Theoretical and practical aspects of evolutionary algorithms with application to the partitioning of multivariate self-affine time series

Christopher Michael Taylor

A thesis submitted for the degree

of Doctor of Philosophy

Department of Mathematical Sciences

University of Essex

February 2019

Contents

List of Figures

List of Algorithms

Abstract

This thesis examines certain aspects of the theory and practice of multiobjective evolutionary algorithms (MOEAs). Firstly a set of theoretical results is developed concerning the operation of ranking schemes under multiobjective tournament selection and the effect on both the development of the rank or fitness distribution. Limiting expressions are shown for the nonsampling probability according to rank, for schemes with and without replacement, with an extension including the effect of elitism. Further limiting expressions under iterated tournament selection for time to convergence and nonsampling probability are also shown. The various effects of varying the tournament size on nonsampling probability, evolution of the rank distribution, and time to convergence are discussed.

Next, the practical problem of partitioning a multivariate self-affine time series with unknown and time-varying joint distribution is considered. The aim is to completely partition the time series, possibly of high dimension, into disjoint, contiguous subseries at one timescale that are similar to subseries at another timescale after application of an affine transformation. The multiobjective combinatorial optimization problem is defined with limited assumptions as a biobjective one, and a specialized MOEA is presented which finds optimal self-affine time series partitionings with a minimum of choice parameters. The MOEA is highly specialized and contains a number

of unusual features, including permuted multiobjective tournament selection, in line with the theory developed earlier in the thesis. The algorithm seeks to simultaneously minimize both the similarity in terms of the covariance structures between successive partitions, and also the difference between the partitions defined at one timescale and at another, shorter timescale. The resulting set of Pareto-efficient solution sets provides a rich representation of the self-affine properties of a multivariate time series at different locations and time scales.

Chapter 1

Introduction

Multiobjective optimization problems often offer challenges that cannot be met by standard optimization theory and deterministic algorithms. The hurdles are set particularly high when the problem itself is hard, when specialized knowledge is required to address it, and when the global optimum is particularly difficult to find; there may be many local optima, and surrounding solutions may give little indication of their locations. As a result, a variety of stochastic search methods have become popular amongst both academics and practitioners; multiobjective evolutionary algorithms (MOEAs) are often employed when other approaches have failed, or when it is difficult to see how else to approach the problem.

Criticisms of MOEAs often fall into two main categories: the reputed slow performance of the algorithms, and their "black box" nature, meaning that they are not tractable to traditional mathematical or statistical analysis. The main contributions of this thesis fall into two principal areas which touch upon these criticisms. Firstly, advances are made in the theory of multiobjective tournament selection, which can shine a light into the admittedly often very black box of MOEA design choices. Secondly, it is demonstrated that a well-formulated, specialized MOEA not only can address a hard problem that has not been attempted before in the literature, but can do so with acceptable computational performance and indicative results, using models and techniques based on sound

theory.

The "no free lunch" (NFL) theorems for optimization [162] show that for general-purpose algorithms, better performance over one class of problems compared to other algorithms implies worse performance over another class of problems. However, the theorems do not imply that specialized algorithms cannot be designed that perform better over a specific class of problems; indeed, they support the idea that specialization may be necessary to improve performance, and this chimes with lessons we learn from evolution in nature. This does not mean that general purpose algorithms are devalued; they often form the base of design for specialized algorithms, and can be used when the problem itself is not well enough understood to design a specialized algorithm, or when the time or resources to do so are not available. However, there are some problems that it is difficult or impossible for a general purpose algorithm even to process; the problem addressed in chapters 3 - 5 of this thesis appears to be one such problem. As we shall see, there is evidence that many types of data in different domains are self-affine in nature, and that in some domains such as finance there are sharp differences between different time periods so that time series can be divided into regimes based on covariance structure. However, the problem of how actually to divide a multivariate self-affine time series, possibly of high dimension, into such regimes has received little attention in the econometrics in the literature, even though many econometric models depend on the ability to identify such regimes as a prerequisite. As we shall also see, some useful contributions have been made elsewhere in the literature to the problem of optimally partitioning a multivariate time series, but not to the biobjective problem as considered in this thesis.

There are three main difficulties in developing an MOEA -based solution for such a biobjective multivariate partitioning problem: formulating the theoretical approach; ensuring that an adequate search is performed over a complex dataset, even where little if anything is known in particular about the self-affine patterns it contains; and designing an algorithm that can cope with these issues,

producing a reasonable result with finite computational resources, and ensuring that infeasible individuals do not overwhelm the population, making effective search impossible. When designing the representation and the genetic operators that govern the search by changing the representation from generation to generation, whilst theory may guide us on how the distribution of schemata and alleles within the population may change, the correlation between these changes and the fitness distribution may be harder or impossible to explain; instead, design is often guided by practicalities, especially when addressing a problem which is hard to compute and one is attempting to make the algorithm run faster and more reliably.

On the other hand, when designing or choosing a method of selection, at least in the single objective case there is a bedrock of theory to guide us on the likely effect on the development of the fitness distribution, with the considerable caveat that the interaction with the effect of genetic operators may remain unclear. The contribution of the work set out in Chapter 2 is to extend the understanding of the choice of selection scheme and of parameter values on the development of the fitness distribution in multiobjective problems, with several new results regarding limits to nonselection probabilities, asymptotic equivalence of different schemes, and time to convergence. These results inform the choice of selection scheme for the specialized MOEA described in subsequent chapters. However, as we shall see, specialization in design of the MOEA is by no means limited to the design of the tournament selection algorithm, though many other design choices for the specialized MOEA are driven as much by practical considerations, to ensure the MOEA runs efficiently and produces an acceptable set of feasible solutions.

The remainder of this introductory chapter is arranged as follows. Sections 1.1 and 1.2 provide background material on the history and development, typology and variations in structure of evolutionary algorithms (EAs). Section 1.3 considers some of the key trade-offs in MOEA design, and Section 1.4 comments briefly on parameter design and control, and finally Section 1.5 gives an

overview of the structure of the rest of this thesis.

1.1 What is an evolutionary algorithm?

In this section, we will provide a brief overview of how the modern landscape of EAs of many different types emerged as well as an introduction to key concepts in EA design, with particular reference to MOEAs. We will also discuss an important assumption underlying the development, use and study of EAs.

1.1.1 Early history and fundamentals of evolutionary algorithms

EAs form one branch of a broader family tree of algorithms that attempt to find solutions, by iterative, stochastic means, to problems for which there is no known efficient deterministic algorithm. EAs rely on selection and genetic operators to determine the search space, which may change as generations pass, and may be regarded as an evolving set of (possibly disconnected) subspaces within the overall feasible search space.

If we take EAs as the highest rank within one branch of the taxonomic order of algorithm types, its history as a field proper might begin in the late 1950s [8, 56, 61, 62, 22, 60]; in the 1960s, several sub-branches emerged in the development of EAs [57, 79, 141], which we will discuss further in Subsection 1.1.2. From the beginning, it was common for leading researchers to make contributions to both theory and practice; for example, Holland worked on the basic theory of EAs [78] and developed early genetic algorithms (GAs), amongst many contributions, whilst Bremermann developed one of the earliest practical applications - to the solution of nonlinear simultaneous equations [24] - and also proved one of the earliest theorems, on the optimal mutation rate [25]. It is worth noting that pioneering scholars often straddled the fields of computer science and mathematical biology, yet had come originally from completely different fields, as is still the case today.

An important and recurring concept throughout this thesis will be the distinction in EAs between representation space and fitness space. A representation is simply whatever form is given to an individual in a population for the purposes of computation; in earliest work, this was generally a binary string, but it could take any form that can be stored and which can be translated into a rank, score or set of scores via a fitness function. If the representation has ℓ distinct elements, then the total representation space will be of size:

$$\Lambda(\ell) = \prod_{i=1}^{\ell} \lambda_i, \tag{1.1}$$

where the λ_i represent the number of possible states for each element. If any element has infinite states then $\Lambda(\ell) = \infty$, and if all the elements are continuous, real numbers, then the representation space is a subspace or union of subspaces of $\mathbb{R}^\ell$. However, the elements of the representation space may be continuous or discrete, bounded or unbounded, and do not necessarily have to be numbers at all - they could be representations of executable code in the case of genetic programming (GP), or conceivably many other things, as long as they can be stored in computer memory, and processed by an EA.

A fitness space, on the other hand, depends on the ability of some fitness function to map any feasible representation into a fitness score or rank. A rank is always a single integer value $\in [1, n]$, where n here represents the maximum feasible number of ranks (which may be infinite), whilst a score may consist of one or, in the multiobjective case, of more than one number, each of which may be discrete or continuous and may have a different range. In the context of MOEAs, we may refer to these values as objective values, and if all these ω values are continuous, real numbers, then the representation space is a subspace or union of subspaces of $\mathbb{R}^\omega$. A fitness function is a mapping between representation space and fitness space, in the sense that any feasible individual - that is to say, an individual that obeys all restrictions and constraints - should always map to a valid point in fitness space, but typically we cannot say much more about

this mapping; it could be one-one or many-one, points that are close in a given region of representation space may or may not be close in fitness space, and there may be great variation between regions. If this is not so, and changes in the representation produce very predictable changes in fitness, it is likely that some simpler, possibly deterministic algorithm can be used to find optima, which is always the point of an EA even if the particular algorithm is not explicitly presented as solving an optimization problem. In still other cases, if the problem is simple enough and enough computing resource is available, a purely stochastic approach such as a Monte Carlo simulation (MC) may also suffice. For example, when generating the simulated data used for testing in Chapter 5 in such a way that different data series were sufficiently correlated, an MC approach was used; had it been harder to find the correlated series, an EA might have been used instead.

Hence it is important to introduce at this stage an assumption that underpins much of the rest of this thesis; namely that knowledge even of many points in fitness space as they correspond to individuals in representation space will not lead us easily to global optima, and that this is precisely why we are using EAs in the first place. This assumption has many important implications, of which we have already touched upon two, namely: the inability to predict the fitness of an individual from that of others in its neighbourhood in representation space; and the inability to easily find global optima from a partial set of representation-fitness correspondences . A third is that the use of genetic operators makes it hard if not impossible to predict with any precision the change, from generation to generation, in the fitness distribution of the population, that is, the number of individuals of a given fitness value or in a given fitness rank. This has consequences inter alia for the predictability of convergence time (if the algorithm is convergent) and the likely optimality of the final solution or solution set upon termination of the EA. At best we can generally speak only of correlations between fitness space and representation space, and such correlations are likely to vary across these spaces. However, if we consider selection

pressure alone, we can in fact make some predictions, as we shall see in Chapter 2.

1.1.2 Further development and typology of EAs

Throughout the 1960s and 1970s several distinct branches had emerged in the development of EAs, and we will discuss three broad types; GAs, genetic programming (GP), and evolutionary strategies (ES) .

1.1.2.1 Genetic algorithms

Holland and his research group developed both theory and practice of GAs throughout the 1960s and 1970s, leading to publication of the influential 1975 book *Adaptation in Natural and Artificial Systems* [80]. GAs are discussed in more depth in Chapter 4, as the specialized MOEA discussed can be categorized as a multiobjective GA. However, it is worth mentioning Holland's development of the theory of schemata. GAs in general display their inspiration from genetics more explicitly perhaps than other types of EA, and it is common to refer to the data in the representation of an individual as a chromosome, a single element within a representation as a gene and the value ascribed to a particular individual as an allele. A schema, as originally envisaged, is a binary string forming a section of a chromosome, for example 1*010**0, where the * symbol indicates the bit in that position can be either a 1 or a 0.

The underlying idea here is that individuals carrying a particular schema, or set of schemata, may tend to be fitter than others and hence by analysing such schemata, one may analyse or predict the evolution of the population in terms of the frequency of occurrence of the schemata in the population, as a function of the average fitness of individuals having such schemata and the influence of the probability of mutation and crossover. However, this type of analysis, and Holland's schema theorem in particular, have been shown to have a number of limitations and [3] showed that the schema theorem can be restated in terms of the covariance between fitness and the appearance of schemata, but that

even a GA with a fitness function that is in fact a random function (i.e. not related to the schemata at all) will obey the theorem, whilst also providing counterexamples of types of fitness landscape in which GAs will not perform in accordance with this type of analysis. An alternative approach is to look directly at the evolution of the fitness distribution directly, as influenced by selection; this is the approach taken in Chapter 2.

1.1.2.2 Genetic programming

Also referred to as evolutionary programming, the technique was devised by Fogel [57] as early as 1960, though the original concept has been ascribed to Turing[157]. The main difference with Holland's GAs was that the representations were not fixed-length strings, but rather automata, originally referred to as finite-state machines, which in essence were themselves computer programmes; an early application was in prediction of nonstationary time series, and the fitness function compared the prediction of the next symbol in a series with the actual symbol.

Only mutation was used initially to introduce variation, but crossover was also introduced later, and latterly the only real differences perhaps between GP and GA are that in GP the representation does not have a fixed length, and that an additional step (the running of the programmes) is generally required to assess fitness. As a consequence, not only schemata and alleles but also genes and chromosomes can evolve within the representation space of GP.

In the 1990s, John Koza [94, 95, 96] showed that GP could solve extremely complex problems of a sort not usually tractable to classical optimization, such as the design of circuits, and developed new techniques such as the use of automatic functions, essentially reusable subprograms, within the evolved representations. Because the representation is not of fixed length, GP can be very flexible in the way that solutions evolve, but the solutions can also be very complex, meaning that they are hard to audit or interpret, and as the representation size grows, operations such as calculation of the fitness function can become very

slow, so that automatic pruning of the representation to remove redundant code can become essential. It has been proposed that fitness based selection leads to bloat [100].

1.1.2.3 Evolutionary strategies

ES, originally Evolutionsstrategien in German, were first developed in the mid-1960s [8, 15] as a form of stochastic iterative optimization procedure using either only mutation for variation, or using crossover governed by a mixing number $\rho > 1$ indicating the number of parents involved in producing one offspring, in contrast to GAs where typically two parents produce two offspring [65]. The first form [141], later known as a (1+1)ES, featured a constant population of just 1 individual, mutated randomly using discrete, binomially distributed variables centred on current values for the representation; later research used Gaussian mutation for real-valued numbers. The natural extensions which were subsequently considered were:

- the $(\mu + 1)$ES, which used a steady-state model with $\mu \geq 1$ parents and a single offspring;
- the $(\mu + \lambda)$ES, with $\mu \geq 1$ parents producing $\lambda \geq 1$ offspring;
- the (μ, λ)ES, in which selection takes place only between the $\lambda \geq 1$ offspring, and the μ parents are forgotten, regardless of fitness.

A distinguishing feature of ES from the early days is the mix of exogenous parameters such as μ and λ which are kept fixed over a run and endogenous parameters used for example to control the mutation operator which are subject to self-adaptation, changing during the run. In the case of a (1+1)ES, it was found that tuning mutation to achieve a $1/5$ success rate, that is, the rate at which offspring replace parents, is optimal but this is subject to certain restrictions and is not generally true for other types of ES. Hence it was proposed that endogenous parameters adapt, in particular by various methods of collective learning across a population.

1.1.2.4 Limitations of typology

From a historical point of view, the differences between types of EA sometimes seem to have more to do with the approaches of the groups of researchers who initially devised them than actual distinctions, theoretical or practical, between algorithms, and lines between types of algorithm can easily become blurred. This is even more so when one considers the associations with and influence between the larger group of heuristic and metaheuristic search [16] techniques, including not only EAs but neural networks, particle swarm optimization (PSO), and other nature-inspired algorithms [145, 54] including ant colony optimization, plant propagation algorithms, and many more. The design of algorithms is itself arguably an evolutionary process, where new types constantly emerge and others die out, or at least lose popularity and fall into disuse, so that clear distinctions within the state-of-the-art at one time become less clear as new algorithms that take and improve on elements of existing ones constantly appear. For this reason, generic reference is made to EAs rather than specific types through out this thesis, and in particular the algorithm described in Chapter 4 is referred to as a specialized MOEA in line with accepted nomenclature, although it fulfils the requirements to be identified more specifically as a multiobjective GA.

Another objection to the use of terms like 'genetic' is that the relation of the operation of algorithms to genetic or other natural structures and processes is not strict and certainly highly simplified, and can also imply that he design is based on a reductive simulation of natural processes, rather than sound mathematical foundations. A further objection is that some elements of algorithms are scarcely if at all based on nature. Consider for example tournament selection; whilst it is true that in the natural world, some species see competition for mates in the form of actual physical contests, this is not generally the case and even when it is, forms only part of the selection process. Tournament selection is, rather, a convenient and robust method of selection which, as we shall see in Chapter 2, is at least somewhat tractable to mathematical analysis.

Algorithm 1.1 Structure of a generic EA

1. **Initialization:** create the initial population in terms of a representation, by random or other means;
2. **Iteration:** at each generation:
   (a) **Fitness evaluation:** assign fitness value(s) or rank to each individual;
   (b) **Selection:** assign current individuals to be kept as is (cloned), destroyed/forgotten, or passed to one or more genetic operators;
   (c) **Crossover:** recombine two or more parents to create one or more offspring;
   (d) **Mutation:** change one or more values in the representation randomly;
   (e) **Loop to 2** until stopping condition(s) satisfied;
3. **Return the solution** or set of solutions.

1.2 Generic structure of EAs and variations in structure

We shall return to the structure of EAs several times in this thesis, but Algorithm 1.1 presents a generic EA structure. Full commentary on each of these phases is beyond the scope of this work, but brief comments on each follows.

1.2.1 Initialization

Initialization creates the individuals forming the initial population of size N, according to the specified representation, which is typically a string of binary, integer or real values but could take any computable and storable form, including text, symbols and many other forms. It is generally the case that initialization is random, but it would also be possible to use some determined or pre-existing data set. If there are constraints or other conditions on feasibility of individual representations, it may be required that these are fully complied with by the initial population, partially complied with inside certain bounds, or it may be accepted in the algorithm design that there will be some proportion of infeasible individuals that will be discarded through selection. Some distribution is inev-

itably imposed on the randomly generated individuals, typically either uniform in the case of discrete values and normal or lognormal in the case of continuous values, but other distributions can also be used. The initial population size can vary from a single individual, as in the case of a (1+1)ES, to a population of millions, and can be held in a single, unitary block or in groups, typically based on restrictions or different centroids for the randomly generated values, as in island-type models such as the one used in the MOEA described in Chapter 4. Initial representations in GP are very varied compared to other types of EA but generally equate in some sense to computer code, decision trees or similar logical structures, or data structures [99] and may be small compared to the size reached in subsequent iterations of the algorithm, as well as varying in size from individual to individual.

1.2.2 Fitness evaluation

Fitness evaluation is often the most computationally intensive part of an EA, and as such it is a great advantage that usually these values can be calculated in parallel, and possibly by advanced means such as the use of graphics processor unit (GPU) computing[1]. A disadvantage of using a rank-based approach is that in addition to requiring an extra calculation step, as a form of sorting the complexity is likely to be higher than linear, unlike that of simple fitness evaluation which is linear in the population size N. Furthermore, the calculation of a ranking is inherently sequential in form, meaning that even if an EA is implemented in parallel, a number of processor cores may be idle whilst the ranking is found or updated. The difficulty of the actual calculation of the values depends usually on the complexity of the fitness function, and researchers may choose a simpler function form to speed calculation, possibly at some cost to the usefulness of the solution(s) found. In the case of GP, a further step may be necessary to evaluate fitness; for example, if the representation is a mathematical function, then that function must be evaluated over some set of data, and then

[1] Applications of GPU computing to EAs is discussed in the Appendix.

the fitness function applied to the output values. A further complication is that the tendency towards code bloat may also lead to an increase in time required to calculate the fitness values over a run.

All EAs need some way, typically an explicit fitness function, of mapping individuals in representation space to points in fitness space. In single-objective optimization, the points are often identified and located by a single, typically integer or real-valued number, and these fitness values may be unique to each distinct possible representation by construction or by virtue of being continuous, or it may be possible for more than one representation to have a particular fitness value, and of course identical copies of an individual will always have the same fitness value. It is generally not possible, or desirable, for a given representation to have more than one different fitness value. In the case of MOEAs, several distinct fitness values are calculated for each individual, so that Pareto fronts (PFs) may be formed; these may be passed directly to selection, or used to calculate a ranking consisting of some number of ranks assigned by testing nondominance or by some other means, as discussed in more depth in Chapter 2.

Design of the (possibly vector-valued, i.e. multiobjective) fitness function is critical to the success of an EA. If the fitness function does not accurately reflect the actual utility of the solution set in solving the problem being addressed, then even if the true global optimum is found in terms of the fitness function used, the actual result will be of little or no use. One example of this might be a fitness function that seems apposite conceptually, but in practice produces identical outputs for many inputs from the dataset used to assess fitness, making it difficult to distinguish fitness levels between different individuals. Hence fitness functions, in addition to being designed specifically for a given problem, may have to take into account the actual data used. However, it is often possible to change fitness functions without changing any of the rest of an EA, such that the fitness function can even be regarded as an input to, or parameter of, the EA.

1.2.3 Selection

The selection phase of an EA can decide any or all of: which individuals are passed to genetic operators; which are kept unaltered in the population; which are archived but not kept in the population; and which are forgotten/destroyed. Methods of selection include [66, 20]:

- **Proportional selection** - the original method proposed by Holland [80], under which the selection probability is simply proportional to normalized fitness;
- **Truncation selection** - individuals are sorted by fitness and only a top fraction can be randomly selected, with equal probability [130];
- **Linear and exponential ranking**- individuals are sorted by fitness and assigned a linearly or exponentially calculated selection probability based on rank[161];
- **Tournament selection** - fitness or rank of two or more individuals are compared in a series of tournaments using some random scheme until the required number of individuals is found and passed to genetic operators [65].

Tournament selection is considered the most popular form of selection and is considered in more depth in Chapter 2, as well as being implemented in an unusual form (permuted multiobjective tournament selection) in Chapter 4. Tournament selection can be implemented in many forms, including:

- with or without replacement, or using permutations;
- single objective, rank-based, or multiobjective;
- positive or negative (negative tournaments select less fit individuals for deletion);
- binary or higher arity;

- implemented in series or parallel;
- steady state or generational.

Choice of selection method will depend on the desired statistical properties, including the effect of the resultant fitness distribution, and on consideration of computation time and method, in particular whether and how parallel processing is implemented. There have been many claims and counterclaims about which method of selection is best, but as with most other decisions taken in the design of an EA, there are multiple factors to be considered; the choice depends on or has consequences for other choices made in design, and there are often trade-offs, for example between achievement of desired statistical properties and computation time. Some of these tradeoffs are considered further in Section 1.3 below.

