, Schwefel 2.13, and Weierstrass, whose details are given in Table 4. A bias term (Suganthan et al., 2005) was not used in this study.
- iv) *Function dimension*: Affects the optimization difficulty ('curse of dimensionality'). The experiments included a low setting of dimension 10 and a high setting of dimension 50.

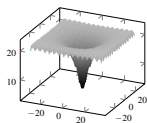
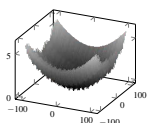
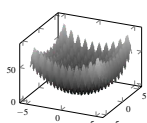
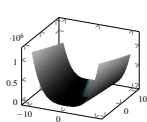
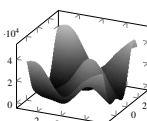
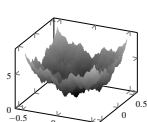
3.4. Optimization Algorithm

As mentioned in Section 3.1, in each numerical experiment after generating the initial sample an optimization search was performed. The latter achieved by a metamodel-assisted EA based on the representative algorithm of Ratle (1999).

In this algorithm the vectors evaluated so far are used to train a Kriging metamodel, which is described later in this section. The real-coded EA of Chipperfield et al. (1994) is then invoked to search for an optimum of the metamodel, where the EA is run for 10 generations. The EA follows the description in Section 3.2 and its internal parameter settings are given in Table 5. To further improve the effectiveness of the EA search, during the optimization search it employed a large population size. After the EA search has been completed, the ten best population members are evaluated with the true objective function and are added to the memory storage. Another iteration is then performed, until the number of analyses, namely, calls to the true objective function, reaches the prescribed limit. To complete the description, Algorithm 3 gives the pseudocode of the algorithm.

In this study the Kriging metamodel was employed (Forrester and Keane, 2008; Queipo et al., 2005). It takes a statistical approach to interpolation by combining two components: a 'drift' function, which is a global coarse approximation of the true function, and a local correction based on the correlation between the interpolation vectors. Given a set of evaluated vectors, $\mathbf{x}_i \in \mathbb{R}^d$,

Table 4. Test functions

Function	Definition, $f(\mathbf{x}) =$	Domain	Plot of bivariate case
Ackley	$-20 \exp\left(-0.2 \sqrt{\sum_{i=1}^d x_i^2 / d}\right) - \exp\left(\sum_{i=1}^d \cos(2\pi x_i) / d\right) + 20 + e$	$[-32, 32]^d$	
Griewank	$\sum_{i=1}^d \{x_i^2 / 4000\} - \prod_{i=1}^d \{\cos(x_i / \sqrt{i})\} + 1$	$[-100, 100]^d$	
Rastrigin	$\sum_{i=1}^d \{x_i^2 - 10 \cos(2\pi x_i) + 10\}$	$[-5, 5]^d$	
Rosenbrock	$\sum_{i=1}^{d-1} \{100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\}$	$[-10, 10]^d$	
Schwefel 2.13	$\sum_{i=1}^d \left\{ \sum_{j=1}^d \left[(a_{i,j} \sin(\alpha_j) + b_{i,j} \cos(\alpha_j)) - (a_{i,j} \sin(x_j) + b_{i,j} \cos(x_j)) \right]^2 \right\}$	$[-\pi, \pi]^d$	
Weierstrass	$\sum_{i=1}^d \left\{ \sum_{k=0}^{20} 0.5^k \cos(2\pi 3^k (x_i + 0.5)) \right\} - d \sum_{k=0}^{20} 0.5^k \cos(\pi 3^k)$	$[-0.5, 0.5]^d$	

Algorithm 3: The optimization algorithm used in the main search

while *maximum number of analyses not reached* **do**
 Train a metamodel with the vectors which have been evaluated with the true function;
 Search for an optimum of the metamodel by using a real-coded EA;
 Evaluate with the true function the ten best vectors from the resultant EA population;
Return the best solution found;

Table 5. Internal parameters of the EA used in the main search

Population size	100
Generations	10
Selection	Stochastic universal selection (SUS)
Recombination operator	Intermediate, $p = 0.7$
Mutation operator	Breeder Genetic Algorithm (BGA) mutation (Chipperfield et al., 1994), $p = 0.1$
Elitism	10%

p : The probability of applying the operator

$i = 1 \dots n$, the Kriging metamodel is trained such that it exactly interpolates the observed values, that is, $m(\mathbf{x}_i) = f(\mathbf{x}_i)$, where $m(\mathbf{x})$ and $f(\mathbf{x})$ are the metamodel and true objective function, respectively. Using a constant drift function gives the Kriging metamodel

$$m(\mathbf{x}) = \beta + \kappa(\mathbf{x}), \quad (1)$$

with the drift function β and local correction $\kappa(\mathbf{x})$. The latter is defined by a stationary Gaussian process with mean zero and covariance

$$Cov[\kappa(\mathbf{x})\kappa(\mathbf{y})] = \sigma^2 c(\theta, \mathbf{x}, \mathbf{y}), \quad (2)$$

where $c(\theta, \mathbf{x}, \mathbf{y})$ is a user-prescribed correlation function. A common choice for the latter is the Gaussian correlation function (Forrester and Keane, 2008), defined as

$$c(\theta, \mathbf{x}, \mathbf{y}) = \prod_{i=1}^d \exp(-\theta (x_i - y_i)^2), \quad (3)$$

and combining it with the constant drift function transforms the metamodel from

(1) into the following form

$$m(\mathbf{x}) = \hat{\boldsymbol{\beta}} + \mathbf{r}(\mathbf{x})^T \mathbf{R}^{-1} (\mathbf{f} - \mathbf{1} \hat{\boldsymbol{\beta}}). \quad (4)$$

Here, $\hat{\boldsymbol{\beta}}$ is the estimated drift coefficient, \mathbf{R} is the symmetric matrix of correlations between all interpolation vectors, \mathbf{f} is the vector of objective values, and $\mathbf{1}$ is a vector with all elements equal to 1. \mathbf{r}^T is the correlation vector between a new vector \mathbf{x} and the sample vectors, namely,

$$\mathbf{r}^T = [c(\boldsymbol{\theta}, \mathbf{x}, \mathbf{x}_1), \dots, c(\boldsymbol{\theta}, \mathbf{x}, \mathbf{x}_n)]. \quad (5)$$

The estimated drift coefficient $\hat{\boldsymbol{\beta}}$ and variance $\hat{\sigma}^2$, which are required in Equation 4, are obtained as follows

$$\hat{\boldsymbol{\beta}} = (\mathbf{1}^T \mathbf{R}^{-1} \mathbf{1})^{-1} \mathbf{1}^T \mathbf{R}^{-1} \mathbf{f}, \quad (6a)$$

$$\hat{\sigma}^2 = \frac{1}{n} \left[(\mathbf{f} - \mathbf{1} \hat{\boldsymbol{\beta}})^T \mathbf{R}^{-1} (\mathbf{f} - \mathbf{1} \hat{\boldsymbol{\beta}}) \right]. \quad (6b)$$

Fully defining the metamodel requires the correlation parameters $\boldsymbol{\theta}$, which are commonly taken as the maximizers of the metamodel likelihood. This is achieved by minimizing the expression (Sacks et al., 1989)

$$\psi(\boldsymbol{\theta}) = |\mathbf{R}|^{1/n} \hat{\sigma}^2 \quad (7)$$

which is a function only of the correlation parameters $\boldsymbol{\theta}$ and the sample data. In this study a single correlation parameter was used, as is commonly done in the literature to simplify the parameter tuning (Martin and Simpson, 2005).

To demonstrate the effectiveness of the metamodel-assisted algorithm employed, it was applied to the six test functions mentioned in Section 3.1, with a budget of 2000 function evaluations and a LHS initial sample with a size of 200 vectors. Ten repetitions were performed per function. Table 6 gives the resultant test statistics from which it follows that the algorithm typically identified a good final solution, given the limited optimization budget and the highly nonconvex objective functions involved. The final solutions obtained with the Rosenbrock and Schwefel 2.13 were typically higher, but in these objective functions the function value increases sharply when moving away from the global optimum, and therefore even final solutions which were adjacent to the true global optimum corresponded to a high objective values. To further demonstrate the behaviour of the algorithm employed, Figure 2 gives three representative plots,

each corresponding to a single run with the Ackley (10D), Rastrigin (50D), and Schwefel 3.12 (50D) functions, respectively. The plots show the distance (l_2 norm) between the best solution found to the true global optimum. It follows that in all cases the algorithm approached the true global optimum, which indicates that it effectively coped with the issue of metamodel inaccuracy. Overall, the results above show that the algorithm employed was suitable to be used an optimizer in the numerical experiments.