Whilst the concept of selection is inspired by evolutionary processes in nature, the real processes involved in selection amongst living organisms are complex, and selection algorithms either greatly simplify, or in some cases, including tournament selection, arguably largely depart from phenomena observed in natural selection. Additional features not generally observed in nature include elitism, in which individuals of high fitness or rank are automatically passed to genetic operators, and archiving, where the fittest individuals from any generation are recorded such that they can form part of the final solution set if fitter individuals are not found, but do not necessarily remain in the population.

1.2.4 Crossover

The classical form of crossover involves choosing two parents from those in the population after selection and recombining them twice to produce two offspring, specifically by selecting some locus in the representation string and swapping the substrings after that point, producing two offspring. However, many variations are possible; generic types for algorithms with fixed-length representations include [93]:

- **Single point crossover**, as described above;
- **two point crossover**, where only the substring between two points is swapped;
- $k-$**point crossover**, where alternating substrings between $k > 1$ points are swapped;
- **uniform crossover**, where each point has a random chance of being swapped.

There are variations even to these generic schemes; for example in $k-$point crossover, k can be fixed or random, and in uniform crossover, the swap probability can be set in many ways. Many other generic schemes exist, including ones that create only one offspring such as average crossover and flat crossover; schemes also exist that use more than two parents.

However, such generic schemes are by no means suitable for all problems, depending on the constraints and requirements for feasibility of offspring; some number of infeasible offspring may be acceptable if a generic scheme saves computation time, but if the number is too large, a more specialized scheme may be required. A standard example is the generic travelling salesman problem (TSP), where a requirement for feasibility is that representations are a complete permutation of all the cities to be visited. Now, any swap of substrings between parents that replicates cities and/or misses cities out will render offspring infeasible. Many solutions have been proposed; one early example is the partially-mapped crossover (PMX) [67], where a substring is determined by two crossover points in each parent and passed to two offspring with position preserved, but then the mapping between the two substrings is used to swap out repeated elements. However, many problems may require some adaptation of existing schemes, or even a completely new scheme; the specialized MOEA described in Chapter 4 required development of a new crossover type.

In GP, crossover is similar in concept but takes different forms because the representation size is not fixed. The most common generic type is subtree

mutation, in which, assuming the representation can be viewed as a tree, a mutation point is selected in each of two parents and the subtrees below that point swapped to produce two offspring. Many other types of GP crossover have been proposed, often, but not always, analogues of the types described above for fixed size representations, and again, specialized schemes have been developed to address specific problem types.

If we think in terms of alleles and schemata, crossover can never create new alleles, and in many schemes where two parents create two offspring, alleles cannot be destroyed either, though this is not true in all schemes; for example, schemes which involve the production of a single offspring may not guarantee preservation of alleles. Schemata can be created or destroyed in most if not all crossover schemes. If we put these two observations together, then we see that in most two-parent, two-offspring cases, the range of available schemata is not changed by crossover, but the schemata actually present in the population is changed. Another way to put this is that the total representation space does not change by the action of two-parent, two-offspring crossover, but the union of subspaces which contains all possible offspring after crossover does change. This is the mechanism by which crossover controls the search, or more accurately, the search space, though it is selection that leads directly to changes in the fitness distribution.

1.2.5 Mutation

Picking up from the previous discussion on crossover, mutation can both create and destroy alleles, and hence can also create and destroy schemata; however, compared to crossover, which depending on form can create offspring which are quite different to either parent, mutation is generally implemented in a way that produces only small and local changes to an individual, though as ever in the broad field of EAs, there are exceptions to this. As with crossover, there are many forms, of both general and specialized application. Generic forms of mutation for fixed-length representations include[138, 101]:

- **bit-flip mutation** - the most basic form of mutation, where there is a probability (for example $1/l$, where l is the length of the binary representation string) that each 0 becomes a 1 or vice versa; a separate probability may be applied that any mutation occurs for a particular individual;

- **random mutation** - for integer or real-valued representations, a random variable is added to the existing value at one or more loci, according to some distribution (for example, binomial or normal) and possibly constrained to be within a certain range, to ensure variation is not too large and/or feasibility is guaranteed;

- **swap mutation** - values at two loci are interchanged. This means the set of values within the representation are unchanged, potentially making it suitable for permutation-based representations, as used for example in TSP algorithms;

- **inversion mutation** - reverses the order of a substring, so also suitable for permutations.

For GP, as with crossover, there exist analogues of many such techniques as well as more specialized mutation schemes. Subtree mutation [95] substitutes a randomly-generated subtree for a randomly determined subtree in the existing representation. The GP form of point mutation replaces a particular node with another primitive of the same arity [138].

Mutation rates are generally set quite low, though they may be adaptive, especially in ES. The low fixed or starting probabilities are employed to limit destruction of useful alleles/schemata, i.e. ones with a high correlation to better fitness.

1.2.6 Cloning, archiving and elitism

If an EA allows some probability that an individual is subject to neither crossover nor mutation, then that individual is considered to be cloned in the next

generation; cloning is also sometimes referred to as reproduction. Depending on the distribution and any restrictions on the probabilistic application of genetic operators, any individual could be cloned, including one of low fitness or rank. On the other hand, archiving refers to the recording of the optimal individual or set of individuals; archived individuals are not kept in the population, but if they are not superseded in fitness/rank at the end of the run, form all or part of the final solution set.

Elitism is explored in more detail in Chapter 2, with particular reference to its effect on selection pressure. The idea is that all individuals of a certain fitness level or rank and above are automatically selected, and passed to genetic operators; however it is possible that they could be passed unchanged, either because they happen not to have crossover or mutation applied, i.e. cloning is applied instead as described above, or because they are cloned automatically, or with a different probability to other individuals. Elitism and archiving are not mutually exclusive; an EA may employ both. Elitism may also have other, possibly beneficial, effects such as reduction in code bloat in GPs [139].

In the case of the specialized MOEA described in Chapter 4, archiving is deployed, but elitism is not, because the permuted form of selection renders it unnecessary. Crossover is applied to all individuals in a permuted fashion, i.e. every selected individual becomes a parent and there is no cloning; the actual form of crossover is highly specialized, because of the difficulties in maintaining feasibility of the offspring. Point mutation is applied to offspring in a more conventional manner, with a relatively low starting probability, and choice of a mutation probability always balances the benefits to diversity in representation with the need to preserve alleles and schemata correlated with higher fitness.

1.3 Trade-offs in EA design

The design of EAs involves many significant trade-offs, which may be fixed in some sense or may be controlled by one or more parameters used by the EA.

We briefly discuss here three of the main ones: accuracy versus computational resources required; fitness versus diversity; and accuracy versus robustness.

1.3.1 The computational complexity trade-off

When brute force methods which guarantee finding the global optimum are too computationally expensive and we wish to avoid use of greedy algorithms which may find only a local optimum, we often use iterative methods such as EAs which although they may not find the true global optimum within a limited time frame given finite computational resources, will find a useful approximation depending on the computational resources available. Furthermore, EAs are often designed so that the solution at each generation is at least as good as that provided at the previous generation, and so the top level problem common to all such procedures may be stated as:

Minimize

$$z = \frac{T_C}{C_T} \tag{1.2}$$

subject to

$$\mathbf{f}(\boldsymbol{\Omega}) \leq \boldsymbol{\theta}. \tag{1.3}$$

Here, T_C is the computational cost for the algorithm; this could be measured in computation time if resources are fixed, proportion of resources used if resources are shared, or even a monetary cost if for example resources are paid for on a time-per-node basis. C_T is the value of the solution according to some measure, which might be a theoretical measure of progress, or else could be measured in more concrete ways, for example as a cost saving in a logistics problem or a trading profit in a financial application. The set of constraint parameters $\boldsymbol{\Omega}$ is a subset of the parameters used in Equation 1.2; $\mathbf{f}(\boldsymbol{\Omega})$ is often nonlinear (and may be combinatorial). This naturally includes all of the constraints used in the underlying optimization problem, but in addition could

include parameters which have upper bounds set in the vector of constants $\boldsymbol{\theta}$ governing not only cost or time or share of resources but also available memory, for example, including aspects such as the maximum size of individual arrays or maximum total size of all stored arrays. These parameters, which depending on algorithm design may also include population size or number of runs or number of generations, may be included in both the numerator and denominator of Equation 1.2.

In practice, we may not be able to calculate z a priori from analysis of the algorithm and computing resources, but may need to rely on empirical estimation. We will return to these issues as we examine the structure of the specialized MOEA described in Chapter 4.

1.3.2 Accuracy and available resources

In the context of evolutionary algorithms, accuracy is taken to mean the proximity, by some measure, of the interim or final solution or solution set to the optimal solution or solution set. Many ways to measure accuracy can be imagined [64], and if there is a known global optimum, this is simply a matter of choosing a suitable distance metric or similar measure. For instances where the global optimum is not known, or possibly not knowable, for example in Chapter 5 , two specialized metrics are used to test the dissimilarity of the coarse-grained regimes from each other, and the similarity of the fine-grained to the coarse-grained partitions; hence in this case, the testing is effectively biobjective, like the partitioning problem itself.

It is often stated that the computation cost of EAs compared to other types of algorithms is high, and to justify this, it should also be that the value of the solution is high; this can be particularly noticeable in multiobjective problems, where there can be many local optima, making the global optimum potentially very hard to find. EAs do not in general guarantee that the global optimum is found, but may have a higher probability of doing so, or else at least of finding a better local optimum (or set of local optima).

1.3.3 Fitness and diversity

Unlike the trade-off between accuracy and available resources, which refers principally to the final result produced by an EA, the trade-off between fitness and diversity affects both the representation and fitness space from generation to generation during the run. The most obvious aspect of the tradeoff is the number of copies of individuals in the population, or to put it another way, the number of unique individuals in the population. For example, as we shall see in Chapter 2, in a crowding-out type convergent EA, in which copies of elite individuals propagate until eventually they crowd out all other individuals, under many tournament selection schemes with large populations, the number of copies of the highest fitness individual in a population will increase at each generation by a factor approaching the tournament size. This increases the speed of convergence, lowering T_C, but also lowers the probability of finding a better solution during the run, in several ways, depending on the operation of the EA. If elite individuals stay within the population, they will quickly crowd out all other individuals, leading potentially to premature convergence, that is, convergence to a local optimum before the representation space is adequately mapped to the fitness space, so that the global optimum is never found. More generally, proliferation of offspring of closely related individuals may lead to less diversity of alleles and/or schemata, meaning less of the representation space is searched through recombination; mutation may only partially offset this, depending on the mutation rate and how mutation operates.

A detailed analysis of diversity is beyond the scope of this work, but diversity is analysed extensively in the literature, both as part of schema theory and other attempts to analyse changes in representation space, and in mathematical analysis of selection on the fitness space [78, 80]. In developing the specialized MOEA, the decision was made to emphasize and control diversity in two main ways. Firstly, the population was divided into separately processed subgroups, following one of numerous forms of parallel genetic algorithm (PGA) [132] , and the deliberate segregation of the population ensures a degree of diversity for the

overall population throughout a run, even if the local population in a particular subgroup is potentially less diverse, through lack of other alleles and schemata present in other island subgroups. In particular, the island structure ensures a wider range of alleles and schemata overall survives the whole run. Secondly, the algorithm is deliberately nonconvergent; because the crossover rate is set at 1, and explicit elitism is not employed, no individuals stay in the population from generation to generation, and all individuals at each generation are new.

1.3.4 Accuracy and robustness

It is generally the aim of any optimization process, whether by EA or otherwise, to discover the true, global optimum, though in some cases, such as the specialized EA described in Chapter 4, it is implicitly accepted that discovery of the true optimum is unlikely, and the aim of the EA is rather to discover a set of good approximations. The more specialized the EA is, the better it may be able to provide good solutions for a particular type of problem, but may be less suited, or entirely unable to find solutions for a different type of problem.

It is also important that an EA be robust to changes in the type of problem, within the range of problems for which the EA is designed, or to changes in any data inputs. In other words, even if an EA is to a greater or lesser extent specialized in nature, within the domain for which it is designed, although the NFL theorems suggest here will be some variation in performance depending on the problem, the EA should not have variations in performance that are unacceptably large depending on problem specification; if it does, this could be an argument either for redesign to make the algorithm more robust, or for further specialization.

1.4 Parameter design and control

The choice of parameter type, parameter form and method of control have been recognized as critical to the performance of EAs [47, 89, 90]. It is not necessarily

easy to separate good or poor performance of an EA, or compare performance of EAs, as applied to different problems, data sets or time periods, between good design on the one hand and judicious or fortuitous choice of parameter values on the other. This problem becomes harder as the number of choice parameters increases, and EAs with more than a few parameters may be difficult to adequately test, as well as being intractable to theoretical analysis.

One approach is therefore to make the EA parameterless, or as close as possible to having no parameters [77, 75]. In fact, 'parameterless' is often a misnomer, as there are several ways to reduce or eliminate choice parameters, including inter alia:

- fix the parameter in question at a particular value for any run;
- decide the parameter in terms of a formula involving other parameters;
- vary the parameter over different runs, or across population subgroups run in parallel;
- adapt the parameter value over different runs.

The specialized MOEA described in Chapter 4 uses a combination of these approaches to reduce the number of choice parameters, and other approaches may also prove useful. All can however be criticized for potential drawbacks. Fixing the parameter can only be done using strong assumptions. Using a formula for a parameter passes off responsibility for performance to other parameters, and may make performance analysis trickier. Varying the parameter of multiple runs in effect nests an EA inside a Monte Carlo simulation, and is quite computationally inefficient, though it can be effective in providing information about parameter sensitivity at the same time as searching for solutions. Adapting parameters can be more efficient, but runs the risk of overfitting and can make it even more difficult to separate the effect of parameters from general EA design when analysing performance.

Finally, insofar that choice parameters may be unavoidable to some extent, parameter tuning [47, 48, 46] can be extremely important in getting good results

from an EA, and this process can to some extent be optimized and automated. However, if very different results are produced from small changes to parameters, this may be an indication that the EA tends to produce a similar effect to overfitting in a standard statistical model. This may suggest that the solution found is unlikely to be robust to other changes, or across different runs, making results hard to interpret or rely on, and raises the question of whether the EA is doing a good job or if there is merely a fortuitous matching of parameters to the particular testing method chosen.

1.5 Structure of this thesis

The thesis hereafter is structured as follows. Chapter 2 is the longest chapter and lays important theoretical groundwork for the choice of selection scheme used in EAs, with particular reference to different schemes used for multiobjective tournament selection, in part using the device of Pareto rankings. The analysis includes schemes with and without replacement and permuted schemes, and the particular advantages of permuted schemes are highlighted. The important role of the tournament size parameter τ is discussed, and conclusions drawn regarding its importance in determining nonsampling probability, post-tournament rank or fitness distribution, speed of increase of elites in the population, and speed of convergence. The effect of elitism on the development of the rank or fitness distribution is also discussed. Several new theorems are developed.

Chapter 3 continues the theoretical discussion for the subsequent development of the specialized partitioning MOEA by looking at the background to partitioning time series of possibly high dimension, and at the concept of self-affinity; some important statistical models are discussed. An optimization model is then developed for the purpose of partitioning multivariate self-affine time series. Chapter 4 describes the specialized MOEA, with particular reference to the trade-offs inherent in design and the specialized nature of the MOEA. It explains the design choices made for each phase of the specialized MOEA, in-

cluding a section on the mathematical implications of the choice of tournament selection scheme, which links to the discussions in Chapter 2; several other important areas of specialization are also discussed. Chapter 5 describes the methodology for testing the specialized MOEA using simulated and real data, and reports results. This is followed by concluding remarks, including discussion of possible directions for future research.

Additional information on computation, in particular use of parallel processing, is contained in Appendix A.

Chapter 2

The Mathematics of Pareto Rankings

2.1 Introduction

Ranking schemes are a useful and widely used tool to reduce multiobjective Pareto fronts to a single statistic that can then be used efficiently in selection within an MOEA. Selection has long been recognized as the central process in EAs, whether single-objective or multiobjective, by which the fitness distribution of the population at each generation is improved [66], and binary tournament selection with replacement is described earlier in [26]. Tournament selection has been widely used because of desirable properties, including simplicity of programming, relative tractability to theoretical examination, and the ability to the ability to tune selection pressure through choice of tournament size; many comparisons have been made between tournament selection and other methods of selection [65, 20, 18, 34, 103, 81, 76]. Some types of tournament selection also have the benefit that it can be implemented in parallel, potentially reducing the computation cost, though this will depend on the overall architecture of the EA. Although binary tournaments, which emphasize maintenance

of diversity, are often the default choice in algorithm design, selection pressure can be effectively tuned by varying the number of participants in each tournament, though as we explain at various points in this chapter, the choice of this parameter value affects the outcome in several different ways.

Notable work has been done sporadically on the mathematics of single-objective tournament selection since the technique was first adopted, though relatively little has appeared in the literature in recent years. Explicit analysis of multiobjective tournament selection has been limited, and in both the single objective case, there has been little written regarding the effects of the use of selection without replacement and elitism on the distribution of the resultant population once the tournament selection phase of an algorithm is complete. The main contribution of this chapter is to extend theoretical analysis of multiobjective tournament selection, using a nondominated sorting scheme as a framework, though much of the analysis is valid even when an MOEA does not explicitly calculate a Pareto ranking, as considered in Chapter 4.

We start by considering the mathematical implications of the use of Pareto rankings in tournament selection for multiobjective problems, in particular those produced by nondominated sorting, and show that a similar methodology to that used for single objective problems can be applied to derive the expected population by rank. We state a set of axioms governing such rankings, and a set of propositions about the nature of ranking systems using the axioms. Then, building on the methodology put forward in [19] to predict the expected fitness distribution in the single objective case at each generation for a given population after tournament selection, we extend the analysis to the multiobjective case using Pareto rankings. Most of the theoretical literature, including inter alia [65, 66, 9, 19, 126, 8, 148, 129, 137, 164], assumes use of selection without replacement, with few references to alternatives, even though use of selection with replacement has been considered since the early development of the technique. The effect of elitism and archiving are also largely overlooked;, these subjects will also be examined in the context of rank-based multiobjective

tournament selection. The prevalence of selection with replacement is perhaps explained by ease of implementation, or because of assumptions made that in large populations, there will be little difference in outcome between selection with and without replacement, or that use of elitism will obviate any potential problems. We will examine the validity of such assumptions later in his chapter.

A general form for a limit under multiobjective selection with replacement in rank-based tournaments is derived, which implies that no matter how large the population, elite individuals are at risk of not being selected because they are never sampled, unless algorithms are specifically designed to counter this. The limit also applies to at least one form of selection without replacement, but not to a completely permuted scheme, which guarantees all individuals are sampled for at least one tournament, and hence that elite individuals are always selected. This limit can apply to any number of ranks starting from the first rank, and is approached from below, so that the nonsampling risk grows with population size, up to the limit. Asymptotic equivalence of the expected rank sizes between different types of tournament is also demonstrated in some cases, and the effect of elitism on selection is examined. An expression is derived that links tournament size to the growth rate of the elite cohort, and shows that in large populations, the elite cohort will increase in size at each generation by a factor approaching the tournament size. A limiting expression is also derived for the expected number of generations to convergence. Whilst this formulation ignores the effect of genetic operators or other features of algorithms, it may serve as a useful check for algorithm development.

The remainder of this chapter is arranged as follows. In Section 2.2 the mathematics of multiobjective tournament using Pareto rankings are examined, in order to illuminate the workings of ranking-based tournament selection, including in relation to speed of convergence. In Section 2.3, the mathematics of tournament selection without replacement are reviewed, the inherent problem of nonselection of elite individuals is examined, and a general limiting expression for the nonselection probability is derived. In Section 2.4 the distributional

properties of selection with and without replacement are compared, and several conclusions follow.

2.2 Pareto ranking systems

2.2.1 Ranking algorithms

Ranking systems have long been used to simplify multiobjective problems in operational research, decision science and other areas, as a way to deal with multiple valid solutions a posteriori. MOEAs form just one part of the methods of multi-criteria decision analysis (MCDA), and in other areas of MCDA, ranking methods are generally used to choose between valid solutions once the Pareto front (PF), or some other representation of multiple optimal solutions, is obtained, and many methods exist to do so [32, 155, 88, 111, 59, 86]. Within MOEAs, ranking methods are rather used to help determine fitness at each generation whilst the MOEA is running. For the purpose of fitness comparison, ranking methods effectively reduce a multiobjective problem to a single fitness statistic, though as with single-objective methods, there must be a way to determine the outcome when there is a tie in a tournament.

To set the scene, for a population size N let there be $\tilde{N} \leq N$ unique vectors of fitness values $\mathbf{v}_j = [v_{1j}, v_{2j}, \ldots v_{\omega j}]^{\mathrm{T}}, j = 1 \ldots \tilde{N}$, with one value $v_{kj},\ k = 1 \ldots \omega$ for each of the ω objectives. Each individual in the population is represented by, and maps to, just one point in fitness space represented by such a vector. This is a many-one mapping from individuals in representation space to points in fitness space, as in general there may be more than one individual with identical fitness values in the population; these may be copies of the same individual, but we also account for the possibility that individuals with different representations have identical fitness values. Hence there may be fewer unique vectors than individuals in the population, and so $\tilde{N} \leq N$.

First, we define in the usual way vector $\mathbf{v}_i$ as Pareto dominant with respect

to individual $\mathbf{v}_j$ iff:

$$(v_{ki} \geq v_{kj} \; \forall k \in [1, \omega]) \quad \wedge \quad (\exists v_{ki} > v_{kj}) \,. \tag{2.1}$$

It follows that an individual is non-dominated iff:

$$(\exists v_{kj} > v_{ki}) \vee (v_{kj} \geq v_{ki} \; \forall k) \,, \tag{2.2}$$

and we can narrow the second condition to equality because of the existence of the first, so that the property of being non-dominated can also be defined as:

$$(\exists v_{kj} > v_{ki}) \quad \vee \quad (v_{kj} = v_{ki} \; \forall k) \,. \tag{2.3}$$

That is to say, an individual is non-dominated with respect to another if and only if it either has any objective value strictly greater than the corresponding value for the other individual, or all the fitness values are equal. In some algorithms, we may be able to dispense with the second condition if we know or can safely assume that all individuals are unique and at least one objective value will be different for each individual.

True multiobjective tournament selection can be complex both in theory and in practice when designing algorithms; we examine one such scheme in Chapter 4. As a result, the most common approach is to reduce the problem to a discrete, single objective one by using a nondominated sorting technique in which successive Pareto fronts are found, the nondominated individuals are assigned a rank number, stored and then removed from the population for the rest of the sort, and the next Pareto front found until all individuals have been assigned a rank $\mathbf{r}_i, \; i = 1 \ldots n, \; n \leq \tilde{N}$, with $\mathbf{r}_1$ the top rank. Note that the convention for the ordering of ranks is thus the reverse of that often used in the literature for the single objective case, and used above in Equations 2.1 - 2.3 when considering the ordering of individuals by fitness level; this ordering is more natural in relation to the operation of Algorithm 2.2, which finds the top

rank first, as well as in the context of our examination of the problem of elite nonsampling in Section 2.3.