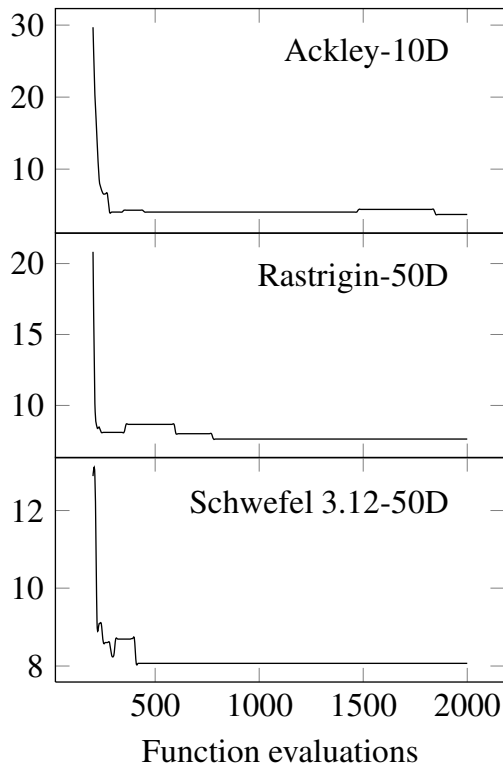


Figure 2. The distance between the best solution found during the search and the true global optimization. Each plot corresponds to one test run from used to generate Table 6.

Table 6. Test statistics for the metamodel-assisted framework employed

		10D	50D
Ackley	Mean	4.091e+00	8.575e+00
	SD	2.118e-01	2.363e-01
	Median	4.109e+00	8.510e+00
	Min(best)	3.701e+00	8.307e+00
	Max(worst)	4.345e+00	8.914e+00
Griewank	Mean	7.491e-01	1.551e+00
	SD	1.049e-01	4.434e-02
	Median	8.087e-01	1.552e+00
	Min(best)	5.641e-01	1.485e+00
	Max(worst)	8.360e-01	1.600e+00
Rastrigin	Mean	2.378e+01	3.461e+02
	SD	4.380e+00	1.590e+01
	Median	2.408e+01	3.432e+02
	Min(best)	1.827e+01	3.289e+02
	Max(worst)	2.932e+01	3.682e+02
Rosenbrock	Mean	3.560e+01	4.439e+03
	SD	6.746e+00	6.635e+02
	Median	3.618e+01	4.259e+03
	Min(best)	2.532e+01	3.805e+03
	Max(worst)	4.734e+01	5.566e+03
Schwefel 2.13	Mean	2.450e+03	3.118e+06
	SD	9.369e+02	3.254e+05
	Median	2.594e+03	3.146e+06
	Min(best)	1.176e+03	2.491e+06
	Max(worst)	4.061e+03	3.565e+06
Weierstrass	Mean	2.648e+00	2.774e+01
	SD	3.352e-01	7.148e-01
	Median	2.729e+00	2.799e+01
	Min(best)	1.986e+00	2.654e+01
	Max(worst)	2.952e+00	2.861e+01

In all cases, at the global optimum the value of the true objective function is 0.

Table 7. Test statistics for the numerical experiments

	Optimization budget=200 evaluations						Optimization budget=2000 evaluations					
	Sample size=10%			Sample size=25%			Sample size=10%			Sample size=25%		
	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS
Ackley-10D												
Mean	5.071e+00	5.036e+00	5.740e+00	4.911e+00	5.034e+00	6.751e+00	4.014e+00	3.825e+00	3.902e+00	3.903e+00	3.959e+00	3.932e+00
SD	7.940e-01	6.907e-01	1.170e+00	7.868e-01	7.415e-01	2.039e+00	2.819e-01	3.572e-01	3.689e-01	3.501e-01	3.579e-01	2.613e-01
Median	4.999e+00	5.059e+00	5.389e+00	4.852e+00	4.979e+00	6.357e+00	4.050e+00	3.806e+00	4.024e+00	3.907e+00	4.006e+00	3.961e+00
Min(best)	3.714e+00	3.716e+00	3.534e+00	3.545e+00	4.117e+00	4.272e+00	3.486e+00	3.095e+00	2.745e+00	2.941e+00	2.444e+00	3.265e+00
Max(worst)	7.396e+00	6.816e+00	9.866e+00	6.739e+00	7.584e+00	1.343e+01	4.585e+00	4.499e+00	4.429e+00	4.482e+00	4.456e+00	4.343e+00
Ackley-50D												
Mean	9.439e+00	9.313e+00	9.529e+00	9.281e+00	9.402e+00	9.610e+00	8.629e+00	8.624e+00	8.681e+00	8.743e+00	8.624e+00	8.711e+00
SD	3.072e-01	2.714e-01	3.778e-01	3.432e-01	3.508e-01	6.233e-01	2.709e-01	2.434e-01	2.278e-01	2.278e-01	2.816e-01	2.363e-01
Median	9.466e+00	9.358e+00	9.484e+00	9.268e+00	9.292e+00	9.616e+00	8.625e+00	8.618e+00	8.698e+00	8.765e+00	8.699e+00	8.746e+00
Min(best)	8.873e+00	8.887e+00	8.432e+00	8.647e+00	8.740e+00	8.174e+00	7.854e+00	8.152e+00	8.138e+00	8.150e+00	7.793e+00	8.029e+00
Max(worst)	1.007e+01	9.766e+00	1.024e+01	9.934e+00	1.002e+01	1.070e+01	9.042e+00	9.161e+00	8.971e+00	9.057e+00	9.045e+00	9.129e+00
Griewank-10D												
Mean	9.593e-01	9.329e-01	9.550e-01	9.472e-01	9.287e-01	1.015e+00	7.309e-01	7.057e-01	7.323e-01	7.305e-01	7.722e-01	7.455e-01
SD	7.098e-02	8.261e-02	1.715e-01	5.681e-02	8.293e-02	9.314e-02	9.894e-02	1.219e-01	1.065e-01	8.088e-02	9.177e-02	8.081e-02
Median	9.739e-01	9.454e-01	9.731e-01	9.656e-01	9.613e-01	1.015e+00	7.580e-01	7.287e-01	7.484e-01	7.233e-01	7.935e-01	7.602e-01
Min(best)	7.738e-01	7.564e-01	4.846e-01	7.892e-01	7.035e-01	7.562e-01	4.826e-01	3.167e-01	4.284e-01	5.822e-01	5.517e-01	5.777e-01
Max(worst)	1.105e+00	1.039e+00	1.643e+00	1.019e+00	1.032e+00	1.324e+00	8.903e-01	8.910e-01	8.946e-01	8.845e-01	9.155e-01	9.111e-01
Griewank-50D												
Mean	1.707e+00	1.701e+00	1.768e+00	1.670e+00	1.728e+00	1.811e+00	1.577e+00	1.569e+00	1.562e+00	1.568e+00	1.573e+00	1.554e+00
SD	7.716e-02	9.085e-02	1.231e-01	8.741e-02	7.512e-02	1.357e-01	4.306e-02	4.219e-02	5.766e-02	5.007e-02	4.900e-02	6.140e-02
Median	1.696e+00	1.721e+00	1.774e+00	1.657e+00	1.719e+00	1.816e+00	1.579e+00	1.567e+00	1.567e+00	1.561e+00	1.565e+00	1.559e+00
Min(best)	1.531e+00	1.543e+00	1.517e+00	1.507e+00	1.612e+00	1.561e+00	1.493e+00	1.494e+00	1.433e+00	1.482e+00	1.474e+00	1.400e+00
Max(worst)	1.877e+00	1.898e+00	2.123e+00	1.892e+00	1.903e+00	2.142e+00	1.670e+00	1.691e+00	1.687e+00	1.680e+00	1.695e+00	1.644e+00
Rastrigin-10D												
Mean	4.362e+01	4.392e+01	4.446e+01	4.633e+01	4.413e+01	4.768e+01	2.261e+01	2.262e+01	2.362e+01	2.441e+01	2.436e+01	2.345e+01
SD	7.226e+00	8.564e+00	8.538e+00	8.702e+00	7.127e+00	1.018e+01	3.920e+00	5.430e+00	6.220e+00	5.694e+00	4.737e+00	3.921e+00
Median	4.410e+01	4.598e+01	4.319e+01	4.648e+01	4.375e+01	4.680e+01	2.254e+01	2.429e+01	2.484e+01	2.571e+01	2.499e+01	2.323e+01
Min(best)	2.347e+01	2.650e+01	2.693e+01	2.407e+01	2.527e+01	3.174e+01	1.367e+01	1.258e+01	7.150e+00	1.293e+01	1.525e+01	1.432e+01
Max(worst)	5.758e+01	6.095e+01	6.050e+01	6.333e+01	5.490e+01	7.303e+01	2.957e+01	3.058e+01	3.247e+01	3.421e+01	3.207e+01	3.150e+01
Rastrigin-50D												
Mean	3.872e+02	3.914e+02	3.933e+02	3.943e+02	3.857e+02	3.897e+02	3.467e+02	3.475e+02	3.423e+02	3.516e+02	3.526e+02	3.496e+02
SD	1.896e+01	2.365e+01	1.873e+01	2.478e+01	1.971e+01	1.975e+01	1.877e+01	1.625e+01	1.588e+01	1.688e+01	1.497e+01	1.314e+01
Median	3.865e+02	3.958e+02	3.911e+02	4.012e+02	3.873e+02	3.911e+02	3.477e+02	3.502e+02	3.416e+02	3.580e+02	3.532e+02	3.531e+02
Min(best)	3.457e+02	3.299e+02	3.460e+02	3.407e+02	3.406e+02	3.588e+02	3.031e+02	2.992e+02	3.131e+02	2.978e+02	3.035e+02	3.218e+02
Max(worst)	4.218e+02	4.414e+02	4.259e+02	4.373e+02	4.172e+02	4.403e+02	3.726e+02	3.713e+02	3.697e+02	3.734e+02	3.764e+02	3.677e+02

Table 7. Test statistics for the numerical experiments (cont.)

	Optimization budget=200 evaluations						Optimization budget=2000 evaluations					
	Sample size=10%			Sample size=25%			Sample size=10%			Sample size=25%		
	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS
Ackley-10D												
Mean	5.071e+00	5.036e+00	5.740e+00	4.911e+00	5.034e+00	6.751e+00	4.014e+00	3.825e+00	3.902e+00	3.903e+00	3.959e+00	3.932e+00
SD	7.940e-01	6.907e-01	1.170e+00	7.868e-01	7.415e-01	2.039e+00	2.819e-01	3.572e-01	3.689e-01	3.501e-01	3.579e-01	2.613e-01
Median	4.999e+00	5.059e+00	5.389e+00	4.852e+00	4.979e+00	6.357e+00	4.050e+00	3.806e+00	4.024e+00	3.907e+00	4.006e+00	3.961e+00
Min(best)	3.714e+00	3.716e+00	3.534e+00	3.545e+00	4.117e+00	4.272e+00	3.486e+00	3.095e+00	2.745e+00	2.941e+00	2.444e+00	3.265e+00
Max(worst)	7.396e+00	6.816e+00	9.866e+00	6.739e+00	7.584e+00	1.343e+01	4.585e+00	4.499e+00	4.429e+00	4.482e+00	4.456e+00	4.343e+00
Ackley-50D												
Mean	9.439e+00	9.313e+00	9.529e+00	9.281e+00	9.402e+00	9.610e+00	8.629e+00	8.624e+00	8.681e+00	8.743e+00	8.624e+00	8.711e+00
SD	3.072e-01	2.714e-01	3.778e-01	3.432e-01	3.508e-01	6.233e-01	2.709e-01	2.434e-01	2.278e-01	2.278e-01	2.816e-01	2.363e-01
Median	9.466e+00	9.358e+00	9.484e+00	9.268e+00	9.292e+00	9.616e+00	8.625e+00	8.618e+00	8.698e+00	8.765e+00	8.699e+00	8.746e+00
Min(best)	8.873e+00	8.887e+00	8.432e+00	8.647e+00	8.740e+00	8.174e+00	7.854e+00	8.152e+00	8.138e+00	8.150e+00	7.793e+00	8.029e+00
Max(worst)	1.007e+01	9.766e+00	1.024e+01	9.934e+00	1.002e+01	1.070e+01	9.042e+00	9.161e+00	8.971e+00	9.057e+00	9.045e+00	9.129e+00
Griewank-10D												
Mean	9.593e-01	9.329e-01	9.550e-01	9.472e-01	9.287e-01	1.015e+00	7.309e-01	7.057e-01	7.323e-01	7.305e-01	7.722e-01	7.455e-01
SD	7.098e-02	8.261e-02	1.715e-01	5.681e-02	8.293e-02	9.314e-02	9.894e-02	1.219e-01	1.065e-01	8.088e-02	9.177e-02	8.081e-02
Median	9.739e-01	9.454e-01	9.731e-01	9.656e-01	9.613e-01	1.015e+00	7.580e-01	7.287e-01	7.484e-01	7.233e-01	7.935e-01	7.602e-01
Min(best)	7.738e-01	7.564e-01	4.846e-01	7.892e-01	7.035e-01	7.562e-01	4.826e-01	3.167e-01	4.284e-01	5.822e-01	5.517e-01	5.777e-01
Max(worst)	1.105e+00	1.039e+00	1.643e+00	1.019e+00	1.032e+00	1.324e+00	8.903e-01	8.910e-01	8.946e-01	8.845e-01	9.155e-01	9.111e-01
Griewank-50D												
Mean	1.707e+00	1.701e+00	1.768e+00	1.670e+00	1.728e+00	1.811e+00	1.577e+00	1.569e+00	1.562e+00	1.568e+00	1.573e+00	1.554e+00
SD	7.716e-02	9.085e-02	1.231e-01	8.741e-02	7.512e-02	1.357e-01	4.306e-02	4.219e-02	5.766e-02	5.007e-02	4.900e-02	6.140e-02
Median	1.696e+00	1.721e+00	1.774e+00	1.657e+00	1.719e+00	1.816e+00	1.579e+00	1.567e+00	1.567e+00	1.561e+00	1.565e+00	1.559e+00
Min(best)	1.531e+00	1.543e+00	1.517e+00	1.507e+00	1.612e+00	1.561e+00	1.493e+00	1.494e+00	1.433e+00	1.482e+00	1.474e+00	1.400e+00
Max(worst)	1.877e+00	1.898e+00	2.123e+00	1.892e+00	1.903e+00	2.142e+00	1.670e+00	1.691e+00	1.687e+00	1.680e+00	1.695e+00	1.644e+00
Rastrigin-10D												
Mean	4.362e+01	4.392e+01	4.446e+01	4.633e+01	4.413e+01	4.768e+01	2.261e+01	2.262e+01	2.362e+01	2.441e+01	2.436e+01	2.345e+01
SD	7.226e+00	8.564e+00	8.538e+00	8.702e+00	7.127e+00	1.018e+01	3.920e+00	5.430e+00	6.220e+00	5.694e+00	4.737e+00	3.921e+00
Median	4.410e+01	4.598e+01	4.319e+01	4.648e+01	4.375e+01	4.680e+01	2.254e+01	2.429e+01	2.484e+01	2.571e+01	2.499e+01	2.323e+01
Min(best)	2.347e+01	2.650e+01	2.693e+01	2.407e+01	2.527e+01	3.174e+01	1.367e+01	1.258e+01	7.150e+00	1.293e+01	1.525e+01	1.432e+01
Max(worst)	5.758e+01	6.095e+01	6.050e+01	6.333e+01	5.490e+01	7.303e+01	2.957e+01	3.058e+01	3.247e+01	3.421e+01	3.207e+01	3.150e+01
Rastrigin-50D												
Mean	3.872e+02	3.914e+02	3.933e+02	3.943e+02	3.857e+02	3.897e+02	3.467e+02	3.475e+02	3.423e+02	3.516e+02	3.526e+02	3.496e+02
SD	1.896e+01	2.365e+01	1.873e+01	2.478e+01	1.971e+01	1.975e+01	1.877e+01	1.625e+01	1.588e+01	1.688e+01	1.497e+01	1.314e+01
Median	3.865e+02	3.958e+02	3.911e+02	4.012e+02	3.873e+02	3.911e+02	3.477e+02	3.502e+02	3.416e+02	3.580e+02	3.532e+02	3.531e+02
Min(best)	3.457e+02	3.299e+02	3.460e+02	3.407e+02	3.406e+02	3.588e+02	3.031e+02	2.992e+02	3.131e+02	2.978e+02	3.035e+02	3.218e+02
Max(worst)	4.218e+02	4.414e+02	4.259e+02	4.373e+02	4.172e+02	4.403e+02	3.726e+02	3.713e+02	3.697e+02	3.734e+02	3.764e+02	3.677e+02

4. Results and Discussion

This section provides the results and discussion of the numerical experiments performed, first where mathematical test functions were used, and then where an engineering simulation code served as the expensive objective function.

4.1. Experiments Involving Mathematical Test Functions

As mentioned in Section 3, 48 optimization scenarios were considered and 30 runs were performed with each method in turn, where each run consisted of an initial sampling stage followed by a full optimization search. In the following analysis, the effectiveness of the sampling methods was compared based on the final result obtained in the optimization search, which follows the approach of other studies such as Toal et al. (2008). Table 6 gives the resultant test statistics of mean, standard deviation (SD), median, minimum (best), and maximum (worst) function value. In each scenario, the best mean statistic is emphasized.

To identify significant trends in the results, the following analysis was used:

- Step 1: In each optimization scenario the three sampling methods were assigned scores, as follows:
 - The three methods were ranked based on their corresponding mean statistic, where the method yielding the best (lowest) mean statistic was assigned a score of two, the second best a score of one, and the worst a score of zero.
 - Independently of the latter scores, and for each method in turn, its corresponding results were compared to those of the other two methods, to determine if its results achieved a *statistically significant* advantage based on the nonparametric Mann–Whitney test (Sheskin, 2007, p.423–434). Differences between the test results were deemed as statistically significant at the 0.05 significance level. Each method was assigned a score which is the number of comparisons in which its corresponding test results achieved a statistically significant advantage, namely, either two, one, or zero.

Using these two statistics allowed to identify trends of different magnitudes in the data. Table 8 gives the resultant scores for the three sampling methods.