Since the ranks are discrete and there can be more than one individual of the same rank in a given tournament, some further step is necessary in constant population algorithms to deal with ties. In the case of NSGA-II [44], for example, a tiebreaker score function is used, based on niching distance of the individual on the Pareto front, which may also increase diversity, at least in objective space. Tiebreaker functions are typically continuous, meaning they generally give a definitive winner for each tournament. Such ranking schemes are certainly convenient in that they guarantee a constant population, and the tiebreaker may add some useful properties to the selection process. However, rank sorting algorithms are inherently sequential and have high complexity with large populations; for this reason, some algorithms calculate only some fixed number of ranks, and consign all other individuals to a single bottom rank. In subsequent discussions, it is assumed that in the case of ties, the victor is selected at random using a uniform distribution.

The concept of a Pareto ranking is a natural extension of the idea of a Pareto front, and various ways of constructing them for MOEAs have been proposed since the early days of research in the area [65, 14, 58, 2]. The best known and most widely used type is the nondominated sorting technique used in NSGA-II and many other algorithms; non-dominated sorting techniques have been extensively studied and refined, calling on various areas of mathematics [98, 87, 97, 53, 27]. In an exhaustive algorithm such as Algorithm 2.2, the ranking is constructed by finding the PF for the population, removing the individuals on the PF and repeating until all individuals are assigned a rank. Note that if the algorithm only finds the first few ranks and the remainder of the population are assigned to a bottom rank, then should all individuals participating in a tournament happen to be of this bottom rank, the tiebreaker effectively becomes the sole measure by which selection is decided. Approximate methods for finding the ranking have also been proposed [158, 105], given the complexity of finding

Algorithm 2.1 fast Pareto front algorithm

Step 0: initialize O with the set of unique fitness function value vectors $\left\{\mathbf{v}_j = [v_{1j}, v_{2j}, \ldots v_{\omega j}], j = 1 \ldots \tilde{N}\right\}$ and set the objective counter $o = 1$, noting the map of all N individuals to the $\tilde{N} \leq N$ unique fitness vectors;
Step 1: whilst O is not empty:
Step 2: find and record the subset O^o of individuals that have the minimum value for the current objective o and remove the members of O^o from O;
Step 3: from these individuals, find the nondominated set $\hat{O}^o$ by applying the "two-way $>$" condition 2.4 and add these to the PF;
Step 4 find and eliminate all points in O that are set-dominated by $\hat{O}^o$ by applying the "one-way $>$" condition 2.5;
Step 5 : set $o = \mathrm{mod}\,(o, \omega) + 1$ and loop to Step 1;
Step 6: return the PF.

all the ranks for large populations and/or numbers of objectives. It is nondominated sorting as a method of Pareto ranking that we will investigate and use in the rest of this chapter.

The problem remains however that if finding a single PF can be computationally expensive, finding many - possibly thousands - at each generation is far more expensive still, and truncated or approximate methods may not be acceptable. A fast method to obtain a complete ranking from a population is introduced in Algorithm 2.1. The algorithm uses two different types of condition; a "two-way $>$" condition, namely:

$$(\exists v_{ki} > v_{kj}) \vee (\exists v_{lj} > v_{li})\, \forall \mathbf{v}_i, \mathbf{v}_j \in \hat{O}^o, i \neq j, k \neq l, \tag{2.4}$$

that is, for each pair of vectors in the nondominated set, each has at least one objective value strictly greater than the other; and a "one-way $>$" condition:

$$(\exists v_{ki} > v_{kj}) \vee \neg\,(\exists v_{lj} > v_{li}, l \neq k)\, \forall \mathbf{v}_i \in \hat{O}^o, \mathbf{v}_j \notin \hat{O}^o, l \neq k, \tag{2.5}$$

that is, for each pair of vectors, one in the nondominated set and the other not, the first has at least one objective value strictly greater than the second, but the reverse is not true. Equations 2.4 and 2.5 follow from the definitions of dominance and nondominance in Equations 2.1 and 2.3.

Algorithm 2.2 Exhaustive Pareto ranking algorithm

Step 0: initialize O with $\tilde{N}$ vectors of fitness values $\mathbf{v}_j$ and set the rank counter i to 1;
Step 1: find the PF for the current population by calling Algorithm 2.1, or an equivalent algorithm, add these individuals to rank $\mathbf{r}_i$ and eliminate them from O;
Step 2: increment i and loop to Step 1 whilst any points in the population remain;
Step 3: return the set of ranks $\left\{\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_n,\ n \leq \tilde{N}\right\}$.

Algorithm 2.2 summarizes the generic, exhaustive method to find each of n ranks one by one, and subsequent analysis is based on the assumption that rankings are found using this algorithm or one exactly equivalent in that it is guaranteed to always produce the same output for the same inputs. Note that the algorithm operates only on the $\tilde{N} \leq N$ unique vectors of fitness values, with these vectors mapped to the N actual individuals in the population, eliminating unnecessary computation. Faster methods such as updating [104] may be used to generate rankings, but for what follows to apply to any other method, it must be shown that the method is exactly equivalent to Algorithm 2.2, that is to say, that the methods will always produce exactly the same result as Algorithm 2.2.

2.2.2 Theory of ranks and rankings

This subsection develops theoretical arguments based on rankings produced by Algorithm 2.2. Application to results produced by other methods can be guaranteed if and only if those methods are exactly equivalent to this algorithm. Any algorithm that finds the nondominated points and conforms to the standard definition of Pareto dominance in Equation 2.1 may be called to find the PFs themselves; in addition to Algorithm 2.1, a fast algorithm for biobjective problems is presented later as Algorithm 4.3. Although what follows may be of more general application, it would be necessary to check the validity of definitions and propositions for algorithms using different methods.

2.2.2.1 Definitions and ranking lemmas

Let A be a finite set of unique points in fitness space, each represented by a vector of fitness values $\mathbf{v}_j,\ j = 1 \ldots \tilde{N}$, with each individual in the population mapped to exactly one point, and all individuals with identical fitness values mapped to the same point. A ranking $\Re = \left\{\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_n,\ n \leq \tilde{N}\right\}$, where $\mathbf{r}_1$ is the highest rank as generated by Algorithm 2.2 or an exactly equivalent algorithm, is a complete, non-overlapping partition of A, consisting of n ranks $\mathbf{r}_i$, each of which is a unique subset of A. Such a ranking $\Re$ has the following properties:

1. **Completeness** - $\cup_{i=1}^{n} \mathbf{r}_i = A$, and all individuals in the population map to some point in some rank in $\Re$;

2. **Non-overlapping ranks** - $\mathbf{r}_i \cap \mathbf{r}_j = \emptyset \ \ \forall i \neq j$ and all individuals with identical fitness values map to the same point in the same rank;

3. **Inter-rank nondominance** - each point $\in \mathbf{r}_i$ is nondominated with respect to all other points in the same rank, and hence all individuals mapped to points in the same rank are nondominated with respect to each other.

These properties follow directly from the method of construction set out in Algorithm 2.2; as such, we take them to be axiomatic in what follows.

Lemma 2.1. *Each point in rank* $\mathbf{r}_i$ *is nondominated with respect all points in* $\cup_{l=i}^{n} \mathbf{r}_l$.

Proof: If any point in $\mathbf{r}_l,\ l > i$ were to dominate some point in $\mathbf{r}_i$, then either Algorithm 2.2 would allocate that point to a higher rank than $\mathbf{r}_l$, or the dominated point would be allocated to a lower rank. For points of the same rank, the axiom of inter-rank nondominance applies.

Lemma 2.2. $\mathbf{v}_j \in \mathbf{r}_i$ *does not imply that* $\mathbf{v}_j$ *dominates any point in* $\cup_{l=i+1}^{n} \mathbf{r}_l$.

Proof: The points allocated to rank $\mathbf{r}_{i+1}$ must all be dominated by some point in $\mathbf{r}_i$, but this does not imply that a particular point $\mathbf{v}_j \in \mathbf{r}_i$ dominates any

point in $\mathbf{r}_{i+1}$, and because nondominance is not transitive, then by induction, $\mathbf{v}_j$ does not necessarily dominate any point in a lower rank.

Lemma 2.3. *The set of points in rank* $\mathbf{r}_i,\ i = 1 \ldots n-1$ *rank-dominates every set of points in any rank* $\mathbf{r}_{l>i}$*; that is every point in* $\mathbf{r}_l,\ l > i$ *is dominated by at least one point in* $\mathbf{r}_i$*, and no point in* $\mathbf{r}_i$ *is dominated by any point in a lower rank* $\mathbf{r}_{l>i}$*.*

Proof: The second part of the requirements for rank-dominance follows from Proposition 2.1. As to the first part, if any point in $\mathbf{r}_l$ were not dominated by some point in $\mathbf{r}_{l-1}$, the collection of points of one rank higher, that point would be allocated to a higher rank by Algorithm 2.2, not to $\mathbf{r}_l$, and because dominance is transitive, it must be that each point in $\mathbf{r}_l$ is dominated by some point in any rank $\mathbf{r}_i,\ i < l$.

Lemma 2.4. $\mathbf{v}_j$ *may dominate some point in* $\mathbf{r}_l$ *if and only if* $\mathbf{v}_j \in \mathbf{r}_{i<l}$*.*

Proof: This follows directly from the properties of inter-rank nondominance and Lemma 2.3.

Note however that together with Lemma 2.2, the implication is that although a given point cannot dominate another in the same or higher rank, it does not follow that the point necessarily dominates any other point in A.

Lemma 2.5. *For any set* A *of unique points* $\mathbf{v}_j$ *in fitness space, there is exactly one corresponding ranking* $\Re$*.*

Proof: Firstly note that a valid ranking always exists, since Algorithm 2.2 will always allocate all individuals to a rank. Then note that if more than one such ranking existed, then given a valid ranking $\Re$, we could move one or more points from one rank to another and produce another valid ranking. However, if we move any point $\mathbf{v}_j \in \mathbf{r}_i$ to a lower rank, it must be dominated by some point in its original rank, by Proposition 2.3; yet we know the point is nondominated with respect to all points in $\mathbf{r}_i$, a contradiction. Similarly by the same axiom, if we move the point to a higher rank, it must be nondominated with respect

to all other members of that rank, yet we know it is dominated by at least one point in each rank $\mathbf{r}_{l<i}$, because Pareto dominance is transitive. Hence each point can belong to only one rank, and so there can only be one, unique valid ranking.

Lemma 2.6. *A ranking* $\Re = \{\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_n\}$ *over a set* A *represents a strict total order over* A*.*

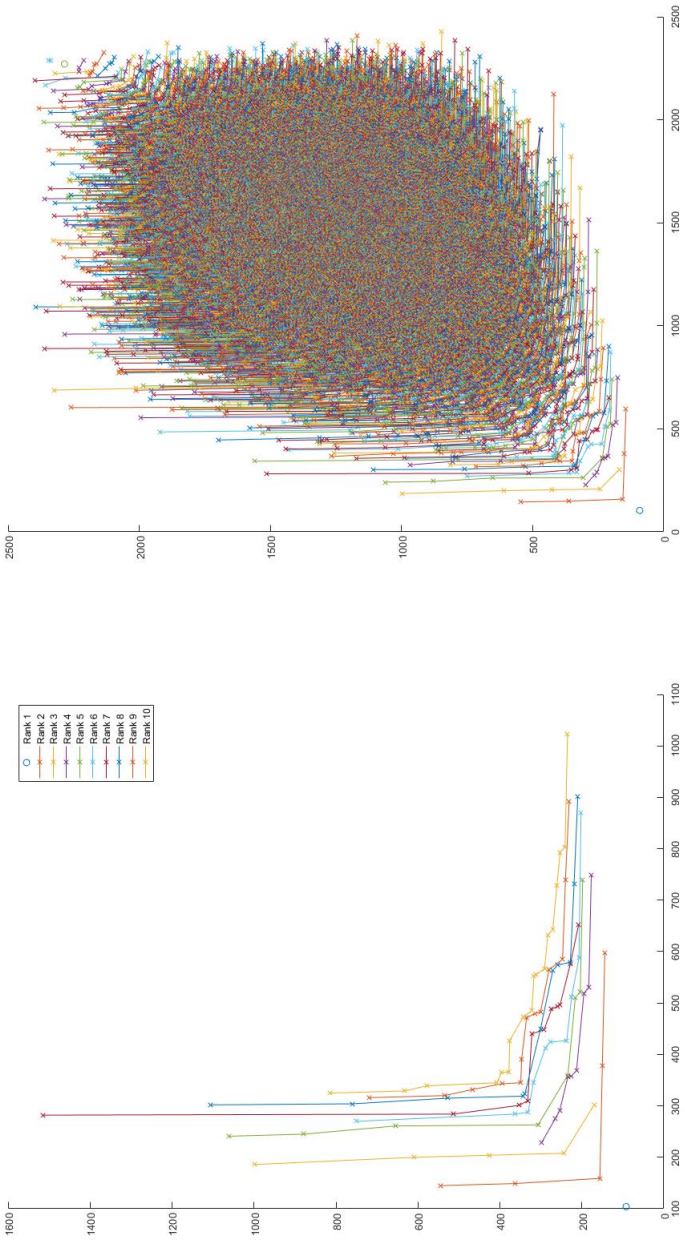
Proof: Consider any two successive ranks $\mathbf{r}_i, \mathbf{r}_{i+1}$. The ranking is antisymmetric since $\mathbf{r}_i \succ \mathbf{r}_{i+1} \iff \mathbf{r}_{i+1} \not\succ \mathbf{r}_i$, and is a total relation since the ranks cover A. Finally, the ranking is transitive, since each successive rank dominates the next, i.e. $\mathbf{r}_1, \succ \mathbf{r}_2 \succ \ldots \succ \mathbf{r}_n$, by Proposition 2.3. Hence a Pareto ranking is a total order over A, and since there is never equality between ranks, it is a strict total order.

Recall that for the points in the rank PFs, neither dominance nor non-dominance are equivalence relations. Dominance has transitivity but not symmetry or reflexivity [43]. Non-dominance has reflexivity and symmetry but not transitivity. As such, dominance represents a strict order, whilst nondominance is a semi-order; but the existence of nondominated points means that dominance is in general a partial order. As noted in [167], the addition of a tiebreaker function can facilitate the construction of a complete partial order, i.e. the points within a rank are also ordered, but Lemma 2.6 emphasizes that a ranking is a total order in terms of the ranks, without considering a tiebreaker function. In any case, a tiebreaker function will guarantee a complete total order if and only if each unique individual in a given rank also has a unique value for the tiebreaker function.

To illustrate the construction of rankings and some of the propositions above, consider Figure 2.1, which shows a full ranking (over 1000 ranks) for a randomly generated biobjective TSP with $N = 10^6$ in the right panel, and the first 10 ranks only on the left[1]. The first rank contains only one point (in blue), which

[1]This is a simulated, toy problem but the axis values can be interpreted as objective values with, for example, total distance on one axis and total cost on another.

Figure 2.1: Typical ranking



dominates all other points. If we consider the second rank (in red), note that the rightmost two points do not dominate any point in the third rank (in yellow). Of course, no point in any rank dominates any point in a higher rank, or is dominated by any point in a lower rank. Note that the plots of the ranks do not necessarily form approximations to convex curves; for example, some points have a higher value for one objective than the points in the same rank with both the next highest and the next lowest value for the other objective, forming a kink in the curve.

In practice, genetic operators in MOEAs create new individuals and the ranking is recalculated at each generation, meaning the membership and number of ranks can change. However for the purpose of theoretical analysis, it is assumed in the rest of this chapter that the set of available unique points is fixed at generation 0, and that the number of ranks n and the membership of each rank $\mathbf{r}_i,\ i = 1 \dots n$ is also fixed, so that only the number of copies r_i of individuals mapped to points in each rank changes.

2.3 Elite nonsampling and nonselection in ranking systems

2.3.1 Background

The classical tournament selection algorithm [65, 66] consists of two phases; in the first, individuals are selected at random to participate in a given tournament, and in the second, exactly which individual passes to subsequent algorithms within the EA (i.e. genetic operations) is decided by comparing the outputs of a given fitness function. These have been called the sampling and selection phases, respectively [137]. We will adopt this terminology and consequently refer to the probability of sampling and selection as p^{samp} and p^{sel}, respectively, and use the terms "sampled" and "participating" interchangeably. Also we shall refer to the probability of victory, that is the probability of an individual being

selected given that it is sampled, as p^{vic}. In general, if sampling and selection are independent, then

$$p^{vic} \quad = \quad p^{samp} \cdot p^{sel}. \tag{2.6}$$

The nonsampling problem can be simply described thus: that depending on how the selection scheme is designed, there may be some probability that a particular individual is never sampled for any tournament at a given generation; of particular concern is elite nonsampling, i.e. the probability that one or more individuals $\in \mathbf{r}_1$ or more generally, that are members of higher ranks are not sampled and therefore not selected. This is distinct from the multisampling issue, which in the literature on the single objective case refers to the probability that more than one copy of an individual is sampled in a tournament; we can refer to this as the individual multisampling issue. In the multiobjective case, we must also consider the probability that more than one distinct individual of the same rank is sampled in a tournament; we can refer to this as the in-rank multisampling issue. Whilst elite nonsampling is clearly a problem as depending on the design of an EA, it can lead to the elimination from the population of the fittest individuals, multisampling alters the expected rank sizes, though in what way depends on the tournament selection scheme used.

Here and for the rest of this chapter, we assume all variables to be non-negative and in general non-zero and finite except where noted. In place of the setup for single-objective problems, where scholars have described a distribution in terms of the number of individuals for each fitness level [18, 19] , we rather look at the number of individuals r_{ig} at generation g, $g = 0 \ldots G$ that are members of each rank $\mathbf{r}_i$ as fixed at generation 0. Note that the natural ordering of these ranks is to denote the highest rank $\mathbf{r}_1$ as it is the first rank discovered by the ranking algorithm, but this is the opposite order to that generally used in the literature in the single objective case, hence care must be taken in interpretation.

Let $\mathbf{R}_i^+$ be the set of all individuals mapped to points in any rank down to

$\mathbf{r}_i$. That is:

$$\mathbf{R}_i^+ = \bigcup_{k=1}^{i} \mathbf{r}_k. \tag{2.7}$$

We now define the actual cumulative total number of individuals mapped to points in ranks from $\mathbf{r}_1$ down to a given rank $i \leq n$ after the tournament selection phase at the g-th generation as:

$$R_{ig}^+ = \sum_{k=1}^{i} r_{kg}, \tag{2.8}$$

where r_{kg} represents the number of individuals in rank $\mathbf{r}_k$ at generation g. Thus there are R_{ig}^+ individuals in total that are mapped to points in ranks down to and including $\mathbf{r}_i$ after selection at generation g, and in a constant population algorithm, $R_{ng}^+ = N \; \forall g, \; g = 1 \ldots G$.

Also note that whilst early research on the single-objective case [65] considers a sequential steady-state algorithm where the distribution of the population available for sampling changes after each tournament, it is easy to implement tournament selection in parallel [66], and in subsequent literature including [19], each tournament selection phase usually consists of m tournaments, often with $m = N$ so that population size stays constant across generations.

2.3.2 Tournament selection with replacement

Consider firstly a tournament with τ participants, drawn from a population of size N, $2 \leq \tau < N$, where each individual participating in a tournament is drawn with replacement from a discrete uniform distribution on $[1, N]$, and denote this tournament type Scheme A. This type of scheme is addressed for the single objective case inter alia in [19, 147, 149]; for the rank-based multiobjective case, we write the sampling probability for an individual in rank $\mathbf{r}_k$ at generation g as p_{kg}^{samp}. The cumulative probability that any particular participant in a

tournament at generation g will be one that is mapped to a point in $\mathbf{R}_i^+$ is:

$$\begin{aligned} P_{ig}^{samp} &= \sum_{k=1}^{i} p_{kg}^{samp} \\ &= \sum_{k=1}^{i} \left(r_{kg-1} \cdot \frac{1}{N} \right) \\ &= \frac{R_{ig-1}^+}{N}. \end{aligned} \tag{2.9}$$

The expected number sampled is:

$$E\left[R_{ig}^+\right] = \frac{\tau \cdot R_{ig-1}^+}{N} \in [0, \tau], \tag{2.10}$$

where τ represents the number of participants in each tournament (the tournament size), with variance:

$$\mathrm{var}\left(R_{ig}^+\right) = \frac{\tau \cdot R_{ig-1}^+ \cdot \left(N - R_{ig-1}^+\right)}{N^2}. \tag{2.11}$$

In [137], a negative exponential approximation of the number of individuals sampled (or not sampled) in the single-objective case is obtained by analogy to the coupon collector problem, and it is noted in [164] that the probability does not depend on N. We now demonstrate that a general limit can be found directly for the non-sampling probability for large N for all individuals mapped to points down to a given rank. If we conduct m tournaments independently, as is the case in parallel tournament selection, then the probability of no individual mapped to a point in rank $\mathbf{r}_i$ and above being sampled in any tournament is:

$$\Pr\left(R_{ig}^+ = 0\right) = \left(1 - \frac{R_{ig-1}^+}{N}\right)^{m\tau}. \tag{2.12}$$

This leads us to the following results:

Lemma 2.7. *In a generational MOEA using parallel tournament selection with*

constant population so that $m = N/\alpha$, $\alpha < N$, *and* α *divides* N, *then for finite* R^+_{ig-1} *and* τ, *Equation 2.12 approaches:*

$$\lim_{N\to\infty} \Pr\left(R^+_{ig} = 0 | m = N/\alpha\right) \quad = \quad e^{-\tau \cdot R^+_{ig-1}/\alpha} \tag{2.13}$$

for N *that is large with respect to* $\tau \cdot R^+_{ig-1}$.