Table 8. Scores of the three sampling methods in each scenario

(a) Scores based on the mean statistic

Function	Optimization budget=200 evaluations						Optimization budget=2000 evaluations					
	Sample size=10%			Sample size=25%			Sample size=10%			Sample size=25%		
	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS
Ackley-10D	1	2	0	2	1	0	0	2	1	2	0	1
Ackley-50D	1	2	0	2	1	0	1	2	0	0	2	1
Griewank-10D	0	2	1	1	2	0	1	2	0	2	0	1
Griewank-50D	1	2	0	2	1	0	0	1	2	1	0	2
Rastrigin-10D	2	1	0	1	2	0	2	1	0	0	1	2
Rastrigin-50D	2	1	0	0	2	1	1	0	2	1	0	2
Rosenbrock-10D	1	2	0	2	1	0	0	1	2	0	2	1
Rosenbrock-50D	1	2	0	2	1	0	2	1	0	0	2	1
Schwefel-10D	2	0	1	0	2	1	2	1	0	0	2	1
Schwefel-50D	2	1	0	2	1	0	2	0	1	1	2	0
Weierstrass-10D	1	2	0	2	1	0	0	1	2	0	1	2
Weierstrass-50D	1	2	0	2	1	0	1	0	2	0	1	2

(b) Scores based on statistical significance tests

Function	Optimization budget=200 evaluations						Optimization budget=2000 evaluations					
	Sample size=10%			Sample size=25%			Sample size=10%			Sample size=25%		
	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS	LHS	SDS	MCS
Ackley-10D	1	1	0	1	1	0	0	1	0	0	0	0
Ackley-50D	0	1	0	1	1	0	0	0	0	0	1	0
Griewank-10D	0	0	0	1	1	0	0	0	0	1	0	0
Griewank-50D	1	1	0	2	1	0	0	0	0	0	0	0
Rastrigin-10D	0	0	0	0	0	0	0	0	0	0	0	0
Rastrigin-50D	0	0	0	0	1	0	0	0	0	0	0	0
Rosenbrock-10D	0	0	0	1	1	0	0	0	0	0	2	0
Rosenbrock-50D	0	0	0	1	1	0	0	0	0	0	0	0
Schwefel-10D	0	0	0	0	0	0	0	0	0	0	0	0
Schwefel-50D	0	0	0	1	0	0	0	0	0	0	1	0
Weierstrass-10D	0	2	0	1	1	0	0	0	0	0	0	0
Weierstrass-50D	1	1	0	1	1	0	0	0	0	0	0	0

- Step 2: The individual scores were aggregated to yield *cumulative scores* which highlight the effect of the different control variables on the results. The exception is the *function type* control variable (Table 2) since in practise the features of a black-box function are not known prior to the optimization search. Therefore, the impact of the other three control variables (*initial sample size*, *function dimension*, and *optimization budget*) were calculated across the full set of test functions, and not on a per-function basis.

For the analysis, cumulative scores were calculated based on three, and based on each control variable individually. For each sampling method, its cumulative score for a given combination of control variables settings was calculated by aggregating its individual scores from the optimization scenarios across all test functions in which the control variables had the settings in question. Cumulative scores were calculated separately based on the mean statistic scores and the statistical significance scores, and they indicate which sampling method was most beneficial at a given combination of control variables settings.

Figure 3 and 4 show plots of the resultant cumulative scores, where in the x -axis label, the control variables are abbreviated by b for optimization budget, d for function dimension, and s for sample size. In the following analysis, the term *performance* relates to the cumulative score a sampling method as compared to the other methods:

- Cumulative scores based on three control variables (Figure 3): The mean statistic analysis shows that the micro-EA SDS performed well in scenarios with a large optimization budget. LHS and Monte Carlo sampling performed similarly across the different scenarios.

The statistical significance analysis shows that micro-EA SDS did not outperform the DOE methods in any scenario. Similarly to the mean statistic analysis, the performance of LHS and Monte Carlo sampling was comparable.

- Analysis based on a single control variable (Figure 4):
 - Function dimension: The mean statistic analysis shows that all three methods performed consistently, namely, were similarly affected by the increase in function dimension. The statistically significant

analysis shows that the performance of LHS slightly improved with function dimension.

- Sample size: The mean statistic analysis shows that the performance of micro-EA SDS improved with the sample size, which is attributed to the extended micro-EA search. However, both LHS and Monte Carlo sampling outperformed the micro-EA SDS across the two settings.

The statistical significance analysis shows that the performance of LHS and Monte Carlo sampling improved with the sample size.

- Optimization budget: The mean statistic analysis shows that the performance of micro-EA SDS significantly improved with the optimization budget size. Monte Carlo sampling performed consistently, while the performance of LHS degraded.

The statistical significance analysis shows trends similar to those above.

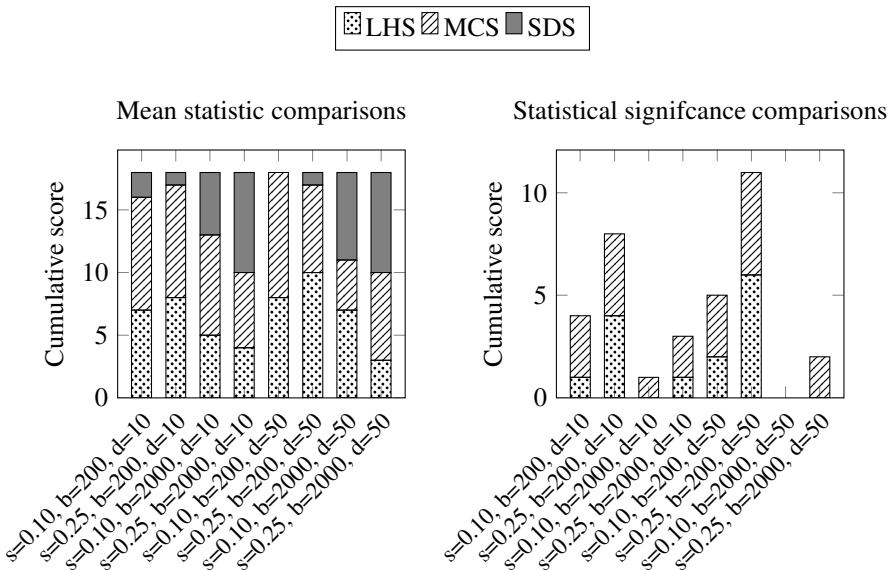


Figure 3. Cumulative scores based on three control variables.

In summary, the above analysis shows that the performance of the micro-EA SDS method was strongly affected by the size of the initial sample. As

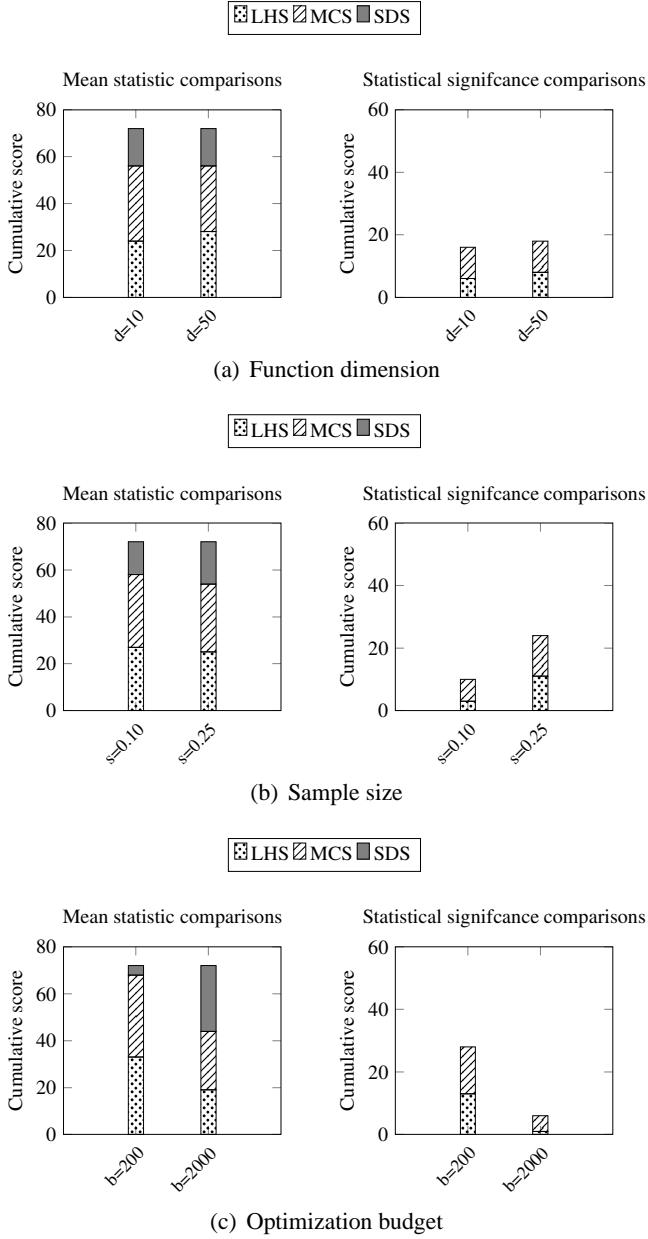


Figure 4. Cumulative scores based on a single control variable.

the sample size increased so did the relative performance of the micro-EA SDS method, as this enabled a more extended and hence more effective micro-EA search. In scenarios with a small sample size the DOE methods performed better since they were able to distribute the small number of sample vectors more effectively in the search space when compared to micro-EA SDS, which in turn yielded a more accurate metamodel and improved the optimization search .