Proof:

$$\begin{aligned}
&\lim_{N\to\infty} \left(1 - \frac{R^+_{ig-1}}{N}\right)^{\frac{N\cdot\tau}{\alpha}} \\
&= \lim_{N\to\infty} e^{\ln\left[\left(1-\frac{R^+_{ig-1}}{N}\right)^{\frac{N\cdot\tau}{\alpha}}\right]} \\
&= \lim_{N\to\infty} e^{\frac{N\cdot\tau}{\alpha}\ln\left(1-\frac{R^+_{ig-1}}{N}\right)} \\
&= e^{\lim\limits_{N\to\infty}\left[\frac{N\cdot\tau}{\alpha}\cdot\ln\left(1-\frac{R^+_{ig-1}}{N}\right)\right]} \\
&= e^{\frac{\tau}{\alpha}\cdot\left[\lim\limits_{N\to\infty}\frac{\ln\left(1-\frac{R^+_{ig-1}}{N}\right)}{1/N}\right]}.
\end{aligned}$$

Apply L'Hôpital's rule:

$$\begin{aligned}
&= e^{\frac{\tau}{\alpha}\cdot\left(\lim\limits_{N\to\infty}\frac{N\cdot R^+_{ig-1}}{R^+_{ig}-N}\right)} \\
&= e^{\frac{\tau}{\alpha}\cdot\left(\lim\limits_{N\to\infty}\frac{R^+_{ig-1}}{\frac{R^+_{ig-1}}{N}-1\cdot}\right)}.
\end{aligned}$$

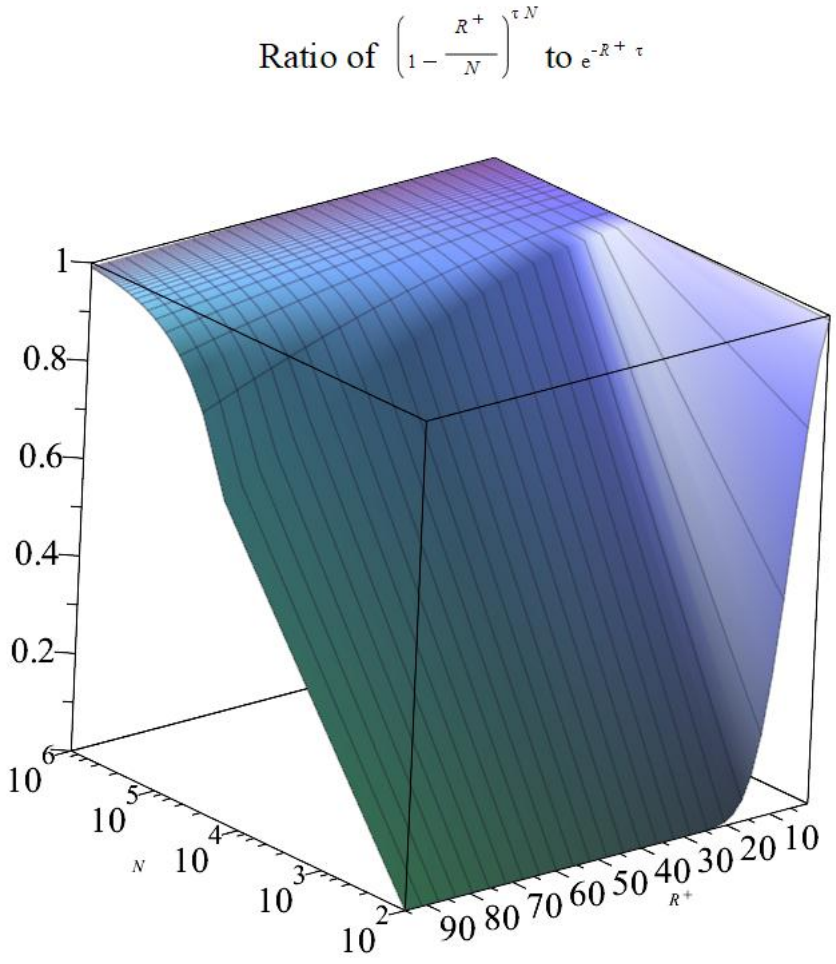
But as $\frac{R^+_{ig-1}}{N} \to 0$ as $N \to \infty$,

$$= e^{-\tau\cdot R^+_{ig-1}/\alpha},$$

as required.

Lemma 2.8. *The asymptotic probability at generation* g *that some elite individual* $\in \mathbf{r}_1$ *is sampled and selected in at least one of* N *tournaments approaches:*

Figure 2.2: Ratio of Equations 2.12 and 2.13



$$\lim_{N\to\infty} \Pr\left(r_{1g} = 0 | m = N\right) \quad = \quad 1 - e^{-\tau \cdot r_{1g-1}}. \tag{2.14}$$

Lemma 2.8 follows from Lemma 2.7 and the observation that victory is automatic for an elite individual sampled in any tournament, that is, $P_{1g}^{sel} = 1$.

Much of the analysis in this paper assumes that $m = N$, that is, the number of tournaments at each generation equals the population size so that with a single victor form each tournament, the population size does not change from generation to generation, though some EAs may conduct a different number, depending on for example crossover design. Lemma 2.7 permits a number of tournaments different to N by introducing the variable α, which leads to fewer than N tournaments if α is an integer greater than 1 (that divides N), and more if $\alpha < 1$; for example an algorithm might require $2 \cdot N$ tournaments to feed a crossover scheme that uses two parents to produce one offspring, in which case $\alpha = 1/2$.

In a given parallel tournament selection phase with constant population, asymptotically there is a non-zero chance of nonsampling (and therefore nonselection) of all individuals mapped to points in higher ranks, no matter how large the number of tournaments is in each phase. For example in an algorithm using binary tournaments with a single elite individual mapped to a point in the top rank $\mathbf{r}_1$ and $\alpha = 1$, i.e. a generational MOEA with N tournaments per generation, the limiting probability at each generation is $e^{-2} \approx 0.135$, which agrees with the observation made in [137]. The nonsampling probability can be varied by weighting the sampling probability by fitness score, but may not be eliminated, depending on the weighting scheme.

Because the limit in Equation 2.13 is approached from below as the population size rises, conversely the limit in Equation 2.14 is approached from above, with the rate in each case dependent on the size of $\tau \cdot R_{ig-1}^{+}/\alpha$ or $\tau \cdot r_{1g-1}$ respectively, relative to N. Although it is generally recognised that increasing τ

raises selection pressure, it is worth noting that although increasing τ will also decrease the nonsampling problem (but also decrease diversity), the starting distribution in particular can also be expected to have a large effect on how serious the elite nonsampling effect is, with the danger that in the absence of elitism or archiving, elite individuals could be lost in early generations. This could lead to suboptimal results, because good areas of the solution space are never explored, or to an EA taking far longer than necessary to rediscover these good areas through the effect of genetic operators. If elite individuals are archived, they will remain on record, but will not be passed to genetic operators this generation, or in future generations unless reintroduced into the population.

Higher values for the nonsampling probabilities might be acceptable if close neighbours in representation space have similar ranks. However if, as is often the case in real-world problems, there are one or more regions of the representation space where some individuals have much higher ranks compared to close neighbours, then we might particularly value the elite individuals at each generation and require their participation in selection, meaning that we would find a high probability of elite nonselection unacceptable. Hence if avoiding elite nonsampling is a priority, tournaments with larger numbers of participants are preferable (but note the caveat in Subsection 2.4.4). This limiting result is perhaps the reason why researchers have found some form of elitism indispensable when using sampling without replacement, particularly in binary tournaments.

Figure 2.2 shows the relationship between the general prediction for the nonsampling probability in Equation 2.12 and the limit from Proposition 2.7 as a ratio on the z-axis, calculated for values of $R^+ \in [1, 97]$ and $N \in \left[10, 10^6\right]$ with $\alpha = 1$ so that $m = N$, and shown with a log scale for N on the y-axis and a linear scale for R^+, representing the number of individuals in some generic set of elite ranks, on the x-axis.

In selection with replacement, individual multisampling somewhat increases diversity because it implies that where $r_n = 1$, that unique worst individual can still win a tournament by being sampled τ times in a tournament, with

probability $N^{-\tau}$. On the other hand, in the case of the highest rank individuals, the operation of the in-rank multisampling issue means there is some probability that more than one, different individual mapped to a point in $\in \mathbf{r}_1$ can be sampled and hence at least one will not be selected, after the operation of some tiebreaker function. Hence in the multiobjective case, even if all fitness objective values are unique, it is possible for an elite individual to be sampled for a tournament and yet not selected.

2.3.3 Tournament selection without replacement

A first alternative to selection with replacement might be a scheme in which participation in each tournament is decided over a uniform distribution without replacement, so that either no copies or exactly one copy of any given individual will participate, but the rosters for each tournament are independent; we will refer to this as Scheme B. Because we now have sampling without replacement, we can use the hypergeometric distribution to calculate the probability that no individuals in a given rank $\mathbf{r}_i$ or below participate in a particular tournament:

$$\Pr\left(R_{ig}^{+}=0\right) = \frac{\left(\begin{array}{c} N-R_{ig-1}^{+} \\ \tau \end{array}\right)}{\left(\begin{array}{c} N \\ \tau \end{array}\right)}. \tag{2.15}$$

Note again that the expression $N - R_{ig-1}^{+}$ includes all ranks from the i-th to the lowest rank. In the special case $R_{ig-1}^{+} = r_{1g-1} = 1$, Equation 2.15 reduces to

$$\begin{aligned} \Pr\left(r_{1g}=0|r_{1g-1}=1\right) &= \prod_{k=0}^{\tau-1}\frac{N-k-1}{N-k} \\ &= \frac{N-\tau}{N}. \end{aligned} \tag{2.16}$$

Although derived from the hypergeometric distribution with a different sampling probability to the binomial case considered in Scheme A, the expected number of individuals mapped to some point in $\mathbf{R}_i^+$ that are sampled is exactly the same as Equation 2.10 for $\tau \geq 2$, but the variance is smaller, by a factor of $(N-1)/(N-\tau)$, than that under Scheme A (Equation 2.11):

$$\text{var}\left(R_{ig}^+\right) = \frac{\tau \cdot R_{ig-1}^+ \cdot \left(N - R_{ig-1}^+\right) \cdot (N-\tau)}{N^2 (N-1)}. \tag{2.17}$$

Since all the tournaments are independent, the probability that no individuals mapped to some point in $\mathbf{R}_i^+$ participate in any tournament is:

$$\Pr\left(R_{ig}^+ = 0\right) = \left[\frac{\binom{N - R_{ig-1}^+}{\tau}}{\binom{N}{\tau}} \right]^m. \tag{2.18}$$

Whilst Equation 2.18 will produce results that differ from Equation 2.12 for Scheme A when τ is small to N, for large N the limit in Proposition 2.7 will also apply to Scheme B, since in the limit the hypergeometric distribution approaches the binomial. Figure 2.3 shows the relationship of Equation 2.18 to the limit as a ratio on the z-axis, while Figure 2.4 shows the relationship of Equation 2.18 to Equation 2.12, i.e. the relationship of Scheme B to Scheme A. In general, Scheme B approaches the limit more slowly and has a lower nonsampling probability than Scheme A with smaller populations; for large populations, there is no difference.

Although Scheme B eliminates the individual multisampling issue in each tournament, it does not eliminate the in-rank multisampling issue, and because the tournaments are independent, any individual may be sampled in any number of tournaments, including none.

Finally, we consider a completely permuted scheme, i.e. where sets of tournaments at a given generation are formed from complete permutations of all

Figure 2.3: Ratio of Equation 2.18 to the nonsampling limit

$$\text{Ratio of } \frac{\binom{N-R^+}{\tau}}{\binom{N}{\tau}} \text{ to } e^{-R^+\tau}$$

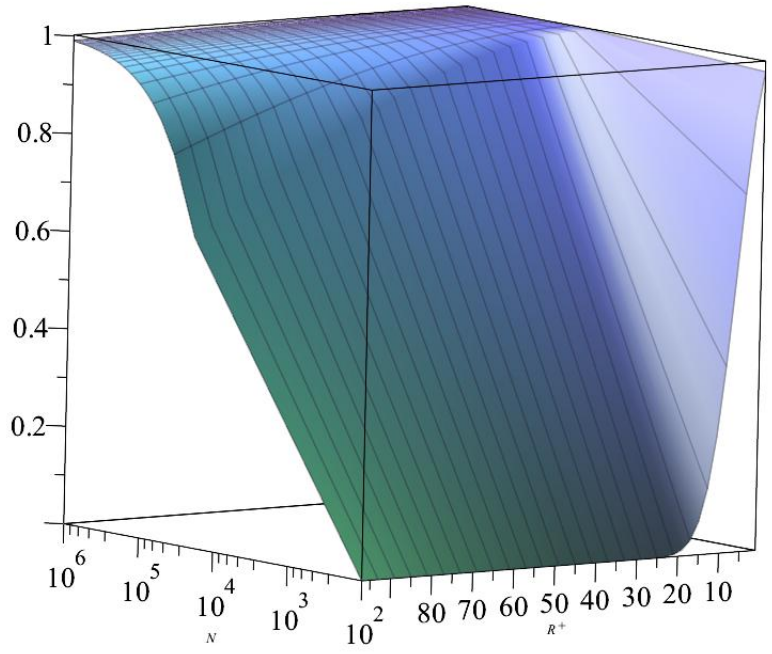
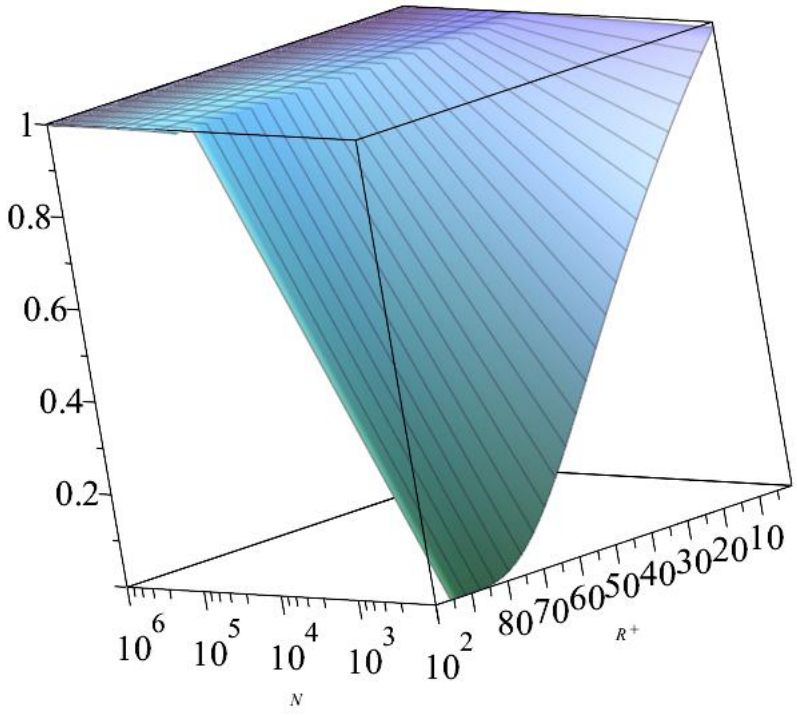


Figure 2.4: Ratio of Equation 2.18 to Equation 2.12

$$\text{Ratio of } \frac{\binom{N-R^+}{\tau}}{\binom{N}{\tau}} \text{ to } \left(1-\frac{R^+}{N}\right)^{\tau N}$$



unique individuals, so that all participate in at least one tournament. In Scheme C, as envisaged in [149], tournaments are formed by lining up τ complete permutations, so that the first participant in each tournament comes from the first permutation, the second from the second and so on. The mean and variance under Scheme C of the number of individuals from a given set of ranks in a given tournament, assuming no knowledge of any other tournaments, is the same as under Scheme A (Equations 2.10 and 2.11). The advantage of a completely permuted scheme is that the probability of nonsampling is always zero for any individual [44], and hence[2]:

$$\lim_{N\to\infty} E\left[R_{ig}^{+}\right] \quad = \quad \tau \cdot R_{ig-1}^{+}, \tag{2.19}$$

as it becomes unlikely with large populations that more than individual mapped to a point in $\mathbf{R}_i^+$ will be sampled in a given tournament, provided R_{ig-1}^+ is small with respect to N, and hence any individual mapped to a point in $\mathbf{R}_i^+$ that is sampled will almost certainly win its tournament.

Scheme C does not eliminate either the individual or in-rank multisampling issues, since the permutations are independent. In Chapter 4, a different permuted scheme is described which does largely eliminate multisampling issues, but does not use rankings explicitly; more details are to be found in Section 4.4.

[2]Note that this scaling property is also proved in different ways for selection with replacement in Theorem 2.3 and in Lemma 2.9.

2.4 How tournament selection changes the fitness rank distribution

2.4.1 Tournaments with replacement under a Pareto ranking system

2.4.1.1 How rank sizes change under tournament selection with replacement

We look now at the expected size of ranks after tournament selection with replacement (Scheme A).

Following the logic established for the single-objective case [19, 101, 137], for an individual mapped to a point in $\mathbf{R}_i^+$ to be selected, all individuals sampled in a given tournament must also be mapped to a point in $\mathbf{R}_i^+$, since the presence of any individual in any higher rank obviates the possibility of victory. We assume here and in all that follows that ties are settled by selecting at random, since this method of tie-break will not affect the expected rank distribution or any of the results since the tied individuals are of the same rank, though of course exactly which individual is selected will have an effect on other aspects of the post-tournament population, including the set of schemata available to genetic operators, but this affects representation space, not fitness rank space. The results may also apply when other types of tiebreaker are used, but only if they do not affect the expected rank distribution.

Now denote the rank of the l-th individual sampled for a tournament at the g-th generation as $\mathrm{r_g}(l)\,,\ l = 1 \ldots \tau$; since sampling is independent for each tournament slot in a scheme without replacement, from Equation 2.9, the probability that an individual not mapped to a point in $\mathbf{R}_i^+$ is sampled and selected is equivalent to the probability that no individual mapped to a point in $\mathbf{R}_i^+$ is sampled:

$$\Pr\left(\neg\left[\exists \mathrm{r}_g(l) \leq i,\ l = 1 \ldots \tau\right]\right) \quad = \quad \left(\frac{N - R_{ig-1}^+}{N}\right)^{\tau}. \tag{2.20}$$

Note here that the expression $N - R^{+}_{ig-1}$ includes all ranks from the i-th to the lowest rank. Hence the expected number of individuals mapped to a point in $\mathbf{r}_i$ that are sampled and selected after tournament selection over N tournaments $E[r_{ig}]$, $i = 1 \ldots n$ can be derived as a difference between expectations of binomially distributed variables:

$$\begin{aligned} E[r_{ig}] &= N \cdot \Pr(\neg[\exists \mathrm{r}_g(l) < i]) - N \cdot \Pr(\neg[\exists \mathrm{r}_g(l) \leq i]) \\ &= N \cdot \left[\left(\frac{N - R^{+}_{i-1,g-1}}{N} \right)^{\tau} - \left(\frac{N - R^{+}_{ig-1}}{N} \right)^{\tau} \right]. \end{aligned} \tag{2.21}$$

This formula corresponds to that derived in a different way for the single-objective case in [19]. Note that if $r_{ig-1} = 0$, then $E[r_{ig}] = 0$. We can also now define the complete set of expected post-tournament ranks as a vector of expected numbers in each rank:

$$E[\mathrm{R_g}] = \begin{bmatrix} E[r_{1g}] \\ E[r_{2g}] \\ \vdots \\ E[r_{ng}] \end{bmatrix}. \tag{2.22}$$

Note however that whilst $E[\mathrm{R_g}]$ can potentially be stated for any selection scheme, it gives information only on expected first moments; no other information about the expected distribution is contained or implied, so we should be careful from the outset not to conclude that, for example, different selection schemes with identical $E[\mathrm{R_g}]$ are necessarily equivalent in a more general sense.

In the particular case of the elite individuals mapped to points in $\mathbf{r}_1$, since $\mathbf{R}^{+}_{i<1} = \emptyset$, that is, there are no individuals in any rank higher than rank 1 and hence $R^{+}_{0g} = 0 \; \forall g$, then:

$$\begin{aligned} E[r_{1g}] &= N \cdot \left[\left(\frac{N - R^{+}_{0,g-1}}{N} \right)^{\tau} - \left(\frac{N - R^{+}_{1g-1}}{N} \right)^{\tau} \right] \\ &= N \cdot \left[1 - \left(\frac{N - r_{1g-1}}{N} \right)^{\tau} \right]. \end{aligned} \tag{2.23}$$

This implies importantly that:

$$E\left[r_{1g}\right] > r_{1g-1},\ 1 \leq r_{1g-1} < N, \tag{2.24}$$

so that selection pressure guarantees that the expected number of elite individuals grows after each tournament phase unless the first rank is empty or all individuals in the population are in the first rank, although as we shall see in Subsection 2.4.4, this is not necessarily true for any other rank. Hence a more complete formulation is:

$$E\left[r_{ig}\right] = \begin{cases} 0, & r_{ig-1} = 0; \\ N \cdot \left[1 - \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}\right], & i = 1; \\ N \cdot \left[\left(\frac{N - R^{+}_{i-1,g-1}}{N}\right)^{\tau} - \left(\frac{N - R^{+}_{ig-1}}{N}\right)^{\tau}\right], & otherwise. \end{cases} \tag{2.25}$$

2.4.1.2 Elitism

Many forms of elitism have been used (see inter alia [44, 37, 139]) and a full investigation of the effects is beyond the scope of this chapter. However, to illustrate the way elitism affects the expected rank sizes, consider a modification of Scheme A where all elite individuals in the highest rank $\mathbf{r}_1$ only are passed automatically, so that only $N - r_1$ tournaments are conducted to find the rest of the new population by sampling the whole population, including the elite individuals. In this case, the expected number of elite individuals after selection becomes:

$$\begin{aligned} E\left[r_{1g}\right] &= r_{1g-1} + (N - r_{1g-1}) \cdot \left[1 - \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}\right] \\ &= N - (N - r_{1g-1}) \cdot \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}. \end{aligned} \tag{2.26}$$

Hence with this particular form of elitism, Equation 2.25 becomes:

$$E[r_{ig}] = \begin{cases} 0. & r_{ig-1} = 0; \\ N - (N - r_{1g-1}) \cdot \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}, & i = 1; \\ (N - r_{1g-1}) \cdot \left[\left(\frac{N - R^{+}_{i-1,g-1}}{N}\right)^{\tau} - \left(\frac{N - R^{+}_{ig-1}}{N}\right)^{\tau}\right], & otherwise. \end{cases} \quad (2.27)$$

The nonsampling limit expressed in Equation 2.13 no longer applies to individuals in the first rank, but still applies to all others, including in particular individuals of high rank but just below the top rank. The effect of elitism can be seen by comparing equations 2.25 and 2.27. For the elite individuals, the difference in the expected size of the first rank between the case with elitism and without is:

$$\left[N - (N - r_{1g-1}) \cdot \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}\right] - N \cdot \left[1 - \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}\right] = r_{1g-1} \cdot \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}, \quad (2.28)$$

which is positive as long as N and r_{1g-1} are non-zero (as they cannot be negative). Hence, unsurprisingly, Equation 2.24 also holds in the case with elitism. For the rest of the population, the difference is:

$$- r_{1g-1} \cdot \left[\left(\frac{N - R^{+}_{i-1,g-1}}{N}\right)^{\tau} - \left(\frac{N - R^{+}_{ig-1}}{N}\right)^{\tau}\right], \quad (2.29)$$

which is always negative provided N , $r_{1g-1}$and $= r_{ig-1}$ are all non-zero.

2.4.2 Tournaments without replacement

For selection without replacement, an equivalent expression to Equation 2.21 is stated for the single-objective version of Scheme B in [23], which using this

chapter's notation and rank-ordering convention becomes:

$$E\left[r_{ig}\right] = N \cdot \frac{\binom{N - R^{+}_{i-1,g-1}}{\tau} - \binom{N - R^{+}_{i,g-1}}{\tau}}{\binom{N}{\tau}}. \tag{2.30}$$

However we should also take account of the fact that under Scheme B, individuals of the lowest fitness levels can never win a tournament unless there are more of them than the tournament size, meaning that such individuals will never be passed to genetic operators or to the next generation; that is:

$$N - R^{+}_{i-1,g-1} < \tau \Longleftrightarrow E\left[r_{jg}\right] = 0 \; \forall j > i. \tag{2.31}$$

This observation has important ramifications if a large tournament size is used; all individuals mapped to a point in the one or more of the lowest ranks $\mathbf{r}_j, \mathbf{r}_{j+1}, \ldots, \mathbf{r}_n$ may be automatically eliminated at a given generation, substantially increasing selection pressure.

We can also specify a special case for the highest rank, in a similar fashion to Equation 2.23 :

$$E\left[r_{1g}\right] = N \cdot \left[1 - \frac{\binom{N - r_{1g-1}}{\tau}}{\binom{N}{\tau}}\right]. \tag{2.32}$$

Hence for Scheme B a full formulation is:

$$E\left[r_{ig}\right] = \begin{cases} 0, & r_{ig-`1} = 0; \\ 0, & N - R^{+}_{i-1,g-1} < \tau; \\ N \cdot \left[1 - \dfrac{\binom{N - r_{1g-1}}{\tau}}{\binom{N}{\tau}}\right], & i = 1; \\ N \cdot \dfrac{\binom{N - R^{+}_{i-1,g-1}}{\tau} - \binom{N - R^{+}_{i,g-1}}{\tau}}{\binom{N}{\tau}}, & otherwise. \end{cases} \tag{2.33}$$

.

Under Scheme C, the total population available for sampling before the first tournament with τ permutations of the complete population is $\check{N}_0 = N \cdot \tau$, and even though tournaments can be computed in parallel once the permutations are chosen, sampling is sequential within the selection phase. Hence in each successive tournament of N in total, the available population for sampling decreases by τ, so that the population available after the t-th tournament is:

$$\check{N}_t \quad = \quad \tau \cdot (N - t)\,,\ t = 1 \ldots m,\ m \leq N. \tag{2.34}$$

The tournaments are not independent with respect to selection probability, but each tournament may include any number between 0 and τ of individuals mapped to a point in $\mathbf{R}^{+}_{i}$, including copies of the same individual, sampled independently for each permutation, and each successive tournament removes one available member of the total population independently from each permutation.

The total population mapped to points in $\mathbf{R}_i^+$ available before the first tournament at generation g across all permutations is equal to $\tau \cdot R_{ig-1}^+$, and let $\tau \cdot R_{igt}^+$ be the size of the remaining total population mapped to points in $\mathbf{R}_i^+$ after the t-th tournament, with $R_{ig0}^+ = \tau \cdot R_{ig-1}^+$. For each successive tournament, we have:

$$E\left[\tau \cdot R_{igt}^+\right] \quad = \quad \tau \cdot \left(R_{igt-1}^+ - \frac{\tau \cdot R_{igt-1}^+}{\check{N}_{t-1}} \right),\ t \in [1, m]\,, \tag{2.35}$$

which as an expectation is not required to be integer. This leads to the following result:

Theorem 2.1. *Expected rank sizes after N tournaments under Scheme* C *are the same as under Scheme A.*

Proof: Solving the recurrence relation in Equation 2.35 yields:

$$\begin{aligned} E\left[\tau \cdot R_{igt}^+\right] &= \tau \cdot R_{ig0}^+ \cdot \frac{N-t}{N} \\ &= \tau \cdot R_{ig-1}^+ \cdot \frac{N-t}{N}. \end{aligned} \tag{2.36}$$

From Equations 2.20 and 2.9, the probability that no individual mapped to a point in $\mathbf{R}_i^+$ is sampled and selected in the $t-$th tournament is:

$$\Pr\left(\neg\left[\exists \mathrm{r}_{gt}\left(l\right) \leq i\right] \forall l = 1 \ldots \tau\right) \quad = \quad \left(\frac{\check{N}_t - \tau \cdot R_{ig-1,t}^+}{\check{N}_t} \right)^{\tau}. \tag{2.37}$$

Hence, substituting from Equations 2.34 and 2.36, we have:

$$\begin{aligned} \Pr\left(\exists \mathrm{r}_{gt}\left(l\right) \leq i\right) &= \left(\frac{\tau \cdot \left[(N-t) - R_{ig-1}^+ \cdot \frac{N-t}{N}\right]}{\tau \cdot (N-t)} \right)^{\tau} \\ &= \left(\frac{N - R_{ig-1}^+}{N} \right)^{\tau}. \end{aligned} \tag{2.38}$$

But this is identical to the victory probability for selection with replacement in

Equation 2.20, and setting $m = N$, we have:

$$\begin{aligned} E\left[r_{ig}\right] &= \sum_{t=0}^{N-1}\left[\left(\frac{\check{N}_t - \tau \cdot R^{+}_{i-1,g-1,t}}{\check{N}_t}\right)^{\tau} - \left(\frac{\check{N}_t - \tau \cdot R^{+}_{ig-1,t}}{\check{N}_t}\right)^{\tau}\right] \\ &= N \cdot \left[\left(\frac{N - R^{+}_{i-1,g-1}}{N}\right)^{\tau} - \left(\frac{N - R^{+}_{ig-1}}{N}\right)^{\tau}\right]. \end{aligned} \tag{2.39}$$

The special case for rank 1 also follows, hence the expected rank sizes under scheme C are identical to those stated for Scheme A in Equation 2.25, as required.

As rank size expectations are the same as under Scheme A, much of the analysis for Scheme A presented in the rest of this section also applies under Scheme C. However, this does not imply that other distributional qualities are the same; in particular, the nonsampling probability for all individuals is zero under Scheme C, and hence elite nonselection is not an issue, unlike under Schemes A or B.

2.4.3 Analysis of convergence

An expression is derived in [66] for the expected takeover time in the context of a single objective problem and successive tournaments with replacement, and we seek a similar expression for a generational MOEA using a ranking system. For clarity, note once again that per the convention used in Subsection 2.2.2.1 above, the first rank $\mathbf{r}_1$ is the highest rank, so that the ordering runs in the opposite direction to that used in the literature on the single objective case, which follows the convention used in [19]. Let $\rho_g = r_{1g}/N$, that is, the ratio of the number of individuals in rank 1 to the overall population after selection at generation g. Then in an algorithm using iterated tournament selection only, i.e. without genetic operators or any other intervention, from Equation 2.23 we can model the expected number of individuals in those ranks at the next generation as:

$$E[\rho_g] = N \cdot [1 - (1 - \rho_{g-1})^{\tau}], \rho_g \leq 1, \tag{2.40}$$

so that Equation 2.50 also applies, substituting ρ_g for r_{1g}/N. Hence we can also appropriately tune τ in ranking-based algorithms based on the desired growth rate of the top ranks.

Theorem 2.2. *Under a ranking scheme and tournament selection with replacement, the expected number of generations to convergence for a large population approaches:*

$$G^* = -\frac{\ln(\rho_0)}{\ln(\tau)}. \tag{2.41}$$

Proof: Assume that the total number of feasible ranks is known and finite. From Equation 2.23 we have:

$$\lim_{N\to\infty} E[\rho_g] = \lim_{N\to\infty} N \cdot \left[1 - \left(\frac{N - \rho_{g-1}}{N}\right)^{\tau}\right] \tag{2.42}$$

$$= \tau \cdot \rho_{g-1}, \tag{2.43}$$

so that we can write down a recurrence relation:

$$\rho_g = \tau \cdot \rho_{g-1}. \tag{2.44}$$

Solving this recurrence relation yields:

$$\rho_G = \tau^G \cdot \rho_0. \tag{2.45}$$

Substituting $\rho_G = 1$ as the convergence condition, i.e. that the whole population consists in the end of individuals mapped to points in rank $\mathbf{r}_1$, and rearranging, we obtain Equation 2.41.

As stated earlier, Equation 2.41 assumes that only tournament selection

takes place in successive generations and ignores inter alia the diversifying effects of genetic operators on the population, which will generally delay convergence, as well as the effect of any explicit elitism in an algorithm, which should speed convergence. Nonetheless, Equation 2.41 should provide a useful baseline check to creators and users of MOEAs that convergence is as expected, and perhaps also aid in parameter tuning, including auto-tuning of the tournament size as examined in [163].

2.4.4 Selection pressure

Equation 2.23, which gives the expected number of elite individuals after the tournament selection phase under Scheme A, has the following partial derivative with respect to tournament size:

$$\begin{aligned} \frac{\partial E\left[r_{1g}\right]}{\partial \tau} &= -N \cdot \left[1 - \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}\right] \\ &\quad \times \ln\left(\frac{N - r_{1g-1}}{N}\right), \end{aligned} \tag{2.46}$$

which is exponentially decreasing in τ, but from Lemma 2.8, asymptotically this becomes:

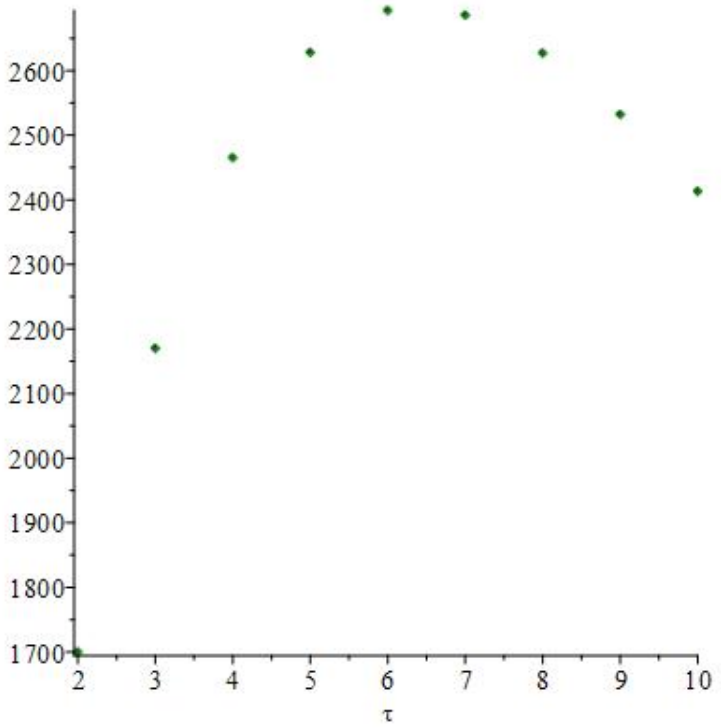
$$\lim_{N\to\infty} \frac{\partial E\left[r_{1g}\right]}{\partial \tau} = r_{1g-1} \cdot e^{-\tau \cdot r_{1g-1}}, \tag{2.47}$$

which is always positive for $r_{1g-1} > 0$ for all positive τ, hence $E\left[r_{1g}\right] > r_{1g-1}$, confirming Equation 2.24. For all other fitness levels however,we have the following result:

Theorem 2.3. *The tournament size that results in the largest expected rank size for a given rank $i > 1$ at generation g is given by:*

$$\tau^*_{ig} = -\frac{\ln\left[\ln\left(\rho_{i,g-1}\right) / \ln\left(\rho_{i-1,g-1}\right)\right]}{\ln\left(\rho_{i,g-1}\right) - \ln\left(\rho_{i-1,g-1}\right)}, i > 1, \tag{2.48}$$

Figure 2.5: Optimal selection pressure - example

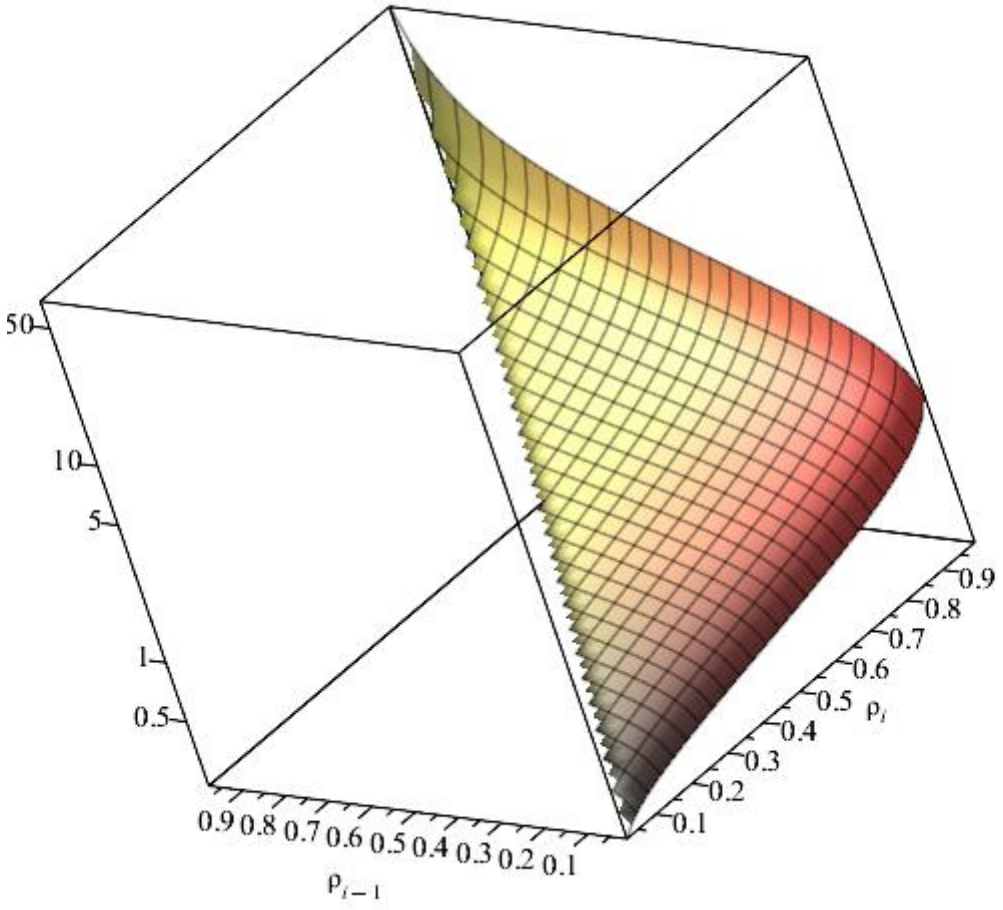


where $\rho_{ig} = \left(N - R_{ig}^{+}\right)/N,\ \rho_{ig} \in (0, 1]$.

Proof: Take the partial derivative of Equation 2.21 with respect to τ, set to zero and solving for τ, the result follows.

In general, $\tau_{ig}^{*} < N$, so that for all individuals mapped to points in $\mathbf{R}_i^{+}$, depending on the values for ρ_{ig-1}^{+} and $\rho_{i-1,g-1}^{+}$, there will generally be a level for τ beyond which selection pressure will decrease, even for individuals of high (but not the highest) fitness. For example in Figure 2.5, we set $N = 10000, R_{ig-1}^{+} = 2000, R_{i-1,g-1}^{+} = 1000$, so that $\rho_{i,g-1} = 0.1$ and $\rho_{i-1,g-1} = 0.2$. The green diamonds show the values of $E[r_{ig}]$ on the y-axis for values of $\tau = 2 \ldots 10$ on the x-axis under selection with replacement (Scheme A); in this example, the expected rank size peaks out with around 6 participants in each tournament. Figure 2.6 shows the general situation with different values of ρ_{ig} and $\rho_{i-1,g}$ on the x-axis and y-axis respectively, with a log scale for values of τ_i^{*} on the z-axis. A value of $\tau_i^{*} = 2$ or below implies that increasing tournament size will always

Figure 2.6: Values of τ^*



decrease the expected post-tournament size of the population mapped to points in $\mathbf{R}_i^+$ whilst $\tau_i^* \geq 3$ implies there is some $\tau > 2$ up to which increasing the tournament size will increase the expected post-tournament size of the population mapped to points in $\mathbf{R}_i^+$, but beyond which the expected population size will decrease again. This suggests that careful consideration should be given to the initial population distribution as well as to the parameter τ, as setting tournament size too high can decrease the number of individuals of high but not elite fitness, not just individuals of low fitness, possibly leading to premature convergence. Given that we know from equation 2.24 that elite populations always increase under tournament selection, in Equation 2.23 we can substitute $E[r_{1g}] = \beta \cdot r_{1g},\ \beta > 1$, where β represents the expected growth rate of the elite population, so that:

$$\beta \cdot r_{1g} = N \cdot \left[1 - \left(\frac{N - r_{1g}}{N}\right)^{\tau}\right]. \tag{2.49}$$

This leads to the following result:

Lemma 2.9. *For large populations, the elite cohort will increase in size after tournament selection by a factor approaching the tournament size.*

Proof: If we solve for τ in Equation 2.49 to obtain the number of participants required to achieve a given expected growth rate, we get:

$$\tau \quad = \quad \frac{\ln\left(\frac{N-\beta \cdot r_{1g}}{N}\right)}{\ln\left(\frac{N-r_{1g}}{N}\right)}. \tag{2.50}$$

If we consider the limit of the RHS of Equation 2.50, then by L'Hôpital's rule:

$$\lim_{N\to\infty} \frac{\ln\left(\frac{N-\beta \cdot r_{1g}}{N}\right)}{\ln\left(\frac{N-r_{1g}}{N}\right)} \quad = \quad \lim_{N\to\infty} \beta \cdot \frac{N - r_{1g}}{N - \beta \cdot r_{1g}}, \tag{2.51}$$

and as both numerator and denominator approach 1 as $N \to \infty$, this expression tends towards β. Hence in the limit, Equation 2.50 reduces to $\beta = \tau$. Equivalently, in Equation 2.23 substitute $E[r_{1g}] = \beta \cdot r_{ig}$ and take the limit of the

RHS:

$$\begin{aligned} \beta \cdot r_{1g} &= \lim_{N\to\infty} N \cdot \left[1 - \left(\frac{N - r_{1g-1}}{N}\right)^{\tau}\right] \\ &= \tau \cdot r_{1g}, \end{aligned} \tag{2.52}$$

so that again, $\beta = \tau$.

This result was already hinted at in Equation 2.42; the intuition is that there will be τ copies on average of each of the members of $\mathbf{r}_1$ selected to participate over the N tournaments and these copies will win all of their tournaments except where they face other individuals mapped to points in $\mathbf{r}_1$, the probability of which tends towards zero provided r_1 is small with respect to N. Knowing this maximal value of τ, at each generation, could aid tuning of the number of participants to achieve the desired level of selection pressure in terms of the rate of growth β of the elite population.

As for iterated tournament selection, an expression is given in [65, 66] for the elite population size under sequential selection (i.e. where the distribution changes after each tournament), and following [19, 18], in the case of parallel tournament selection in a generational MOEA after selection at the g-th generation we can write the expression as:

$$\rho_{1g} = 1 - (1 - \rho_{1,0})^{\tau^g},\ g = 1 \ldots G, \tag{2.53}$$

where $\rho_{1g} = r_{1g}/N$. This allows us to consider the case where elite individuals are archived and kept in the population even if not sampled in the selection phase at any given generation. Substituting into Equation 2.12 with $m = N$ yields:

$$\Pr\left(r_{1g} = 0\right) = \left[(1 - \rho_{1,0})^{\tau^{g-1}}\right]^{N\cdot\tau}, \tag{2.54}$$

which leads to the following result:

Theorem 2.4. *Under iterated tournament selection with replacement (Scheme A), given $s_{n0} > 0$ the nonsampling probability after g generations has the following limit:*

$$\lim_{N\to\infty} \Pr(r_{1g} = 0) \quad = \quad e^{-r_{1,0}\cdot\tau^g}. \tag{2.55}$$

This follows from Equation 2.54 and proof is similar to that of Lemma 2.7. The negative double exponential form of this expression implies that elite nonselection will be much less of an issue in algorithms with parallel tournament selection that include archiving, so that elite individuals are kept in the population even if they are not sampled. That said, nonselection for one or more generations will still delay the introduction of schemata from the elite individuals into the broader population via genetic operators.

2.5 Summary

In this chapter a set of results has been developed concerning both the operation of ranking schemes under multiobjective tournament selection and the effect on both the development of the rank or fitness distribution and the survivability of individuals. Firstly, results were developed regarding the operation of Pareto rankings, based in turn on axioms derived from the operation of a simple ranking algorithm. Then it was shown that there exists a general limiting expression for the probability of nonsampling under multiobjective tournament selection without replacement using Pareto rankings, and hence for the nonselection of elite individuals. This analysis can be adapted to selection without replacement as well as to the effect of the addition of elitism, and we examined the effects of these variations of the tournament selection algorithms, concluding that completely permuted schemes are of interest as they guarantee all individuals will be sampled yet remains tractable to analysis of the expected rank or fitness

distribution.

It has also been shown that there is a limiting expression for the expected number of generations to convergence for an iterative search algorithm using only tournament selection. This expression could be used as a useful check in developing or choosing algorithms. For example, if an algorithm converges close to or even faster than this limiting prediction, it could be considered likely that the algorithm is demonstrating premature convergence. It was then shown that one can find a tournament size that maximizes the expected rank size for any given rank below the first rank. Finally, it was shown that there exists a limit to the nonsampling probability across multiple generations that has a negative double exponential form, implying that archiving can be a partial solution to the elite nonsampling problem.

Whilst it has long been recognized that altering tournament size will tend to increase selection pressure, it has been shown that for all but the individuals in rank 1, there exists a value τ_{ig}^{*} for tournament size, at a given generation, beyond which selection pressure will in fact start to decrease. It has also been shown that the value of τ partially determines the nonsampling limit in some selection schemes, that the effect is doubly exponential under iterated tournament selection, and that it plays a role in expected time to convergence, which implies a trade-off between algorithm efficiency and the potential for premature convergence. Finally, it was shown that varying τ has a predictable effect on the growth rate of the elite population. Lemma 2.9 and Theorem 2.2, Equation 2.50 and Theorem 2.4 all reveal important effects of the choice of tournament size. Hence the value of τ must be carefully chosen, balancing these various considerations.

This chapter has highlighted some of the many choices with regards to multiobjective selection schemes that are available to the designer when creating an EA. In Chapter 4 we will see how the choice of permuted tournament selection with randomized size which does not use or require the calculation of rankings is an appropriate choice for a particular specialized MOEA. Before that however,

in the next chapter we will describe the mathematical model underlying the specialized MOEA.