4.2. Experiments Involving a Computer Simulation as the Objective Function

To augment the preceding analysis an additional set of experiments were performed, but which involved a computer simulation as the objective function. The optimization problem which was being solved was that of airfoil shape optimization which is formulated as follows. During flight, an aircraft generates *lift*, namely, the force which counters the aircraft weight and keeps it airborne, and *drag*, which is an aerodynamic friction force obstructing the aircraft's movement. Both the lift and drag are strongly determined by the wing cross-section, namely, the *airfoil*. The optimization goal is then to find an airfoil shape which maximizes the lift and minimizes the drag.

In practise, the design requirements for airfoils are specified in terms of the nondimensional lift and drag coefficients, c_l and c_d , respectively, defined as

$$c_l = \frac{L}{\frac{1}{2}\rho V^2 S} \quad (8a)$$

$$c_d = \frac{D}{\frac{1}{2}\rho V^2 S} \quad (8b)$$

where L and D are the lift and drag forces, respectively, ρ is the air density, V is aircraft speed, and S is a reference area, such as the wing area. Also important is the *angle of attack* (AOA), which is the angle between the aircraft velocity and the airfoil *chord line*, defined as the straight line joining the leading and trailing edges of the airfoil. Figure 5 gives a schematic layout of the airfoil problem.

Candidate airfoils were represented with the Hicks-Henne parameterization (Hicks and Henne, 1978) where an airfoil profile is defined by

$$y = y_b + \sum_{i=1}^h \alpha_i b_i(x), \quad (9)$$

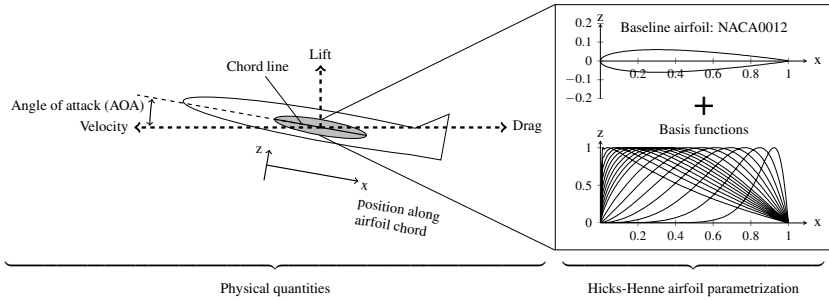


Figure 5. A schematic layout of the airfoil problem.

where y_b is a baseline airfoil profile, which in this study was taken as the NACA0012 symmetric airfoil, b_i are basis functions (Wu et al., 2003) defined as

$$b_i(x) = \left[\sin \left(\pi x \frac{\log(0.5)}{\log(\frac{t}{(h+1)})} \right) \right]^4 \tag{10}$$

and $\alpha_i \in [-0.01, 0.01]$ are coefficients, namely, the problem’s design variables. Figure 5 summarizes the nomenclature of the physical quantities and airfoil parameterization involved.

The lift and drag coefficients of candidate airfoils were obtained using *XFOil*, a computational aerodynamics simulation for analysis of airfoils operating in the subsonic regime (Drela and Youngren, 2001). Each airfoil evaluation required up to 30 seconds on a desktop computer. To ensure structural integrity, between 0.2 to 0.8 of the chord line the airfoil thickness (t) had to be equal to or larger than a critical value $t^* = 0.1$. The flight conditions were set as a cruise altitude of 30 kft, a cruise speed of Mach 0.7, namely, 70% of the speed of sound, and an AOA of 2° . The objective function employed was

$$f = -\frac{c_l}{c_d} + p \tag{11a}$$

where p is a penalty for violation of the thickness constraint, such that

$$p = \begin{cases} \frac{t^*}{t} \cdot \left| \frac{c_l}{c_d} \right| & \text{if } t < t^* \\ 0 & \text{otherwise} \end{cases} \tag{11b}$$

Figure 6 shows the plots for the cumulative scores by a single control vari-

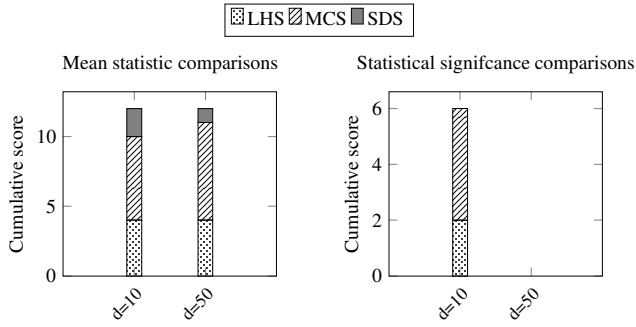
able, and it follows that the results are overall consistent with those in Section 4.1, namely:

- The micro-EA SDS method did not achieve a statistically significant advantage in any scenario.
- The mean statistic analysis shows that the performance of micro-EA SDS improved with sample size.
- The three methods were similarly affected by the increase in function dimension.

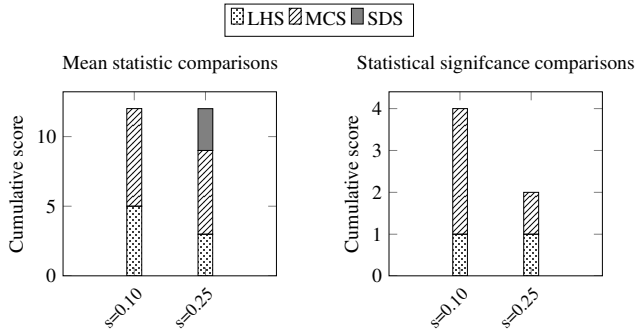
It follows that these trends are consistent with those of Section 4.1, which therefore validates the overall analysis presented.

5. Conclusion

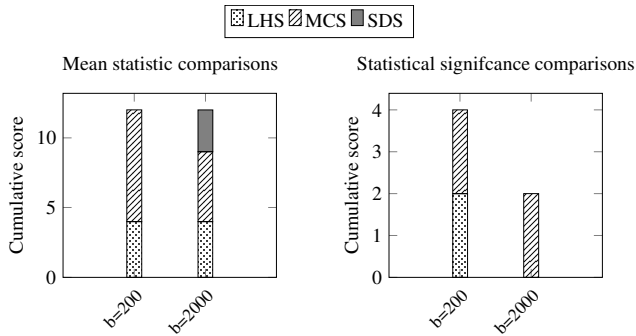
Metamodels-assisted algorithms are often applied to optimization problems involving an expensive black-box function, namely, where a computationally expensive simulation acts as the objective function. Such algorithms typically begin by generating an initial sample of vectors, and use them to initiate the main optimization search. The sample can be generated by a design of experiments (DOE) method which is statistically-based, or by the more recent approach of search-driven sampling (SDS) which uses a direct search optimization algorithm. In particular, a micro-EA can be used for a short duration, after which the vectors it evaluated serve as the initial sample. In this study an extensive comparison was performed between two DOE methods (Latin hypercube design, Monte Carlo sampling) and an SDS approach which uses a micro-EA. The three methods were compared based on their impact on the search effectiveness, and this was achieved by numerical experiments based both on mathematical test functions and simulation-driven problem. The main conclusion from the results is that the micro-EA SDS method was effective when the initial sample was large, since then the larger number of function evaluations allocated to the initial sample allowed for a more effective micro-EA search. In other settings, the DOE methods were more effective as they were able to distribute the small number of sample vectors more effectively in the search space which in turn yielded a more accurate metamodel.



(a) Function dimension



(b) Sample size



(c) Optimization budget

Figure 6. Cumulative scores in the airfoil problem, based on a single control variable.

References

- Y. Tenne, C. K. Goh (Eds.), *Computational Intelligence in Expensive Optimization Problems*, vol. 2 of *Evolutionary Learning and Optimization*, Springer, Berlin, 2010.
- A. I. J. Forrester, A. J. Keane, Recent advances in surrogate-based optimization, *Progress in Aerospace Science* 45 (1–3) (2008) 50–79.
- N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, K. P. Tucker, Surrogate-based analysis and optimization, *Progress in Aerospace Science* 41 (2005) 1–28.
- V. P. Chen, R. R. Barton, M. Meckesheimer, K.-L. Tsui, A Review on Design, Modeling and Applications of Computer Experiments, *IIE Transactions* 38 (4) (2006) 273–291.
- A. Sóbester, S. J. Leary, A. J. Keane, On the Design of Optimization Strategies Based on Global Response Surface Approximation Models, *Journal of Global Optimization* 33 (2005) 31–59.
- G. G. Wang, S. Shan, Review of Metamodeling Techniques in Support of Engineering Design Optimization, *Journal of Mechanical Design* 129 (4) (2007) 370–380.
- E. Zahara, Y.-T. Kao, Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems, *Expert Systems with Applications* 36 (2009) 3880–3886.
- B. V. Babu, A. Rakesh, Modified Differential Evolution (MDE) for Optimization of Non-Linear Chemical Processes, *Computers and Chemical Engineering* 30 (6–7) (2006) 989–1002.
- K. A. de Jong, *Evolutionary Computation: A Unified Approach*, MIT Press, Cambridge, Massachusetts, 2006.
- R. A. Fisher, The arrangement of field experiments, *Journal of the Ministry of Agriculture* 33 (1926) 503–513.
- D. J. Finney, Fractional replication of factorial arrangements, *Annals of Eugenics* 12 (1945) 291–301.

- G. E. P. Box, K. B. Wilson, On the experimental attainment of optimum conditions, *Journal of the Royal Statistical Society Series B (Methodological)* 13 (1951) 1–45.
- R. H. Myers, D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley and Sons, New York, 1995.
- N. Metropolis, S. Ulam, The Monte Carlo Method, *Journal of the American Statistical Association* 44 (247) (1949) 335–341.
- M. D. McKay, R. J. Beckman, W. J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics* 21 (2) (1979) 239–245.
- A. B. Owen, Orthogonal Arrays for Computer Experiments, Integration and Visualization, *Statistica Sinica* 2 (1992) 439–452.
- M. E. Johnson, L. M. Moore, D. Ylvisaker, Minimax and Maximin Distance Designs, *Journal of Statistical Planning and Inference* 26 (2) (1990) 131–148.
- R. Jin, W. Chen, A. Sudjianto, An Efficient Algorithm for Constructing Optimal Design of Computer Experiments, *Journal of Statistical Planning and Inference* 134 (1) (2005) 268–287.
- N.-H. Quttineh, K. Holmström, The influence of Experimental Designs on the Performance of Surrogate Model Based Costly Global Optimization Solvers, *Studies in Informatics and Control* 18 (1) (2009) 87–95.
- D. Büche, N. N. Schraudolph, P. Koumoutsakos, Accelerating Evolutionary Algorithms With Gaussian Process Fitness Function Models, *IEEE Transactions on Systems, Man, and Cybernetics—Part C* 35 (2) (2005) 183–194.
- K.-H. Liang, X. Yao, C. Newton, Evolutionary Search of Approximated N-dimensional Landscapes, *International Journal of Knowledge-Based Intelligent Engineering Systems* 4 (3) (2000) 172–183.
- J. Laurenceau, P. Sagaut, Building Efficient Response Surfaces of Aerodynamic Functions with Kriging and Cokriging, *AIAA Journal* 46 (2) (2008) 498–507.

- P. Mugunthan, C. A. Shoemaker, Assessing the Impacts of Parameter Uncertainty for Computationally Expensive Groundwater Models, *Water Resources Research* 42 (10).
- H. You, M. Yang, D. Wang, X. Jia, Kriging Model combined with Latin hypercube sampling for surrogate modeling of analog integrated circuit performance, in: *Proceedings of the Tenth International Symposium on Quality Electronic Design–ISQED 2009*, IEEE, Piscataway, NJ, 554–558, 2009.
- M. A. El-Betalgy, A. J. Keane, Evolutionary Optimization for Computationally Expensive Problems using Gaussian Processes, in: H. Arbania (Ed.), *Proceedings of the International Conference on Artificial Intelligence–ICAI’2001*, CSREA Press, 708–714, 2001.
- F. Neri, X. del Toro Garcia, G. L. Cascella, N. Salvatore, Surrogate Assisted Local Search on PMSM Drive Design, *International Journal for Computation and Mathematics in Electrical and Electronic Engineering* 27 (3) (2008) 573–592.
- K. Krishnakumar, Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization, in: G. E. Rodriguez (Ed.), *Intelligent Control and Adaptive Systems, Proceedings of SPIE—the International Society for Optical Engineering*, SPIE, Bellingham, Wash., USA, 1989.
- P. K. Senecal, Numerical Optimization Using the GEN4 Micro-Genetic Algorithm Code, *Tech. Rep.*, Engine Research Center, University Of Wisconsin-Madison, Wisconsin, USA, 2000.
- A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca, *Genetic Algorithm TOOLBOX For Use with MATLAB*, Version 1.2, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, 1994.
- P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, S. Tiwari, *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*, Technical Report KanGAL 2005005, Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, India, 2005.
- A. Ratle, Optimal Sampling Strategies for Learning a Fitness Model, in: *The 1999 IEEE Congress on Evolutionary Computation–CEC 1999*, IEEE, Piscataway, New Jersey, 2078–2085, 1999.

- J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and Analysis of Computer Experiments, *Statistical Science* 4 (4) (1989) 409–435.
- J. D. Martin, T. W. Simpson, Use of Kriging Models to Approximate Deterministic Computer Models, *AIAA Journal* 43 (4) (2005) 853–863.
- D. J. J. Toal, N. W. Bressloff, A. J. Keane, Kriging hyperparameter tuning strategies, *AIAA Journal* 46 (5) (2008) 1240–1252.
- D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall, Boca Raton, Florida, fourth edn., 2007.
- R. M. Hicks, P. A. Henne, Wing Design by Numerical Optimization, *Journal of Aircraft* 15 (7) (1978) 407–412.
- H.-Y. Wu, S. Yang, F. Liu, H.-M. Tsai, Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization, in: *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, AIAA 2003-4095, 2003.
- M. Drela, H. Youngren, *XFOIL 6.9 User Primer*, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 2001.

INDEX

A

accounting, 37, 83, 114
action potential, 75
actuators, 130
ADA, 170
adaptations, 31, 78, 109, 110, 193
adjustment, 69
aesthetics, 161, 178
aggregation, 46
algorithm, vii, ix, x, 9, 11, 13, 17, 18, 19,
20, 21, 22, 31, 33, 34, 50, 61, 62, 63, 66,
84, 86, 87, 90, 103, 120, 121, 122, 123,
124, 125, 126, 128, 130, 132, 133, 134,
135, 136, 138, 140, 145, 146, 147, 152,
154, 155, 157, 160, 183, 186, 187, 188,
190, 193, 195, 196, 197, 208
alters, 75
amnesia, 104
amplitude, 80
ancestors, 15
annealing, 13, 156
arithmetic, 46
artificial intelligence, 35
assessment, 17, 147
asset price, vii, 1
assets, 36, 43, 45, 50, 51, 53, 54
atmosphere, 173
AURAL, ix, x, 159, 161, 162, 165, 167,
168, 169, 172, 173, 175, 176, 178, 179,
180
automation, 161
autonomy, 25
axon terminals, 75

B

background information, 185
barriers, 178
base, 5, 6, 125, 136, 171
beetles, 110
behaviors, 177
Beijing, 67
bias, 84, 193
biomass, viii, 107
biotic, 109, 110, 112
birds, 21, 78, 80, 104, 110, 113, 116, 117
Boltzmann distribution, 9
bonding, 77
bounds, 145, 150
brain, 75, 77, 78, 79, 80, 81, 82, 83, 84, 86,
103
brain activity, 82
brain damage, 86, 103
branching, 119
Brazil, 133, 150
Brownian motion, 115, 116

C

C++, 145
caching, 114, 117
calcium, 82
calibration, 119
capital markets, 68
case study, 62, 71
cash, 43, 50, 51
catalyst, 142

CDC, 132
 CEC, 68, 70, 130, 132, 212
 cell body, 75
 cerebellum, 80
 chemical, 141, 148, 156
 Chicago, 38, 130
 children, 122
 chromosome, 15, 18, 26, 37, 42, 49, 51
 cladding, 144
 clarity, 7
 classes, 26, 27, 186
 classification, 25, 26, 28
 clone, 125
 clustering, 28, 33, 123, 135, 137, 138, 155, 156
 clusters, 111, 135, 138
 coding, 5
 cognition, 104, 115
 cognitive tasks, 84
 collisions, 175
 colonisation, 110, 111, 120
 colonization, 130
 color, 169, 177
 combustion, 141
 commercial, 184
 communication, 22, 161, 178
 communities, 109, 113
 community, 14
 competition, 111, 120, 121
 complexity, 17, 74, 85, 143, 154, 170
 composers, 160, 178
 composition, 69, 160, 161, 162, 165, 167, 169, 177, 178, 179, 180
 computation, vii, 17, 37, 64, 70, 102, 126
 computational intelligence, vii, x, 183
 Computational Problem Solving Techniques, 61
 computer, 5, 6, 14, 47, 160, 167, 172, 180, 184, 188, 206, 207, 211
 computer simulations, 184
 computing, 5, 17, 22, 126, 128, 132, 150, 152, 178
 conference, 62, 64, 65, 66, 67, 70
 configuration, 13, 112, 143
 Congress, 63, 68, 70, 128, 130, 132, 156, 157, 212
 connectivity, 78, 83
 consolidation, viii, 73, 76, 77, 79, 80, 81, 82, 84, 86, 102, 104
 construction, 74, 113, 138, 140, 164

consumers, 117
 consumption, 110, 117
 convergence, 22, 126, 188
 cooling, 11, 13
 correlation, 47, 48, 57, 58, 61, 62, 193, 195, 196
 correlation analysis, 57
 correlation function, 195
 correlations, 58, 196
 cortex, 81, 82
 cosmetic, 14
 cost, vii, x, 30, 51, 83, 85, 112, 148, 183, 185, 186
 covering, 138
 creative process, 160
 creativity, 160, 161, 178
 crises, 36
 critical value, 207
 cross-validation, 92, 102
 cycles, 80, 87, 88, 90, 91, 92, 93, 94, 95, 96, 97, 98, 101, 102, 163, 177
 cycling, 13