Chapter 3

Partitioning multivariate self-affine time series

3.1 Introduction

Given a multivariate time series, possibly of high dimension, with unknown and time-varying joint distribution, it is of interest to be able to completely partition the time series into disjoint, contiguous subseries, each of which might be assumed for the purpose of further analysis to have different distributional or pattern attributes from the preceding and succeeding subseries. Whilst in the univariate case it is sometimes quite easy to agree on such a partitioning simply by visual inspection, including the number of partitioned subseries into which it is most appropriate to divide the time series and the location of the transitions between subseries, in the multivariate case visual inspection becomes impossible with more than a few dimensions. Clustering or partitioning of time series data has been widely studied in the machine learning and data mining literature; [106] and [91] contain good introductions to partitioning and clustering algorithms, respectively. In the alternative data analysis literature, where partitioning is generally referred to as change point detection (CPD) or break detection, [4] is

a recent survey of methods with applications in inter alia medical monitoring, climate change detection, speech recognition, image analysis, and analysis of human activity.

In the case of financial data, it has long been recognized that distribution varies over time, not in a smooth manner but rather in jumps between different states. Many well-established econometric models developed to deal with time-varying distributions nonetheless take the states as given, and do not address the question of identifying the different subseries corresponding to such states. Furthermore, most econometric models with time-varying parameters have difficulty dealing with the multivariate case, especially in high dimensions.

An additional feature of many time series is that they display self-affinity, which we may loosely define as the property that subseries at one time scale are similar to subseries at another after application of an affine transformation. Such observations in the natural world date back at least to work in the mid-20th century on hydrography of the Nile [82] and the length of international borders [142], and a more general theory was advanced in [119]. Many examples have since been found; for example one survey of the literature [135] cites inter alia work on air temperature, river discharge and tree ring spectra; variations in solar luminosity, sedimentation, and the earth's magnetic field; the structure of river networks; growth of plankton and many other flora and fauna; and in the human world, automotive and internet traffic flows. Many techniques exist for quantifying self-affinity, of which the best known perhaps is fractal dimension; a related, but different concept is to measure the complexity of a time series in terms of its multiscale entropy (MSE) [39, 36].

In economic and financial data, [117, 118] advanced the idea of self-similarity under power laws, later generalized and developed into the (MMAR) [121]. Ideas of self-similarity and power laws at work in financial data however have a longer history, at least back to work in the 1930s by R.N. Elliott [49, 50] in which he posits particular patterns occurring at different time scales with a relationship governed by a power law based on the golden ratio. A good deal of the most

interesting work on self-affine time series has been conducted within the fields of finance and econometrics, but many of the theoretical findings and tools developed can be of use in analysing a wider range of data types. However, to our knowledge there is nothing in the literature that addresses the inherently biobjective problem of identifying optimal partitionings of multivariate, self-affine data, as we will describe in this chapter.

The limitations shared by most econometric models that allow time-varying parameters in respect of higher dimension time series have already been noted. An additional problem is that most rely on particular assumptions regarding the underlying distributions and processes, and often require large numbers of parameters to be either provided or estimated from the data. This can make it difficult to separate the validity of the model and its assumptions from the particular data set and parametrization applied. An alternative approach is to build a model with as few assumptions and parameters as possible, in particular limiting the number of a priori choice parameters supplied by the modeller. The latter approach used in this chapter, the main contribution of which is is the development of a theoretical model based on self-affinity of a time series in terms of the similarity of a coarse-grained partitioning of the whole to a fine-grained partitioning of a subseries, and in Chapter 4, in which a specialized MOEA with limited parameters is developed to apply the model.

The remainder of this chapter is organized as follows. Section 3.2 explains the necessity of a specialized a approach in designing an MOEA suitable for this task, introduces the concepts of self-affinity and partitioning into distinct subseries for univariate and multivariate time series, and discusses some common statistical models. Section 3.3 discusses more specific aspects of modelling self-affine multivariate time series of financial data including techniques to reduce computational complexity, and defines the biobjective optimization problem to be addressed in validly partitioning such time series. A summary of this chapter follows.

3.2 Background and motivation

3.2.1 Necessity of a specialized approach

In this chapter an approach is presented which is designed specifically to simultaneously identify non-overlapping partitionings of both coarse-grained and fine-grained subseries. The aim is thus explicitly not to define clusters of overlapping or non-contiguous time series subsequences. Although certain types of clustering algorithm, including the widely-used k-means algorithm, have been categorized as partitional algorithms [55], such algorithms are typically not constrained to produce only partitions that are contiguous along one dimension (i.e. time) but if used for time series data, can cluster together individual observations from many different parts of the entire time series. In terms of taxonomy, we thus draw a distinction between partitioning algorithms of the type set out in this paper, which are also called segmentation approaches [133, 70, 140], and the more general case of clustering algorithms. Hence the approach is also significantly different in this respect to clustering EAs [40, 124, 74].

Furthermore, clustering algorithms commonly have an assumption of decreasing similarity of points in a cluster as distance from a centroid increases, as well as decreasing dissimilarity from points in other clusters as distance from them decreases. The time series partitioning algorithm presented here rather assumes homogeneity within partitions and sharp differences with other partitions, and so is arguably much more suitable for data that displays rapid transitions between different subseries in terms of some measure or set of measures. The combination of a population-based approach and subgroup separation means that useful results can be obtained from a single run, although the stochastic nature of the algorithm still means more useful results may be obtained from multiple runs.

In this Chapter, as well as in the description of the specialized MOEA developed in Chapter 4 and of testing methodology and results in Chapter 5, there is an explicit assumption that data being examined have the property

of self-affinity. As we shall see in Subsection 3.2.4, although methods exist to identify partitions in time series, these are generally based on a single objective and are not suitable for, or easily extended to, the inherently biobjective problem described in this chapter. Furthermore, the particular nature of the objective functions required for simultaneous identification of coarse-grained and fine-grained partitions together with the constraints necessary to produce valid partitions cannot be handled by EAs and other methods not created specifically to solve this particular problem, and requires a specialized approach, as we will see in later sections.

Specialization also has implications for the design of all the main elements of the MOEA. In particular, the choice of a tournament selection scheme that uses permutations and randomized tournament size accords with some of the conclusions from Chapter 2. However, whilst choice of selection scheme is certainly key to the development of the multiobjective fitness distribution of the population, specialization involves many other choices in MOEA design, as will be explained in greater detail in Chapter 4. The design of the tournament selection algorithm is founded in the theory developed in Chapter 2; many of the other choices and innovations however are of a more practical nature to ensure the MOEA executes efficiently, or indeed that it can produce an acceptable solution set at all.

3.2.2 Subseries and self-affinity

Consider the general case of a multivariate time series $\mathbf{S}$ with m simultaneous individual time series, each with T time-ordered samples. Initially we know nothing a priori about the distributions of the individual series or about the joint distribution, except that some distributional features may be time-varying, and in particular that $\mathbf{S}$ may be partitioned into two or more disjoint subseries each of which has distinct distributional attributes that distinguish each subseries, or partition, from the preceding or following one, forming a complete and still strictly time-ordered partitioning of $\mathbf{S}$. We also know that the first and last time-

ordered subseries may be incomplete, in the sense that data collected after the T-th sample may still be part of the final subseries, whilst if we could collect data before the first sample, some number of additional, earlier samples might still belong to the first subseries. Hence we may firstly typify such a coarse-grained partitioning $\mathbf{K}_\kappa$ of $\mathbf{S}$ into κ disjoint multivariate subseries $\mathbf{S}_k,\ k = 1 \dots \kappa$, in terms of an ordered set of $\kappa - 1$ cutpoints $C_\kappa = \{c_1, c_{2,\dots}, c_{\kappa-1}\}$, with each $c_k \in [1, T]\,,\ k = 1 \dots \kappa - 1$. By convention we will consider each cutpoint $c_k, k < \kappa$ to be the last point in a subseries, so that the next subseries $\mathbf{S}_{k+1}$ begins immediately after cutpoint c_k , and all the c_k are unique, so that each subseries $\mathbf{S}_k \neq \emptyset,\ k = 1 \dots \kappa$, and the final subseries $\mathbf{S}_\kappa$ ends at time T. Hence a coarse-grained partitioning $\mathbf{K}_\kappa = \{\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_\kappa\}$ is a time-ordered set of non-overlapping subseries of $\mathbf{S}$, and $\mathbf{S} = \cup_{k=1}^{\kappa} \mathbf{S}_k$.

Our motivation is to discover what we can about the partitions, in particular where the dividing points c_k between partitions may most usefully be placed for the type of analysis in which we are interested; yet we do not necessarily even know yet how many such partitions there are. Indeed, several optimal partitionings $\mathbf{K}_\kappa$ with different numbers κ of subseries may be considered possibly equally valid, and indeed as we shall see later, if we have more than one metric for considering different partitionings, there may be more than one Pareto optimal partitioning even when the number of partitions is the same. Such a partitioning of a multivariate time series could have many uses for a variety of different types of data. Beyond using such a partitioning for further analysis of the data set, it may in particular be possible to solve a further multiobjective optimization problem of the form:

Minimize

$$\mathbf{f}_\omega(\mathbf{K}_\kappa) \quad = \quad [\mathbf{f}_\omega(\mathbf{S}_1), \mathbf{f}_\omega(\mathbf{S}_2) \dots \mathbf{f}_\omega(\mathbf{S}_\kappa)]\,, \tag{3.1}$$

where some vector-valued function $\mathbf{f}_\omega(\mathbf{S}_k) = [f_1(\mathbf{S}_k), f_2(\mathbf{S}_k) \dots f_\omega(\mathbf{S}_k)]^{\mathrm{T}}$, calculated over ω objectives, is applied separately to each partitioned subseries

$\mathbf{S}_k, k = 1 \ldots \kappa$ and a Pareto-efficient solution set is obtained. One application of this would be portfolio optimization, where one could produce portfolios robust to several different sets of market circumstances, but many other applications could be imagined. The problem of identifying the nature of the last partition, which we may term the current partition, is of particular interest, but as already noted we may have incomplete information, since the true end point c_κ of the final subseries cannot be known at time T as it falls at some later point.

Furthermore, we will consider data that has a particular structural attribute, namely that of self-affinity. Sets with fractal properties are sometimes loosely referred to as self-similar, but in many datasets true scale invariance or statistical self-similarity as posited in [119] is replaced by self-affinity, and [121] reserves the term self-similarity for geometric objects that are invariant under isotropic contraction. The concept of self-affinity is defined in [120] as follows: let $X(t, \omega)$ be a random function defined on $-\infty < t < \infty$, with $\omega \in \Omega$ the set of possible values of the functions. Then such a random function is self-affine with exponent $H > 0$ if for every $h > 0$ and any t_0,

$$\begin{aligned} X(t_0 + \tau, \omega) - X(t_0, \omega) \quad &\stackrel{d}{=} \\ \left\{ h^{-H} \left[X(t_0 + h\tau, \omega) - X(t_0, \omega) \right] \right\}, \end{aligned} \tag{3.2}$$

where $X(t_0, \omega) \stackrel{d}{=} Y(t, \omega)$ indicates two random functions with the same joint distributions. The literature generally analyses self-affinity in terms of fractionally integrated Brownian motion (FBM) processes, i.e. $X(t, \omega) = B_H(t, \omega)$.

In contrast, except where noted we will make no assumptions about the underlying data-generating process (DGP) and will rather describe self-affine sets in terms of similarity achieved by making affine transformations equivalent to a restriction of the usual geometric type at different time scales; later we will consider the statistical invariances involved. In the most general sense, however, we will define the property of self-affinity as follows: let $\mathbf{F}_\kappa = \{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_{\kappa-2}\}, \kappa \geq 3$ be a fine-grained partitioning of some sub-

series $\mathbf{W}$ of $\mathbf{S}$ which in general terms is sufficiently short compared to $\mathbf{S}$ that it can be considered of a different "scale". Then $\mathbf{S}$ can be described as self-affine if the two-scale partitioning pair $\{\mathbf{K}_\kappa, \mathbf{F}_\kappa\}$ obeys

$$\exists \left(\mathbf{A}\cdot\mathbf{W}_j + \mathbf{B}\right) \overset{\mu}{\approx} \mathbf{S}_k \forall j, k,\ k \in [2, \kappa - 1]\,,\ j = k - 1, \tag{3.3}$$

for at least one such subseries $\mathbf{W}$; that is to say, at least one fine-grained partition of some subseries of $\mathbf{S}$ is by some measure μ similar, after the application of a suitable affine transformation, to a coarse-grained partition of the whole of $\mathbf{S}$, excluding an initial and a final incomplete subseries. We define the relationship $\overset{\mu}{\approx}$ in terms of the similarity, by a measure yet to be defined, of some fine-grained, time-ordered set of subseries $\{\mathbf{W}_j, j = 1 \ldots \kappa - 2\}$ after such an affine transformation to the coarse-grained set of subseries $\{\mathbf{S}_k, k = 2 \ldots \kappa - 1\}$; later we will impose limitations on the nature of $\mathbf{A}$, induced by the choice of measure. The first and last subseries of the coarse-grained partitioning $\mathbf{K}_\kappa$ are excluded because they are incomplete, so we must always have $\kappa \geq 3$, and $\mathbf{F}_\kappa$ comprises $\kappa - 2$ fine-grained partitions. Self-affinity should be understood at this stage in a very general sense; it does not necessarily imply that the compared subseries have the same statistical distribution, though this may be so depending on the measure employed, but might also refer to other attributes, such as the recurrence of certain patterns in the time series data.

Consider initially the case with $\kappa = 3$ so that there is a single complete interior coarse-grained partition, and hence a single fine-grained subseries is considered, so that Equation 3.3 reduces to

$$\left(\mathbf{A}\cdot\mathbf{W}_1 + \mathbf{b}\right) \overset{\mu}{\approx} \mathbf{S}_2; \tag{3.4}$$

that is, an affine-transformed version of the single fine-grained subseries is similar under μ to the second, i.e. the only interior, coarse-grained subseries (the first and third being incomplete). If we compare Equations 3.2 and 3.4, several key differences emerge. Firstly Equation 3.2 refers to affinity between a con-

tinuity of subseries of different lengths starting from t_0 which can be thought of as an expanding moving average, but Equation 3.4 compares the interior part of one coarse-grained partition within the portion of $\mathbf{S}$ that excludes the incomplete initial and final subseries to a single fine-grained subseries which may start and end at any point within $\mathbf{S}$.

Secondly, Equation 3.2 requires some assumptions about the nature of $X(t, \omega)$ in order to be useful. In the literature $X(t, \omega)$ is usually an FBM process; by contrast, Equation 3.4 makes no assumptions about the underlying process. Thirdly, if we consider a univariate version of Equation 3.4 and take $h = b$, Equation 3.2 requires a particular power law relationship between the translation and linear transformations, namely $a = b^H$; Equation 3.4 does not make any assumptions about the relationship between $\mathbf{A}$ and $\mathbf{B}$.

We can now define the principal problem, which we can cast as a biobjective minimization problem. Let $\mathbf{d}_{K_\kappa}$ be a $(\kappa - 1) \times 1$ vector of values of some distance function $d(\mathbf{S}_j, \mathbf{S}_k) \in [0, 1]$, $k = 2 \dots \kappa$, and $g(\mathbf{d}_{K_\kappa})$ be some summarizing function that returns a non-negative scalar, so that:

$$f_1(\mathbf{S}) = g(\mathbf{d}_{K_\kappa}). \tag{3.5}$$

Also, let:

$$f_2(\mathbf{S}) = h(-\mathbf{d}_{F_\kappa}), \tag{3.6}$$

where $\mathbf{d}_{F_\kappa}$ is a vector of the same distance function calculated for each of the $\kappa - 2$ pairs of sequential subseries at the fine-grained and coarse-grained levels, excluding the incomplete first and last subseries of the coarse-grained partition:

$$\mathbf{d}_{F_\kappa} = [d(\mathbf{W}_1, \mathbf{S}_2) \dots d(\mathbf{W}_{\kappa-2}, \mathbf{S}_{\kappa-1})]^{\mathrm{T}}, \tag{3.7}$$

and $h(-\mathbf{d}_{F_\kappa})$ is a summarizing function of the negative of the distance vector, so that it indicates similarity between the coarse-grained and fine-grained subseries.

Then the problem for a partition into a given number of subseries κ is the following biobjective problem:

Minimize

$$\begin{aligned}\mathbf{f}(\mathbf{S}) &= [f_1(\mathbf{S}), f_2(\mathbf{S})] \\ &= [g(\mathbf{d}_{K_\kappa})\, h(-\mathbf{d}_{F_\kappa})].\end{aligned} \tag{3.8}$$

The only explicit assumptions here are that $\mathbf{S}$ may be validly partitioned into κ distinct coarse-grained subseries, the first and last of which may be incomplete; that the coarse-grained partitioning may be typified as similar to some fine-grained partitioning of a subseries of $\mathbf{S}$; and that the differences between the subseries of the coarse-grained partitioning and the similarities between the sequential subseries of the coarse-grained and fine-grained partitionings respectively can be measured by the same distance function d. We make no other assumptions about the actual distributions or DGPs of $\mathbf{S}$ or its subseries. Of course Equation 3.8 allows the possibility that there may be many valid fine-grained partitionings of different subseries that show similarity to any given coarse-grained partitioning, and in forming the Pareto frontier the biobjective function will make use of functions g, h which in some sense summarize the distributions of all solutions found.

3.2.3 Regimes in econometrics and finance

Having established the framework for a general multivariate time series $\mathbf{S}$ and an unspecified distance function, we now make a more specific assumption, namely that changes over time in $\mathbf{S}$ may be identified and partitioned with reference to changes in variance and covariance of subseries of $\mathbf{S}$. Such cases have been extensively studied in econometrics, where the partitioned subseries are commonly referred to as regimes, and we will use the terms partition, regime and subseries interchangeably henceforth. However most of the literature concentrates on univariate data or data with only a very few variables, and rather than seeking

to identify the regimes, concentrates on analysis of the distributional qualities of the data with the regimes taken as given. A very brief summary of a large literature follows, which although principally from the areas of econometrics and finance, contains principles and techniques which may shed light on processes observed in many types of data.

3.2.3.1 Volatility regimes

Although there is no consistent definition of a regime, the terminology comes largely from seminal papers by Hamilton [71, 72, 73] which consider the case where κ possible regimes exist from which a particular observation y_t[1] may be drawn, and an unobserved state variable s_t which takes an integer value $(1 \dots \kappa)$ such that $\boldsymbol{y}_t$ depends on the current and most recent m autoregressive lags of $\boldsymbol{y}_t$, the current and most recent m values of s_t and a vector of parameters $\boldsymbol{\theta}$; that is:

$$\begin{aligned} p\left(\boldsymbol{y}_t | s_t, s_{t-1}, \dots s_{t-m}, \boldsymbol{y}'_{t-1}, \boldsymbol{y}'_{t-2} \dots \boldsymbol{y}'_{t-m}\right) & \\ \equiv \quad p\left(\mathbf{y}_t | \mathbf{z}_t; \boldsymbol{\theta}\right), & \end{aligned} \tag{3.9}$$

$$z_t \quad \equiv \quad \left(s_t, s_{t-1}, \dots s_{t-m}, \boldsymbol{y}'_{t-1}, \boldsymbol{y}'_{t-2} \dots \boldsymbol{y}'_{t-m}\right).$$

Note that there is an assumption that the data is stationary, and it may be necessary to transform the data, for example by taking differences, in order to ensure stationarity. A vector autoregression may be generalized such that the constant terms, the covariance matrix and the autoregressive coefficients may all be functions of the state s_t, and the transition between states is modelled as a Markov chain. Although the subsequent literature explores many of the possibilities of this general formulation, most often the number of states is small (2 or 3) and a change in level or in volatility are considered more often than a change

[1] Note that in general notation follows that of the original authors for the rest of Section 3.2.

in autoregressive structure. An earlier paper [72] considered the proposal that there might be an occasional shift in the constant term around which a scalar fourth-order regression clusters, and many later papers find strong evidence for volatility clustering in financial time series.

Examples of volatility regimes in the literature include the finding in [63] that real interest rates can be partitioned in the time domain into regimes separated by sharp jumps caused by structural breaks such as the oil shocks of the 1970s; the analysis in [33] of volatility regime switches in world stock markets; and the study in [68] of changes in volatility regime evidenced by the behaviour of the VIX volatility index. Econometricians have continued to develop a testing methodology for such regime shifts, and the most commonly used test is the iterated cumulative sum of squares (ICSS) [85], later modified in [146] to be more robust to heteroskedastic[2] and leptokurtic[3] distributions of returns. Such tests however concentrate on detecting structural breaks in volatility in a univariate sense; they cannot analyse shifts in the overall covariance structure of the returns of a set of assets, such as the constituents of an index.

3.2.3.2 Time varying models: ARCH and GARCH

In recent years substantial research has been devoted to applying the self-affine FBM processes described in Section 3.2.2 to the regime-switching paradigm outlined in Section 3.2.3. By way of background, it had long been recognized that standard statistical models which assumed financial time series to be formed from independent, normally distributed random variables do not adequately describe the data. [117] and [52] found that return distributions did not conform to the standard normal distribution but were leptokurtic and [17] noted that returns were also asymmetric. As a result, the autoregressive conditionally heteroskedastic (ARCH) model of [51] and its various generalizations and adaptations have been used extensively to model financial data, as have the

[2]In the context of a time series, heteroskedasticity is observed if the variance is different for different subseries.

[3]Normal distributions with a kurtosis greater than 3 are described as leptokurtic.

stochastic volatility (SV) models of [125] and others.

To recap, the original ARCH model of [51] considers a first-order autoregressive (AR(1)) process of the form

$$y_t = \gamma y_{t-1} + \epsilon_{t,}, \tag{3.10}$$

where $E[\epsilon] = E[y] = 0, V(\epsilon) = \sigma^2$, but the conditional mean $E[y_t|y_{t-1}] = \gamma y_{t-1}$. By introducing heteroskedasticity, that is, time-varying variance,we find the ARCH(1) model:

$$y_t = \epsilon_t h_t^{1/2} \tag{3.11}$$

$$h_t = h(y_{t-1}, \boldsymbol{\alpha}) = \alpha_0 + \alpha_1 y_{t-1}^2, \tag{3.12}$$

where $\boldsymbol{\alpha}$ is a vector of unknown parameters, and the ARCH(p) model follows from a generalization of h_t to p lags. The GARCH model [21] offered a generalization of ARCH which has been shown to outperform empirically and can be more parsimonious than a higher-order ARCH model. The GARCH(p,q) model defines:

$$h_t = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i h_{t-1} \ , \tag{3.13}$$

so that GARCH(0,q)≡ARCH(q).The general null hypothesis is that $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$.

Multivariate extensions to the ARCH/GARCH family exist, but are generally hard to specify and estimate in higher dimensions. In [7], a an asymptotic test procedure introduced that can detect a single break point in a multivariate GARCH process, with extensions proposed, though not tested, for both multiple break points and heavy-tailed distributions, since the latter can make it difficult to successfully fit a GARCH process to the data.