D

data distribution, 92
 data structure, 15, 36
 database, 5, 52, 170, 176, 177
 DAX 30, 1, 35, 52
 decay, 76, 79, 84, 85, 86, 102
 decision trees, 28
 declarative memory, 77, 80, 81, 86
 delta wave, 80
 demography, 112
 dendrites, 6, 75
 dendritic spines, 75, 105
 deposition, 114
 deposits, 44
 derivatives, 86
 designers, 177
 detachment, 115
 deviation, 144
 diffusion, 115
 digestion, 117
 dimensionality, 88, 142, 193
 directionality, 113
 discrete variable, 144
 disorder, 170
 dispersion, ix, 44, 108, 109, 113, 123

displacement, 114, 122
 distribution, 54, 58, 59, 88, 112, 115, 116,
 118, 119, 189
 distribution function, 118
 diversity, 17, 113, 121, 134, 149
 DNA, 126, 132
 dominance, 52
 drawing, ix, 74, 108, 122, 173
Drosophila, 103, 105
 dynamic systems, 161, 173

E

ecology, ix, 108, 115, 117
 electroencephalogram, 80
 electronic circuits, 191
 empirical studies, 113, 116
 encoding, 37, 80, 81, 82, 126, 132
 energy, 13, 83, 108, 111, 112, 171
 energy consumption, 83
 engineering, vii, x, 120, 126, 127, 134, 152,
 156, 183, 184, 185, 201, 210
 England, 52
 environment, ix, 17, 25, 36, 51, 81, 109,
 110, 111, 113, 116, 117, 159, 160, 161,
 162, 170, 172, 173, 177, 178, 180
 environmental conditions, 113
 episodic memory, 78
 equilibrium, 148, 154
 equity, 66
 eukaryotic biomass, viii, 107
 Europe, 137
 evidence, 81, 82, 83, 84
 evolution, vii, ix, 1, 14, 15, 35, 78, 90, 92,
 104, 133, 134, 135, 136, 149, 152, 153,
 154, 155, 156, 157, 162, 167, 177
 evolutionary algorithm, vii, x, 50, 66, 157,
 183, 184, 187, 188
 evolutionary computation, 36, 64, 65, 66,
 67, 70, 149
 evolutionary principles, 17
 evolutionary-based sampling, x, 183
 exclusion, 34
 execution, 147
 exploitation, 122, 126, 134
 exposure, 38, 44, 47, 53, 54, 60, 87
 expressivity, 178
 extraction, 81
 eye movement, 80

F

financial data, 53
 financial markets, 36, 69
 Financial Markets, vii, 1, 2, 37, 69, 70
 financial support, 102, 150
 fires, 75
 first generation, 15, 34
 fish, 21, 110
 fitness, 3, 16, 17, 21, 33, 36, 37, 38, 42, 50,
 56, 74, 120, 121, 122, 123, 124, 125,
 127, 137, 141, 147, 149, 157, 162, 163,
 165, 173
 flight, 116, 206, 207
 flights, 131
 flora, 109
 fluctuations, 53, 115
 food, 19, 108, 114, 116, 117
 force, 3, 206
 forecasting, 62, 69, 105
 formation, 75, 81, 82, 105
 formula, 46
 fragments, 170, 175, 176, 177
 France, 65, 66
 friction, 206
 fruits, 109, 110, 112
 FTSE 100, vii, 1, 35, 52
 function values, 137, 139, 186, 188, 189
 funds, 44
 fungi, 109

G

genes, 15, 42
 genetic programming, 37, 71
 genotype, 163
 genre, 165
 geometry, 77
 Georgia, 64, 65, 66, 67
 Germany, 52
 germination, 108, 112, 113, 117
 glutamate, 76
 google, 180
 graph, 19, 43, 138, 139
 gravity, 109
 growth, 111, 117, 120, 124, 126
 guidelines, 185

H

habitats, 111
 harmony, 162, 163
 hippocampus, 77, 78, 80, 81
 histogram, 88
 historical data, 42, 52
 homeostasis, viii, 73, 74, 79, 80, 82, 104, 105
 horizontal transmission, 18
 human brain, 6
 hybrid, 18, 134, 155
 hydrogen, 142
 hypercube, 138, 188, 190, 191, 208, 212
 hypothesis, viii, 73, 74, 75, 76, 79, 80, 82, 83, 105, 110, 111, 116, 129
 hypothesis test, 129

I

identification problem, 28
 images, 169, 177
 improvements, 13, 61
 income, 43
 indexing, 105
 India, 212
 individuality, 162
 individuals, 15, 18, 21, 32, 33, 34, 135, 137, 140, 141, 147, 149, 163
 insects, ix, 108, 110, 114, 116
 integration, 74
 integrity, 207
 intelligence, vii, x, 3, 66, 178, 183
 interface, ix, 5, 159, 161, 163, 165, 166, 167, 177
 interference, 76
 intervention, 25, 170
 investment, viii, 2, 17, 43, 44, 50, 53, 69, 110, 111, 112, 126
 Investment Strategy, vii, 1, 35
 investments, 43, 44, 110
 investors, 36, 51, 61
 ions, 76, 77
 Ireland, 73, 102, 107
 Israel, 183
 iteration, 13, 21, 31, 122, 186, 189, 193

J

Japan, 52, 180

K

Korea, 40

L

landscape(s), 108, 109, 125, 129, 134, 184, 187
 languages, 62
 larvae, 110
 laws, 14
 lead, 19, 47, 82, 83
 learning, viii, 18, 22, 25, 26, 37, 74, 75, 76, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 102, 104, 105, 173, 175
 learning process, 85, 86, 87, 102
 learning task, 82
 light, 111, 119, 144
 linear model, 85
 lipids, 110
 liquidity, 44
 long-term memory, 81
 Louisiana, 63

M

machine learning, 15, 28, 36, 74, 102
 magnitude, 102, 115
 mammals, 80, 104, 110, 116, 117
 management, viii, 2, 37
 manufacturing, 143
 mapping, ix, 159, 162
 marriage, 18
 mass, 83, 109, 143, 144
 materials, 144, 164
 matrix, 138, 139, 140, 170, 192, 196
 matter, 29, 48
 measurements, 142, 155
 median, 58, 90, 201
 melody, 162, 163
 memory, viii, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 84, 86, 103, 104, 105, 170, 175, 187, 193

memory consolidation, viii, 73, 76, 77, 79,
80, 81, 82, 84, 86
memory formation, 74, 79
memory function, 82, 103
metals, 13
metamodel-assisted optimization, vii, x,
183, 184, 185, 186, 187
meter, 163, 164
methodology, viii, 29, 35, 73, 184, 188
mice, 82
microorganisms, 115
migration, 115
mixing, 160
MLP, viii, 73, 74, 79, 85, 86, 87, 88, 90, 96,
97, 98, 101
mobile robots, 161, 162, 165, 170
model specification, 119
modelling, 114, 118, 129
models, 25, 68, 92, 115, 117, 119, 130, 172,
177, 179, 186
modifications, 152
modules, 37
molecules, 77
Monte Carlo method, 47
morality, 117
morphology, 76, 111
mortality, 111, 112
motivation, 36, 135, 138
multi-layer perceptron, viii, 73
multiples, 6
multivariate distribution, 191
music, 160, 161, 162, 163, 164, 165, 167,
175, 176, 177, 178, 179, 180, 181
musicians, 167, 169, 177, 179
mutation, 14, 15, 32, 36, 50, 134, 136, 154,
155, 187, 192, 195

N

narratives, 160, 178
NASDAQ, vii, 1, 35, 37, 52, 55
National Research Council, 179
natural evolution, 14, 15
neocortex, 77, 78, 81, 84
networking, 181
neural networks, 62, 69, 74, 75, 103, 105,
161, 186, 188
neurobiology, 75
neuronal cells, 6

neuronal circuits, 81
neurons, 7, 75, 76, 77, 82, 83, 84, 86, 87
neuroscience, 178
next generation, 16, 32, 122, 137, 187, 188,
192
NIKKEI 225, vii, 1, 35, 52
nodes, 7, 85, 88, 138
nonlinear systems, 150, 155
normal distribution, 115, 144

O

oligomers, 103
one dimension, 113, 118
operations, 32
opportunities, ix, 108, 112
optimization, vii, viii, ix, x, 2, 3, 4, 9, 13,
14, 15, 17, 18, 22, 28, 29, 30, 31, 32, 35,
36, 47, 48, 49, 51, 58, 61, 62, 65, 66, 69,
126, 130, 132, 133, 134, 135, 136, 137,
138, 140, 141, 142, 143, 144, 145, 146,
147, 149, 150, 151, 152, 153, 154, 155,
156, 157, 183, 184, 185, 186, 187, 190,
191, 192, 193, 195, 196, 197, 201, 203,
204, 206, 208, 210
optimization method, ix, 3, 14, 17, 22, 133,
134, 140, 143, 149, 152, 184
oyster, 109