3.2.3.3 The Multifractal Model of Asset Returns

Likelihood-based estimation of Markov-switching processes in the statistics literature predates the treatment of regimes in finance outlined in Subsection 3.2.3.1

[30]. Stochastic regime-switching models involve the conditional mean and variance being dependent on an unobserved and time-variant latent state which can change markedly and quickly. The latent state $M_t = \left\{m^1, m^2, \ldots m^d\right\}$gives rise to a model of returns:

$$r_t \quad = \quad \mu\left(M_t\right) + \sigma\left(M_t\right)\varepsilon_t. \tag{3.14}$$

The Markov chain M_t is governed by a transition matrix A, the components a_{ij} of which represent the probability that state j will follow state i. In [121], the authors introduced a new model based on FBM incorporating new scaling properties. They note that whilst FBM captures long memory in the persistent case where $1/2 < H < 1, H = 1/2$ representing ordinary Brownian motion, it captures neither the fat tails of the distribution of returns nor the time-variant volatility clustering often observed in financial data. Consider first a generalization of Equation 3.2:

$$\begin{aligned} X(t + c\Delta t) - X(t) \quad &\stackrel{d}{=} \\ M(c)[X(t + \Delta t) - X(t)], \quad &c > 0, \end{aligned} \tag{3.15}$$

where X and M are independent random functions. With certain restrictions on the distribution of the process, this implies the scaling rule:

$$E(|X(t)|^q) \quad = \quad c(q)t^{\tau(q)+1}, \tag{3.16}$$

where $\tau(q)$ and $c(q)$ are both deterministic functions of q, and $\tau\left(q\right)$ is referred to as the scaling function. In the special case of self-affine processes, the scaling function $\tau(q)$ is linear and fully determined by its index H, and such processes are referred to as unifractal or uniscaling; in the more general case, where $\tau(q)$ is non-linear, the processes are referred to as multiscaling.

Now let:

$$X(t) \quad = \quad \ln P(t) - \ln P(0), 0 \leq t \leq T, \tag{3.17}$$

where $P(t)$ is the price of a financial asset at time t. Note that $X(t)$ is generally assumed to be a stationary process in the econometrics literature. Then under the MMAR, it is assumed that:

$$X(t) \quad \equiv \quad B_H[\theta(t)], \tag{3.18}$$

where stochastic trading time $\theta(t)$ is the cumulative distribution function of a multifractal measure defined on $[0, T]$, $B_H(t)$ is FBM with self-affinity index H, and $B_H(t)$ and $\theta(t)$ are independent. Under these assumptions, $X(t)$ is multifractal, with scaling function:

$$\tau_X(q) \quad = \quad \tau_\theta(Hq). \tag{3.19}$$

Trading time $\theta(t)$can be thought of as a time deformation process, squeezing or stretching the price action, and trading time controls the moments of the distribution of $X(t)$.

3.2.3.4 Markov-Switching Multifractal (MSM) models

In [28], a discretized version of trading time is introduced in which prices are driven by a first-order Markov state vector with k components:

$$\begin{aligned} M_t \quad &= \quad \{M_{1,t}; M_{2,t}; \ldots; M_{k,t}\} \in \mathbb{R}_+^k, \\ E[M_{k,t}] \quad &= \quad 1. \end{aligned} \tag{3.20}$$

Returns are modelled as:

$$r_t \quad = \quad \sigma \left(\prod_{i=1}^{k} M_t^{(i)} \right)^{1/2} \cdot \mu_t, \tag{3.21}$$

where σ is the unconditional standard deviation and μ_t the mean return at time t. The multipliers in Equation 3.20 differ in their transition probabilities γ_k but not in their marginal distribution M, and $M \geq 0, E[M] = 1$. The Markov property guarantees that the distribution of the future multipliers depends only on the current multipliers. The transition probabilities are specified as:

$$\begin{aligned} \gamma_k &= 1 - (1 - \gamma_1)^{b^{k-1}}, \\ &\gamma_1 \in (0,1), b \in (0,\infty), \end{aligned} \tag{3.22}$$

and this process is referred to as Markov-Switching Multifractal (MSM) [29]. MSM is the main statistical technique embodying and generalizing the idea of self-affinity in financial time series in use today and indeed has become one of the more popular models in financial econometrics overall. The model can be estimated using maximum likelihood (ML) methods and the log-likelihood has a parsimonious closed form, making estimation relatively simple, albeit for restricted cases. Generalized Method of Moments (GMM) estimation techniques were introduced in [113], which address the case where the distributional model requires an infinite state space, as in the lognormal model proposed in [121], as well as addressing the computational complexity implied by some choices of distribution, for example the $2^k \times 2^k$ transition matrix required by the binomial model proposed in [28]. However the author notes that GMM is itself relatively computationally intensive compared to ML methods.

A sizeable body of empirical work has grown up in recent years around various approaches to the estimation of MSM models on different data sets using differing underlying distributions, including findings in inter alia [12, 29, 35, 69, 83, 84, 107, 108, 115, 114, 127, 144, 166]. However there is evidence that not all financial data fit well with the hypothesis that the data-generating process (DGP) for financial data is FBM; for example [131] finds anomalies in high-frequency data, especially when stock index data from emerging markets is analysed. Also, [38] claims that a new approach, discrete autoregressive stochastic

volatility (DSARV) outperforms MSM, and critique the high-dimensional state space inherent to the MSM approach.

A multivariate extension has been proposed [31], but in the natural generalization of the univariate MSM, the number of parameters therefore grows at least as fast as a quadratic function of the number of assets and so like multivariate GARCH is potentially highly computationally expensive, as well as complicated to analyse, for larger number of assets; hence the authors propose a factor model simplification rather than the true multivariate model. A bivariate model based on univariate decomposition [31, 83, 109] has also been studied, although the approach in [109] requires different volatility parameters at high and low frequencies, and [107] considered an extension of the GMM approach to multivariate models. A further approach to multivariate multifractals in the general literature uses operator fractional Brownian motion (OFBM) [45, 1]. In general however, as with most competing econometric models with time-varying volatility, most studies consider only the bivariate case or at most a handful of assets.

However [128] finds that the degree of multifractality displayed by different stocks is positively correlated to their depth in the hierarchy of cross-correlations, as measured through a correlation based clustering algorithm (DBHT, [150]). Since we know that such hierarchies demonstrate persistence in related correlation-based clustering techniques but nonetheless vary over time [122, 156], this further suggests the possibility that the correlation structure of financial markets may itself display multifractal characteristics.

3.2.4 Single objective methods for partitioning time series

Numerous methods for multiple change point detection are described in the alternative data analysis literature; in this subsection we will briefly examine some more recent statistically based, nonparametric, offline methods which are designed for or can be extended to the multivariate case, these being closest to the problem discussed in the rest of this thesis. Methods for detection of

change points[4] based on minimizing log-likelihood in univariate time series of known distribution have been considered for more than 6 decades [134]. For multivariate data, nonparametric techniques that have been considered more recently include relative density-ratio estimation [110], rank statistics [112] and minimum description length (MDL) [42].

Many, though by no means all, methods that are more specialized to financial time series applications make use of variance-covariance matrices, or their inverse form, sometimes called the precision matrices. These models often make strong assumptions; for example in [154], a graphical model is developed based on graphs formed with near-zero entries in the precision matrices for "blocks" of time-ordered multivariate observations removed; the model assumes inter alia that the data are normally distributed, that the graphs change gradually and that the distribution of block lengths is geometric; it also requires estimation of both parameters and hyperparameters.

By contrast, in [123] a nonparametric model is developed which has application both to financial and other multivariate data. Both agglomerative and hierarchical algorithms are proposed for identifying change points, and the objective function requires maximization of a form of Euclidean distance; in the case of financial data, this is applied to the underlying log returns. Testing is carried out using both simulated and real data.

In [42], financial data is not specifically considered, but a genetic algorithm is deployed to find change points, the fitness function being based on minimization of the Minimum Descriptive Length (MDL) which in this case is in effect a minimization of the residuals from a piecewise autoregressive model. Hence the data is assumed to have such a time-varying autoregressive structure, but has limited assumptions otherwise. The model also imposes constraints on block lengths, and uses an island approach with limited migration to subdivide the search space, as well as elitism, though the section mechanism is not described.

[4]Change points, break points or cutpoints are all interchangeable terms for the points marking the change from one regime, partition, subseries or block (terms that are again interchangeable) to another.

The convergence criteria are based on lack of change in the elite individuals and/or large number of migrations.

Although the approach developed in the rest of this thesis has some similarities with various of the approaches outlines above, as well as others in the literature, the principal differences are that in what follows the data is assumed to be self-affine, and the model developed is biobjective. The aim is thus not just to maximize the difference between partitions, as is generally the case with other approaches in the literature, but to simultaneously minimize the difference between sets of the same number of partitions at different timescales, to ensure the partitioning also fits as well as possible with the self-affine nature of the data. When various constraints on partition length are taken into account, as well as the large size of the search space, the complexity of the problem suggests an MOEA may be well suited, as we shall see in Chapter 4.

3.3 Partitioning based on realized covariance

3.3.1 Correlation of subseries covariances

Consider again a $T \times m$ multivariate time series $\mathbf{S}$ consisting of individual, in general correlated series $\mathbf{s}_j,\ j = 1 \dots m$, where $s_{jt} = \ln x_{jt} - \ln x_{jt-1}, t = 2 \dots T$, the x_{jt} being the raw data observations. We assume that $\mathbf{S}$ is globally weakly stationary and that a valid partitioning $\mathbf{K}_\kappa$ into κ non-overlapping and contiguous coarse-grained subseries is locally jointly ergodic, meaning that over sufficient time each mean $\mu_j \longrightarrow 0$ and that $\mathbf{S}$ may have joint distributional features that vary over time but are similar within a given subseries $\mathbf{S}_k, k = 1 \dots \kappa$. In particular, we assume initially that at a particular level of scaling, the cross-sectional vector of the log differences s_{kjt} of the m time series at time t obeys:

$$\begin{aligned} \left[s_{k1t}, s_{k2t} \dots s_{kmt}\right]^{\mathrm{T}} &= \boldsymbol{\mu}_k + \boldsymbol{\Sigma}_k \boldsymbol{\epsilon}_t, \\ & k \in [1, \kappa], \end{aligned} \tag{3.23}$$

where $\boldsymbol{\mu}_k$ is a vector of the subseries-dependent means for each individual series of log differences $\mathbf{s}_j$, $\boldsymbol{\Sigma}_k$ is the subseries-dependent variance-covariance matrix and $\boldsymbol{\epsilon}_t$ is a vector of random innovations for each $\mathbf{s}_j$ at time t with unknown distribution. Let us further assume that the variances and covariances are sufficiently locally stable for the purpose of forming a valid partitioning.

We might consider a distance function of the form:

$$d\left(\mathbf{S}_j, \mathbf{S}_k\right) \quad = \quad \sqrt{\frac{1}{2}\left(1 - \rho_{jk}\right)},\ k \in [2, \kappa]\,, \tag{3.24}$$

where $\mathbf{S}_j, \mathbf{S}_k$ are two subseries of $\mathbf{S}$, and ρ_{jk} is a standard matrix correlation between two covariance matrices, so that $d\left(\mathbf{S}_j, \mathbf{S}_k\right) \in [0, 1]$. There would however be several problems with using ordinary covariance matrices for such calculations in an EA with a large population and many generations. Firstly, the computational complexity scales in theory quadratically with m, although in practice the use of parallel processing and efficient algorithms reduces this scaling substantially. More seriously, the covariance matrices must be recalculated for each and every instance of the metric. In the next subsection we examine the potential for a technique from financial econometrics to considerably decrease the computational complexity.

3.3.2 Realized covariance

There is always a tension in considering sampling frequency between the desire to gain potentially greater accuracy, especially in complex time-varying frameworks, from a higher sampling rate against the inevitable increase in noise. As higher-frequency financial data has become more freely available, one approach to this has been to use realized volatility, where time-varying volatility is estimated as a series of observations for sub-periods (typically one trading day), the latter in turn being estimated from intraday data sampled at 5-minute or more

frequent intervals, so that an estimator of RV [5, 6, 10, 11] is:

$$\psi_t^2 \quad = \quad \sum_{l=1}^{L} s_{lt}^2, t = 1..T, \tag{3.25}$$

where s_{lt} is the l-th return observation of L total on day t; this very simple formulation is derived from arguments using stochastic calculus. Through aggregation, the use of RV can reduce the complexity inherent in the use of high-frequency data. As such, it is used as an input to many models, including GARCH, but can also be used in and of itself as a non-parametric modelling device; [13] proposes RV as a third dynamic volatility model class in its own right, alongside SV and GARCH, and use RV estimates both as an input to SV and GARCH and on their own to estimate volatility dynamics on the WIG20 stock index and the EUR/PLN exchange rate. The authors conclude that in general, models perform better using progressively higher sampling rates for RV, but that this is also true simply using RV on its own.

A natural extension of RV is realized covariance:

$$\begin{aligned} RC_{ijt} \quad = \quad & \sum_{l=1}^{L} s_{ilt} s_{jlt}, \; i, j = 1 \ldots m, \\ & t = 1..T. \end{aligned} \tag{3.26}$$

Realized variance/covariance has the useful property that it simply sums over periods of aggregation:

$$RC_{ijT} \quad = \quad \sum_{t=1}^{T} \sum_{l=1}^{L} s_{ilt} s_{jlt}. \tag{3.27}$$

In practice this means much simpler recalculation of the realized covariance matrices for a given subseries compared to calculating ordinary covariances. We can then conveniently specify a single partitioned $\frac{1}{2} m (m+1) \times \kappa$ array consisting of the realized variances and covariances for each subseries stacked into column vectors $\boldsymbol{\sigma}_k, k = 1 \ldots \kappa$:

$$
\begin{array}{ccc}
RC_{11t_1} & \dots & RC_{11t_\kappa} \\
\vdots & \ddots & \vdots \\
RC_{mmt_1} & \dots & RC_{mmt_\kappa} \\
RC_{21t_1} & \dots & RC_{21t_\kappa} \\
\vdots & \ddots & \vdots \\
RC_{mm-1t_1} & \dots & RC_{mm-1t_\kappa}
\end{array} \tag{3.28}
$$

$$
= \left[\begin{array}{ccccc} \boldsymbol{\sigma}_1^K & \cdots & \boldsymbol{\sigma}_k^K & \cdots & \boldsymbol{\sigma}_\kappa^K \end{array} \right]. \tag{3.29}
$$

We can then simply define the correlation of covariance matrices for successive coarse-grained partitions as:

$$
\rho_{K_k,K_{k-1}} = \operatorname{corr}\left(\boldsymbol{\sigma}_k^K, \boldsymbol{\sigma}_{k-1}^K\right), k = 2 \dots \kappa, \tag{3.30}
$$

and similarly the correlation of the correlation of a fine-grained partition as:

$$
\rho_{F_k,K_k} = \operatorname{corr}\left(\boldsymbol{\sigma}_k^F, \boldsymbol{\sigma}_k^K\right), k = 2 \dots \kappa - 1, \tag{3.31}
$$

where the variances and covariances of the fine-grained partitions, labelled $\boldsymbol{\sigma}_k^F$, are constructed analogously with Equation 3.29. It then remains only to specify suitable summarization functions g and h that capture enough of the distribution of the correlations. We will use:

$$
g\left(\mathbf{d}_{F_k,K_k}\right) = \sqrt{\frac{1}{4}\left(\overline{\rho_g} + \sup\left\{\rho_{K_k,K_{k-1}}\right\} + 2\right)}, \tag{3.32}
$$

$$
\overline{\rho_g} = \frac{1}{\kappa - 1}\sum_{k=2}^{\kappa} \rho_{K_k,K_{k-1}};
$$

$$h\left(\mathbf{d}_{F_k,K_k}\right) = \sqrt{\frac{1}{4}\left(2 - \overline{\rho_h} - \inf\left\{\rho_{F_k,F_{k-1}}\right\}\right)}, \tag{3.33}$$

$$\overline{\rho_h} = \frac{1}{\kappa - 2}\sum_{k=2}^{\kappa-1}\rho_{F_k,K_k}.$$

Both $g, h \in [0,1]$, and these formulations are suitable for use in a minimization problem per Equation 3.8, given that we wish to simultaneously minimize the similarity between successive coarse-grained subseries and the difference between the coarse-grained subseries and some set of fine-grained subseries.

3.3.3 Formulation of the optimization problem

We are now ready to fully state the computable final form of our optimization problem. Consider a set of m data series each of length T, with the observations grouped into equal periods of length n of log differences of observations. Form a $(T-1) \times m$ matrix which can be completely partitioned into $\kappa \geq 3$ non-overlapping and contiguous coarse-grained subseries $\mathbf{S}_k$ $k = 1 \ldots \kappa$, and call this partitioning $\mathbf{K}_\kappa$, noting that the first and last subseries $\mathbf{S}_1$ and $\mathbf{S}_\kappa$ are considered incomplete. Further, let $\mathbf{F}_\kappa$ be a fine-grained partition of $\kappa - 2$ sub-subseries $\mathbf{W}_j,\ j = 2 \ldots \kappa - 1$ of some subseries $\mathbf{W}$ of $\mathbf{S}$ which is sufficiently short as to be considered of a different scale to $\mathbf{S}$ as a whole.

We further require a minimum number of observations t_{min} in a given sub-subperiod in order for statistics such as covariance to be meaningful, and hence it is also a requirement that the length $\ell\left(\mathbf{W}_j\right)$ of each fine-grained sub-subseries be at least t_{min}. In order to maintain scale differentiation we further require that the length $\ell\left(\mathbf{S}_k\right)$ of each coarse-grained interior subseries be at least $\left(\kappa - 2\right)^2 \cdot t_{min}$, the idea being that any coarse-grained subseries could at least contain a fine-grained partition which has all its sub-subseries of minimum length t_{min}. For the first and last incomplete coarse-grained subseries, we only require that they are of length at least t_{min}. Hence for a given T, t_{min}, we can calculate a

maximum feasible number of coarse-grained subseries:

$$\kappa_{MAX} = \left\lfloor \sqrt{\frac{T-1-2\cdot t_{min}}{t_{min}}} + 2 \right\rfloor, \quad T > 0,\ 0 < t_{min} \leq \left\lfloor \frac{T-1}{2} \right\rfloor. \tag{3.34}$$

We wish firstly to minimize the similarity between each successive coarse-grained subseries $\mathbf{S}_k,\ k = 2 \ldots \kappa$ and the preceding subseries; note that we do not require dissimilarity of non-contiguous subseries, so a partition in which $\mathbf{S}_k$ is statistically very similar to $\mathbf{S}_{k-a},\ a > 1$ is permissible in this scheme. Next, for some fine-grained partition, we wish to minimize the dissimilarity of each sub-subseries $\mathbf{W}_j,\ j = 2 \ldots \kappa - 1$ to its counterpart interior coarse-grained subseries $\mathbf{S}_k,\ k = 2 \ldots \kappa - 1$. We might also consider a third objective, namely the maximization of the dissimilarity of successive fine-grained sub-subseries, but this is implied by the first two objectives, and is omitted in order to simplify the problem. Hence the biobjective minimization problem is:

Minimize

$$\mathbf{f}(\mathbf{S}) = \left[g\left(\mathbf{d}_{K_k,K_{k-1}}\right), h\left(\mathbf{d}_{F_k,K_k}\right) \right], \tag{3.35}$$

subject to:

$$\ell(K_k) \geq (\kappa-2)^2 \cdot t_{min},\ k = 2 \ldots \kappa - 1; \tag{3.36}$$

$$\ell(K_k) \geq t_{min},\ k = 1, k = \kappa; \tag{3.37}$$

$$\ell(F_k) < \ell(K_k),\ k = 2 \ldots \kappa - 1, \tag{3.38}$$

where $g\left(\mathbf{d}_{K_s,K_{s-1}}\right)$ and $h\left(\mathbf{d}_{F_k,K_s}\right)$ are as defined in Equations 3.32 and 3.33.

3.4 Summary

We have seen that the problem of how to best partition a multivariate time series into subseries with different distributional properties is an important one with many potential uses in different areas of research, but is also at root a difficult combinatorial problem with high computational complexity. The problem becomes more complex still if we assume self-affinity in the underlying DGP; yet many real data types, including but by no means limited to financial data, display this attribute. Identifying valid partitions on this basis may not only be the best way to identify time-varying features of the data but may also open the door to prediction of future states, or at least identification of the current state on some scale. However, statistical techniques currently available are not well-suited to high-dimensional multivariate analysis of time series showing time-varying, self-affine distributional attributes. Furthermore, all rely on assumptions about the underlying DGP and often on large numbers of model parameters.

The problem is formulated so as to minimize the similarity between successive coarse-grained subseries and maximize the similarity between this coarse-grained partitioning and some fine-grained partitioning at a smaller time scale, using functions that summarize each objective. This biobjective approach is specialized to self-affine time series, and as such is differentiated from the various single-objective partitioning approaches in the alternative data analysis literature. This summarizing approach significantly simplifies the problem, yet still yields a set of solutions for analysis a posteriori rather than a single solution based on a priori objective weightings, say, which would yield much less in terms of insights into the fitness landscape.

In the next chapter, we will see how this problem can addressed using a specialized partitioning MOEA, and will consider some of the trade-offs involved in designing an algorithm which produces satisfactory results with limited computational resources, as well as outlining the specialized techniques necessary to

solve the particular challenges of partitioning self-affine time series.

Chapter 4

A specialized MOEA for partitioning self-affine multivariate time series

Introduction

Although the theoretical model set out in the previous chapter makes a contribution to the understanding of the problem, unless we can find a way to address the problem in a way that is tractable given finite computational resources, that contribution would be purely theoretical. Unfortunately to develop an effective deterministic algorithm to solve such a complex, biobjective combinatorial problem nay not be possible, and standard stochastic approaches are unlikely to be effective given in particular the difficulties involved in producing solutions that are feasible for both coarse-grained and fine-grained partitionings.

Fortunately, evolutionary algorithms can be particularly well suited to such complex problems if suitably designed, and the main contribution of this chapter is to describe the implementation via a specialized MOEA with limited choice parameters of the theoretical model developed in Chapter 3. In terms of the

computational complexity tradeoff as formulated in Equation 1.2, at the highest level analysis, we can take C_T to be the main optimization objectives as described in Equation 3.8.

This chapter is arranged as follows. After a statement of the general aims of the MOEA, Section 4.2 describes the evolutionary algorithm, as well as certain general principles of design when addressing problems with high computational complexity and limited computational resources. Section 4.3 addresses the important topics of the choice and use of parameters, parallelization and the specialized nature of key aspects of the design of the MOEA. Section 4.4 describes the mathematical implications of the choice of permuted multiobjective tournament selection for the MOEA, and this is followed by a summary of this chapter.