P

palladium, 142
parallel, 81, 134, 154, 168
parallelism, 178
parameter estimation, 142, 152
parenting, 18
parents, 15, 50, 112, 136, 187, 188, 192
Pareto, vii, 1, 29, 30, 33, 34, 35, 49, 51, 52,
54, 56, 68
Pareto optimal, 49, 51
participants, 172
phenol, 142
physical mechanisms, 114, 118
physical structure, 75
physics, 146, 184
physiology, 114, 115
piano, 80, 162, 167
pitch, 162, 163, 167, 177

plant activities, viii, 107
 plant dispersal, viii, 107, 128
 plant life, viii, 107
 plants, viii, ix, 107, 108, 109, 110, 111, 112,
 113, 115, 117, 119, 120, 121, 122, 124,
 125, 126, 127, 128
 plasticity, 75, 82
 platform, 70, 170, 171
 playing, 80, 163
 pollination, 123, 124, 125
 population, 13, 14, 15, 16, 17, 18, 21, 31,
 32, 33, 34, 36, 42, 50, 52, 57, 102, 120,
 121, 122, 123, 124, 125, 127, 134, 135,
 136, 137, 140, 147, 148, 149, 163, 187,
 188, 192, 193, 195
 population size, 50, 120, 121, 122, 123, 136,
 147, 148, 193
 portfolio, 17, 18, 28, 35, 38, 69
 Portugal, 1
 positive correlation, 58
 predation, 78, 103
 predators, 111
 priming, 82
 principles, 160, 178
 probabilistic reasoning, 6
 probability, 6, 9, 10, 12, 13, 20, 24, 50, 87,
 88, 112, 113, 116, 118, 124, 192, 195
 probability density function, 118
 probability distribution, 112, 113, 116
 problem-solving, 2, 17, 18, 28, 128
 procedural memory, 80
 professionals, 17, 61
 profit, 26, 38, 43, 50
 profitability, 45, 62
 programming, 4, 5, 25, 62, 71, 161
 programming languages, 161
 project, ix, 78, 150, 159, 179
 propagation, 7, 80, 87, 88, 112, 119, 120,
 126
 propane, 141
 proteins, 76, 77, 110
 pruning, 85, 86, 104
 psychology, 75

Q

quadratic programming, 156

R

radio, 126
 radius, 124, 135, 137, 143, 149
 random numbers, 122
 random walk, 115, 116
 reaction rate, 142
 real numbers, 14, 146
 real time, 161, 162, 164, 165, 167, 177
 reality, 42, 45, 114, 160
 reasoning, 5, 6, 24, 29
 recall, 78, 81
 recalling, 82
 receptors, 76
 recognition, 86, 87, 152
 recombination, 14, 36, 187
 redistribution, 81, 82
 regression, 26, 28, 39, 85, 87, 88, 99, 100
 reinforcement, 81
 REM, 80, 81
 repetitions, 196
 replication, 210
 reprocessing, 105
 reproduction, 14, 15, 34, 120, 121, 123,
 163, 164
 requirements, 83, 112, 206
 reserves, 112
 resources, 47, 78, 111, 115, 116, 117, 120,
 121, 126, 184
 response, 84, 85, 87, 88, 90, 99, 163, 186,
 188
 rhythm, 162, 167, 177
 risk, vii, viii, 1, 2, 35, 36, 37, 38, 42, 44, 45,
 46, 47, 49, 50, 51, 52, 53, 54, 57, 58, 59,
 60, 61, 62, 69, 78, 103
 risk aversion, 61
 robotics, 160, 178
 rodents, 109
 ROI, 36, 39, 40, 43, 44, 49, 53, 54, 55, 56,
 57, 58, 59
 root system, 109
 roots(s), 26, 109, 119, 120, 125, 150
 rules, 5, 6, 8, 17, 26, 37, 62, 69, 81, 160,
 165, 170

S

S&P 500, vii, 1, 35, 52

- saturation, 84
 scaling, 83, 92, 121, 122, 136, 147
 scarcity, 118
 scatter, 132
 search space, x, 4, 9, 13, 17, 21, 36, 120,
 121, 122, 125, 127, 134, 135, 138, 184,
 188, 206, 208
 securities, 26, 38, 51
 security, 42
 seed, viii, ix, 107, 108, 109, 110, 111, 112,
 113, 114, 115, 117, 118, 119, 120, 121,
 122, 123, 124, 125, 128, 129, 130, 131,
 132
 seeding, 121, 123, 124
 seedlings, 111
 self-organization, 161
 self-regulating systems, 173
 semantic memory, 77
 sensing, 160, 161, 178
 sensors, 178
 sensory systems, ix, 159
 sequencing, 152
 shade, 111
 shape, 7, 119, 206
 shoot, 109
 short-term memory, 13
 showing, 48, 57, 62, 126, 172
 signals, 6, 42, 61, 75, 84
 significance level, 201
 signs, 139, 140
 simulation, vii, x, 50, 51, 183, 184, 185,
 186, 201, 206, 207, 208
 simulation-driven engineering, vii, x, 183
 simulations, 184
 Singapore, 38, 212
 smoothing, 85
 social environment, 117
 social learning, 117
 sodium, 76
 software, 5, 25, 184
 solution, viii, 2, 3, 4, 5, 9, 10, 13, 14, 15, 17,
 20, 21, 29, 30, 33, 36, 48, 49, 50, 51, 62,
 120, 121, 122, 123, 128, 136, 137, 138,
 141, 142, 146, 150, 178, 186, 187, 195,
 196, 197
 solution space, 50, 128
 sonification, ix, 159, 161, 162, 165, 170,
 177, 178
 sowing, 123
 specialists, 6
 species, 78, 79, 108, 110, 111, 112, 113,
 117, 118, 120
 specifications, 31, 162
 speech, 177
 spindle, 82
 spine, 76, 77, 82
 spore, 108, 113
 Spring, 70
 standard deviation, 38, 44, 46, 122, 142,
 144, 201
 standard error, 90, 96, 97, 98, 101
 state, 17, 34, 35, 75, 76, 80, 83, 104, 152
 states, 74, 78, 82
 statistics, 26, 44, 74, 198, 199, 200, 201
 stimulus, 77
 stochastic processes, 17
 stock indexes, vii, 1, 35, 36, 52, 61
 stock markets, vii, viii, 1, 2
 stock price, 36, 37
 storage, 75, 76, 81, 82, 187, 193
 storms, 115
 stress, 15, 41, 83
 striatum, 80
 structure, x, 15, 42, 74, 75, 110, 114, 160,
 164, 165, 167, 172, 177
 structuring, 131
 style, 136
 substrate, 76
 success rate, 147
 succession, 87
 surface area, 109
 surrogates, 141
 survival, 14, 15, 36, 83, 110, 111, 112, 117,
 120, 122
 susceptibility, 111
 Switzerland, 70
 synapse, 75, 77
 synaptic gap, 75
 synaptic homeostasis hypothesis, viii, 73,
 74, 79, 80, 82
 synaptic plasticity, 75
 synaptic strength, viii, 76, 83, 84
 synergetic capacity, x, 159

T

- Taiwan, 67
 target, 74, 75, 88, 92, 111

techniques, viii, ix, 2, 3, 6, 8, 9, 14, 15, 18, 20, 22, 25, 26, 28, 31, 35, 36, 37, 61, 69, 86, 112, 133, 134, 177, 181

technologies, 160

technology, 181

temperament, 162

temperature, 11, 13

territory, 120, 135

test data, viii, 2, 56, 57, 88, 89

test statistic, 196, 201

testing, 9, 52, 57, 61, 62, 88, 90, 92, 113

texture, 175

threshold level, 75

timbre, 162, 163

time frame, 38, 76

time series, viii, 2, 5, 61

tissue, 83, 108, 111

topology, 22, 113, 117

tortoises, 117

tracks, 170, 175

trade, 26, 29, 30, 35, 44, 48, 49, 50, 51, 52, 54, 57, 58, 61, 85, 111, 112, 115, 193

trade-off, 29, 30, 35, 48, 49, 50, 51, 54, 57, 58, 61, 85, 111, 112, 193

training, vii, viii, 1, 2, 8, 35, 38, 47, 49, 50, 51, 52, 53, 55, 56, 57, 58, 59, 60, 61, 62, 69, 73, 74, 75, 79, 82, 84, 85, 86, 87, 88, 89, 90, 92, 102, 186

trajectory, 20, 165, 176

transaction costs, 5

transcription, 76

transformation, 28, 160

translation, 76

transmission, 76, 77

transport, 110

trial, 8, 136, 137

triggers, 170, 171

tropical forests, 110

U

USA, 52, 63, 64, 65, 66, 67, 68, 70, 180, 212

V

validation, 52, 84

valorization, 36

variables, 4, 14, 30, 31, 36, 41, 47, 49, 57, 58, 88, 115, 142, 143, 144, 145, 146, 150, 155, 156, 186, 190, 191, 192, 203, 204, 207, 211

variations, 17, 31, 127, 143

vector, 3, 4, 21, 22, 28, 31, 87, 122, 123, 125, 127, 136, 137, 145, 191, 196

vehicles, 126

velocity, 21, 22, 114, 115, 206

vertical transmission, 18

videos, 169, 179

vision, 116, 161, 165, 168, 170, 171, 174, 175

visual system, 169

volatility, 44, 58, 60, 61, 70

W

waking, 79, 80, 81, 82, 83

Washington, viii, x, 1, 63, 64, 66, 67, 73, 107, 133, 159, 183

water, 108, 109, 110, 132, 143, 144, 173, 191

water resources, 191

wear, 143

web, 75

WIC, 66

William James, 75

wind speeds, 114, 115

windows, 87

Wisconsin, 212

Y

yield, 187, 203

Z

Zone 1, 145

Zone 2, 145

Zone 3, 145