4.1 General aims of the MOEA

Following [37], we note four general goals common in design of (a posteriori) MOEAs:

1. Preservation of nondominated points;
2. Progress towards points on PF_{true}, the Pareto front (PF) representing the global optimal multiobjective solution set;
3. Maintenance of diversity of points on PF_{known}, the set of currently known nondominated solutions;
4. Provide the decision-maker (DM) with a limited number of PF points on termination of the algorithm.

The first goal may be attained by virtue of the operation of genetic operators or through explicit or implicit elitism strategies. The second implies that successive generational PFs should themselves be nondominated with respect to previous PFs and should if possible be better than previous PFs, in that they either contain additional points that fill out the previous PF or contain points that

dominate one or more points from the previous PF. The third implies that points on the PF should not be crowded into a small number of regions, as this may indicate similar crowding into particular regions of the representation space, i.e. a lack of diversity in search directions, and also does not provide the DM with a diverse set of combinations of objective values. The fourth goal highlights that solution sets with a large number of points may be counterproductive in that they make it too hard for the DM to choose between them. We will refer to these goals in describing the specialized MOEA in following sections.

The process of creating a specialized MOEA is in itself a complex, multiobjective optimization problem, with potentially conflicting objectives that include inter alia:

- **accuracy** - actually finding the most optimal solution set;
- **robustness** - avoiding oversensitivity to parameter changes;
- **model parsimony** - using as few parameters as possible;
- **speed of execution** - which should apply to a range of data and parameters;
- **compactness** - making the code as compact as possible;
- **clarity** - making the code easy to understand both for yourself and others;
- **reusability** - making all the elements of the code useful for future projects;
- **platform universality** - the MOEA should run on a variety of hardware platforms (with necessary adjustments) and if possible, be convertible to other languages;
- **suitability** - the MOEA must suit the actual specialized use it is intended for.

A further set of aims in designing the specialized MOEA was to ensure that at least some minimum standard was met in all of the above.

Algorithm 4.1 Random initial assignment of constrained cutpoints (coarse-grained version)

Step 0: initialize a $T \times 1$ vector $\mathbf{w}$ of true binary values and set the first and last t_{min} entries to false, indicating these zones are unavailable for cutpoints
Step 1: randomly assign a cutpoint to any point in $\mathbf{w}$ with a true value and set the up to $(\kappa - 2)\, t_{min}$ entries $w_k, k \geq 1$ preceding the cutpoint and the up to $(\kappa - 2)\, t_{min}$ entries $w_k, k \leq T$ including and following the cutpoint to false, i.e. $[\min\{1, c_k - (\kappa - 2)\, t_{min} + 1\}, \max\{c_k - (\kappa - 2)\, t_{min}, T\}] = false$;
Step 2: loop to Step 1 until $\kappa - 1$ valid cutpoints are found;
Step 3: return the initial cutpoints.

Algorithm 4.2 Random assignment of initial constrained cutpoints (fine-grained version)

Step 0: initialize a $(t_2 - t_1 + 1) \times 1$ vector $\mathbf{w}$ of true binary values, where $[t_1, t_2] \in \mathbf{R}$, $t_{min} < t_1 < t_2 - t_{min} \leq T - 2 \cdot t_{min}$ corresponds to a subinterval of a coarse-grained partition in Algorithm 4.1;
Step 1: randomly assign a cutpoint to any point in $\mathbf{w}$ with a true value and set the up to t_{min} entries $w_k, k \geq 1$ preceding the cutpoint and the up to t_{min} entries $w_k, k \leq T$ including and following the cutpoint to false, i.e. $[\min\{1, c_k - t_{min} + 1\}, \max\{c_k - t_{min}, T\}] = false$;
Step 2: loop to Step 1 until $\kappa - 3$ valid cutpoints are found;
Step 3: return the initial cutpoints.

4.2 Functional description of the specialized MOEA

4.2.1 Representation, initialization and specialization in selection of fine-grained subseries

Each individual is represented by a pair of integer vectors of $\kappa - 1$ cutpoint locations, one coarse-grained and one fine-grained, with the cutpoint representing the first point in each coarse-grained subseries or fine-grained sub-subseries.

Random initialization of the population N of initial solutions is performed using Algorithms 4.1 and 4.2. However, were we to leave a single, unrestricted population, we should leave the evolutionary selection process open to two undesirable features. Firstly, as is usual with unrestricted EAs, solutions would quickly crowd into certain sections of the solution space, potentially leading to premature convergence. Secondly, the sets of fine-grained and coarse-grained cutpoints cutpoints would tend to approach each other in scale and location.

Typical countermeasures to the first effect [37] include:

- A weight-vector approach, where different weights are applied to bias the search and move solutions away from neighbours:
- Niching, where a penalty is applied to the fitness of solutions based on the number of solutions sharing some neighbourhood. MOEAs employing versions of this strategy include NSGA [151];
- Crowding/clustering, where solutions are subject to selection based on a crowdedness metric, as used in NSGA-II [44];
- Relaxed dominance, as in the ϵ-dominance technique used in [102];
- Restricted mating, where a minimum distance is required for recombination between any two given individuals.

In some other approaches, diversity may be intrinsic to the EA design; for example [165] finds that the decomposition technique that forms the basis of MOEA/D leads to a good chance of producing a uniform distribution of Pareto solutions.

In this specialized MOEA, we subdivide the population into a number of subgroups, and restrict the range $[t_1, t_2]$ within which each fine-grained subseries may be initialized, but with the requirement that the whole of $\mathbf{S}$ is covered, so that these ranges usually overlap. No restriction is placed on the coarse-grained subseries, which are initialized over the whole of $\mathbf{S}$. Since crossover is only allowed within a given subgroup, and mutation is also restricted (see Subsection 4.2.6 below), the algorithm is forced to try to find solutions involving fine-grained sub-subseries within the initial range unless mutation changes the available range. This maintains diversity, but it also solves the problem that in the absence of such subdivision, selection would over time push the fine-grained sub-subseries towards the same scale as the coarse-grained subseries, yielding a trivial solution set. This approach differs somewhat from the Island Model of [160] and similar distributed EAs in that no migrations of individuals between subgroups or crossover between individuals from different subgroups is

allowed, and the subgroups overlap, potentially to an increasing degree under the operation of mutation over numerous generations, which to some extent substitutes for migration (see Subsection 4.2.4).

The number of subgroups is set to:

$$I = \left\lfloor \sqrt{T/t_{min}} \right\rfloor ; \tag{4.1}$$

initial cutpoints for the coarse-grained partitions are determined randomly but subject to the length constraints (Equations 3.36 and 3.37), and initial fine-grained partitions are set by first selecting two successive points from a random permutation of available coarse-grained cutpoints as the start and end points and then selecting fine-grained cutpoints randomly, subject to the constraint given in Equation 3.38. This initialization scheme is designed, subject to the amount of data available, to produce a wide variety of initial fine-grained partitions, in terms of both size and location, which also cover the entire data set. On average the length of the initial fine-grained partitions decreases with increasing κ. Note that initialization is in effect a constrained a priori multiobjective optimization process in itself, given the need to to balance variety and total coverage of the fine-grained partitions and obey all length constraints, and in practice Algorithms 4.1 and 4.2 require careful programming to ensure valid results, especially if solutions are sought entailing large numbers of partitions.

Initialization of the population for an EA is sometimes no more than the generation of random numbers, but here the necessity to obey the constraints, set up coarse-grained cutpoints within the different subgroups, and to then set up valid fine-grained cutpoints for each set of coarse-grained cutpoints leads to more intricate programming and quite a high computational burden, though fortunately the setup only takes place once. Although the necessary code is not large, in fact this and the specialized crossover were the most difficult and time-consuming sections of code to get right when constructing the specialized MOEA. After problems in early versions with highly variable and sometimes

large numbers of infeasible individuals generated at various points during the operation of the MOEA, the decision was taken to minimize the number of infeasible individuals at all stages as far as possible. For initialization, this meant designing Algorithms 4.1 and 4.2 in a way that guarantees all of the initial population was feasible, as well as meeting the restrictions imposed by the island structure, whilst still producing an initial population that was as diverse as possible, by making all feasible individuals possible to generate. Key to this was to effectively subdivide all the available points in the input time series, whatever the length, according to the number of regimes required but to make sure the slack space - that is, the remaining points left at the beginning and end of the time series, if the number of regimes does not exactly divide the number of points - is available for deciding the first and last cutpoints, in a random, unbiased way.

4.2.2 Fitness function and invariance properties of correlation

We will now consider the suitability of metrics of the type developed in Section 3.3 for developing a fitness function for use in the MOEA. For the purpose of summarizing the distributional properties of a given subseries, variance-covariance has the advantages of familiarity and well-understood properties, and as explained in Section 3.3, realized covariance has additional advantages in terms of simplicity and speed of computation. Furthermore, there are specific properties of invariance in respect of the standard coefficient of correlation between two sets of subseries variances-covariances that will prove most useful in relation to the value of our fitness function in assessing similarity and difference under affine transformations.

Recall firstly the basic property of the correlation coefficient that

$$\operatorname{corr}(a_1 x_1 + b_1, a_2 x_2 + b_2) \quad = \quad \operatorname{corr}(x_1, x_2), \tag{4.2}$$

provided that $a_1 a_2 > 0$. Now let X and Y be sets of observations, with the latter an affine-transformed copy of the former, and $\rho_{X,Y} = \text{corr}\left(\boldsymbol{\sigma}_X, \boldsymbol{\sigma}_Y\right)$ be the correlation between the set of m realized variances and $m\left(m-1\right)/2$ realized covariances for X and Y, stacked into column vectors $\boldsymbol{\sigma}_X$ and $\boldsymbol{\sigma}_Y$ as defined in Equation 3.29 ; then from Equation 4.2,

$$\text{corr}\left(\left[\mathbf{a} \cdot \boldsymbol{\sigma}_X + \mathbf{b}\right], \left[\mathbf{c} \cdot \boldsymbol{\sigma}_Y + \mathbf{d}\right]\right) = 1, \tag{4.3}$$

where $\mathbf{a}, \mathbf{c}$ and $\mathbf{b}, \mathbf{d}$ are non-zero real-valued vectors. This leaves $\rho_{X,Y}$ invariant to affine transformations. In particular, $\rho_{X,Y}$ is invariant to stretching or squeezing in either the frequency or amplitude domains. Note that since we are dealing with log differences of the form $s_{jt} = \ln x_{it} - \ln x_{jt-1}$, then with regard to the original data, applying a translation will change the slope of the time series trajectory, whilst applying a scalar linear transformation $\mathbf{s}_j \mapsto a_j \cdot \mathbf{s}_j$ is equivalent to applying a nonlinear, power law effect to the original observations.

These properties are inherited by the metrics used in the constrained objective (Equation 3.35). In the case that successive coarse-grained subseries in a given partitioning are all identical except for affine transformations of the types specified, then $g\left(\mathbf{d}_{K_k,K_{k-1}}\right) = 1$, whilst if successive fine-grained subsubseries are all identical to their reference coarse-grained subseries except for affine transformations of the types specified, then $h\left(\mathbf{d}_{F_k,K_k}\right) = 0$. More importantly, in the data if successive coarse-grained subseries are close but for such affine transformations, in other words the underlying DGPs are quite similar, then the objective function f_1 should have values close to 0, whilst if the underlying DGPs are quite different, objective values will approach 1 as the correlation approaches -1. For f_2, the value will approach 1 if the correlation of the relevant pair of coarse-grained and fine-grained subseries approaches -1, and will be close to 0 if they are quite similar.

Hence our biobjective fitness function will be that given in Equation 3.35, with the summarizing functions defined as per Equations 3.32 and 3.33, and

the constraints given in Equations 3.36, 3.37, and 3.38 are dealt with as far as possible by obeying these constraints at all stages of the programming.

4.2.3 Fitness function evaluation and the set of Pareto fronts

After an initial calculation of T vectors $\boldsymbol{\sigma}_t$, each containing sets of $m\left(m+1\right)/2$ stacked realized variance and covariance entries $RC_{ijt}^{(n)}$, fitness function evaluation using the form outlined above requires for each subgroup with population N_g at each generation, only κ summations of vectors, as per Equation 5, which is far simpler in computational terms than recalculating covariance matrices repeatedly. It is also guaranteed that all individuals in the population at all generations will be feasible, provided all the original $\boldsymbol{\sigma}_t$ have valid entries.

We can then find the Pareto front for each subgroup. For a minimization problem with ω objectives, let M be the set of points $\mathbf{v}_i^{min}$ out of N total which have the minimum values for each individual objective f_i; that is:

$$M = \left\{p_i^{min} = \arg\min\left\{f_i\left(p_j\right)\right\}, \right. \tag{4.4}$$
$$\left. i = 1 \ldots k, j = 1 \ldots N\right\}.$$

Define Pareto dominance in the usual fashion:

$$p_A \succ p_B \Longleftrightarrow \tag{4.5}$$
$$f_i\left(\boldsymbol{p}_A\right) \geq f_i\left(\boldsymbol{p}_B\right) \forall f_i \textbf{and} \tag{4.6}$$
$$\exists f_i | \left[f\left(\boldsymbol{p}_A\right) > f_i\left(\boldsymbol{p}_B\right)\right].$$

This leads to the definition of the nondominated set:

$$PF = \left\{p \in \Omega : p \nprec p' \forall p',\ p \neq p'\right\}. \tag{4.7}$$

It follows that

$$\|p_j\|_1 > \|\sup M\|_1 \iff p_j \notin PF, \tag{4.8}$$

where $\|p\|_1$is the taxicab norm of the objective function values, that is

$$\|p_j\|_1 \quad = \quad \sum_{i=1}^{k} |f_i(p_j)|, \tag{4.9}$$

and $\|\sup M\|_1$ is the norm of the lowest values for each objective of any of the points in M; all such points must be dominated by at least one point in M, and we can immediately eliminate them from consideration as members of the PF. In the biobjective minimization case, this then implies that if we first take the point with the lowest global value of f_1, all points with a higher value for f_2 than that point can be excluded; we then find the point with the next lowest value for f_1 from the points not dominated by the first point, and so on.

We do not need to sort the points at any stage, only find the suprema of successively smaller nondominated sets. Taking the number of points on the final PF as n_{PF} and the number of remaining nondominated points at each iteration as $m_i^{\not\prec}$ the complexity is thus

$$O\left(m + \sum_{i=2}^{n_{PF}} m_i^{\not\prec}\right) \quad \leq O(n_{PF} \cdot M). \tag{4.10}$$

This compares favourably with the $O\left(kM^2\right)$ complexity of exhaustive algorithms. Because and all individuals are unique, and because this is a biobjective problem using continuous values, meaning we can assume that all values for each objective are unique for the individuals in a given tournament, we can use a particularly fast and simple algorithm to find the PF; Algorithm 4.3 works as a simplification of Algorithm 2.1, and does not need to calculate norms.

Each subgroup thus yields its own PF at each generation and these are stored and added to the next generation only for the purpose of calculating the new PF, which will change only if new points are found that dominate points on the old PF so that the location of the PF changes and the number of points

Algorithm 4.3 Fast biobjective Pareto front algorithm

Step 0:initialize with m pairs of fitness function values $\{f_1, f_2\}$;
Step 1: find and archive the point p_1^{min} with the lowest value for f_1;
Step 2: eliminate all points with a value for f_2 more than or equal to $f_2\left(p_1^{min}\right)$;
Step 3: loop to Step 1 whilst any points remain;
Step 4: return the PF.

thereon may shrink, or if additional nondominated points are found, so that the number of points on the PF grows. We can apply Algorithm 4.3 a second time to these points if we wish to find a global PF for the union of populations all the subgroups, and the result will be the same as if we had calculated the global PF directly from this total population, but a further advantage of the subgroup approach is that computation is generally faster if we compute in two stages. However the union of the subgroup PFs is itself of interest, as discussed in Subsection 5.4.2.

Although the nature of the fitness function and the method devised for its calculation mean that successive covariance calculations can be greatly simplified, the calculations are still highly computationally intensive. It was here that he greatest gains were seen from experimentation with the use of GPUs for the comparatively simple calculations (in essence, mostly vector addition), although this was not implemented in the final version for reasons discussed in Appendix A. Infeasible individuals are also checked for and eliminated from the population after fitness function evaluation. After improvements to the initialization, crossover and mutation algorithms, which were found to be the main sources for the creation of infeasible individuals, error logs created in the MOEA revealed generally modest numbers of such individuals, generally zero to at most a low single digit percentage of the population in each island subgroup, which was considered acceptable, and the problem is not compounded because such individuals are removed and not passed to genetic operators.

Algorithm 4.4 Permuted tournament selection

Step 0: initialize with objective values of current population, including those on the PF;
Step 1: randomly permute the current population;
Step 2: Take the next $\hat{\tau}$ of $\check{N}_t$ remaining individuals in the permutation, where $\hat{\tau}$ is uniform random $\in \left[\sqrt{\check{N}_t}, 2 \cdot \sqrt{\check{N}_t}\right]$; use Algorithm 4.3 to find non-dominated individuals, and add copies of these to the list to be passed to genetic operators;
Step 3: while individuals remain in the permutation which have not competed, loop to Step 2;
Step 4: whilst the list of copies of individuals to be passed is smaller than the twice the desired population N, loop to Step 1;
Step 5: if necessary, select at random from the nondominated individuals in the final tournament;
Step 6: Return in total $2 \cdot N$ selected individuals.

4.2.4 Permuted tournament selection

Selection is performed per Algorithm 4.4; note that the number of participants in each tournament, $\hat{\tau}$, is randomized so that selection pressure [19] for each subgroup at each generation is also random. Individuals may compete in more than one tournament and all individuals will compete at least once, and individuals in the population PF found using Algorithm 4.3 will win all tournaments in which they take part. The use of complete permutations of the population also removes the chance that elite individuals fail to participate in any tournament and obviates the need for explicit elitism. Note also that elite individuals are archived and hence will remain nondominated until the MOEA terminates unless new individuals emerge that dominate them. Finally, note that in this implementation, tournaments are performed sequentially with all nondominated individuals from each tournament passed, up to $2 \cdot N$ individuals; only the final tournament potentially requires a tiebreaker.

The mathematical implications of this choice of tournament selection algorithm are discussed in Section 4.4.

The mathematical arguments underpinning the choice of permuted biobjective tournament selection are discussed in Section 4.4, as well as in various discussions in Chapter 2. From a computational point of view, the advantage of avoiding explicit calculations of rankings is that as noted in Section 3.2, ranking

algorithms have high complexity and avoiding calculating complete rankings at each generation, or even updating them (which in the case of the specialized MOEA, would not save time as all the individuals change at each generation), can save a great deal of computation time. Instead, Algorithm 4.3, which determines biobjective nondominance and is in turn called by Algorithm 4.4, the tournament selection routine, has only to determine nondominance amongst $\hat{\tau} \in \left[\sqrt{\check{N}_t}, 2 \cdot \sqrt{\check{N}_t}\right]$ individuals, in a very simple way. This algorithm was also used to produce the larger PF used for determining which elite individuals across the whole population should be archived. The simple *while* loop structure used in coding was found in testing to outperform versions using recursive programming and *arrayfun*, as well as optimized code found in online forums. The key to its success is that it progressively eliminates many points from consideration, but it was found to be as fast as even simpler alternatives for comparing just two points in tournament selection.

4.2.5 Permuted affine crossover

Crossover is performed as per Algorithm 4.5. All individuals are subject to crossover and existing individuals are passed intact only if two copies are sampled for the same tournament, although individuals on the PF are ultimately recorded and stored; hence there is no parameter associated with crossover probability. The spacing conditions referred to are those used in Algorithms 4.1 and 4.2. In practice, programming Algorithm 4.5 in such a way that each new individual is guaranteed to have a feasible pair of sets of cut points obeying all constraints may be very computationally expensive and it may be necessary to allow through a certain number of infeasible individuals to be caught and eliminated by error checking later. The effect of combining sets of cutpoints in this way is to stretch or squeeze the coarse-grained subseries and fine-grained sub-subseries in a way tantamount to applying offsetting affine transformations in the time domain so that the new set of cutpoints does not change the length or location of the specialized subperiod on which the fine-grained cuts are determined.

Algorithm 4.5 Affine crossover

Step 0: initialize with list from Algorithm 4.4;
Step 1:randomly split list into 2 parent lists;
Step 2: take the next parent from each list; if both parents are copies of the same individual, add an untransformed copy of that individual to the next generation and move to the next entry pair;
Step 3: merge the sets of coarse-grained and fine-grained cut points from each parent;
Step 4: check which cutpoints in the combined lists (if any) violate no spacing conditions;
Step 5: randomly remove one cutpoint not included in those found in step 4 (if any) or otherwise randomly remove any one cutpoint;
Step 6: loop to Step 4 until the required number of cut points for the number of subperiods κ is reached and all spacing conditions are met;
Step 7: if left with insufficient cutpoints, randomly add back deleted cutpoints which do not violate spacing conditions;
Step 8: loop to Step 1 until all pairs of parents have been subject to crossover;
Step 9: return the new population.

It is worth noting that the crossover scheme implemented in Algorithm 4.5 is highly specialized and it is not possible to replicate its operation using general purpose EAs. Employing a more standard crossover scheme, even with many constraints, results in an unacceptable number of infeasible individuals at each generation.

Although Algorithm 4.5 was coded to be able to handle a situation where two copies of the same individuals, one in each permutation, are used as parents, in practice this rarely happens and because the parents are removed and replaced by offspring at each generation, multiple copies of any individual do not propagate as a result. Two separate but similar algorithms are run to create first the new coarse cutpoints for each individual and then the new fine cutpoints; in both cases, the resultant new child individuals are checked for violation of spacing conditions, and repair is attempted; if all else fails, some possibly infeasible are allowed through but as mentioned above, if they are indeed infeasible they will be caught at the next generation when fitness values are calculated, and in practice the number was small in experiments run. The alternative would be to continue to try to generate feasible individuals, leading to a still more computationally expensive crossover phase with computation time that is highly

Algorithm 4.6 Mutation

Step 0: initialize with new population generated from Algorithm 4.5;
Step 1: randomly select individuals based on the mutation probability (4.11);
Step 2: randomly select one cutpoint for each selected individual, delete these cutpoints and replace with new valid cutpoints subject to length constraints; for the fine-grained partitions, allow new cutpoints up to t_{min} before and after the original range of the partition;
Step 3: loop to Step 2 until mutation complete;
Step 4: return the new population.

variable and difficult to predict. Even so, after fitness evaluation, crossover as implemented was still the second most computationally expensive part of the MOEA.

4.2.6 Mutation

Mutation is implemented using Algorithm 4.6A simple random point mutation is applied to the whole representation of the individuals in each generation, so that either either the coarse-grained and fine-grained part of the representation is affected. If an individual is selected for mutation, one cut point is deleted and another one randomly inserted in such a way that spacing conditions are again met. A difference with crossover is that a small time window of length t_{min} is added to the beginning and end of the list of valid locations in the specialization subperiod for the purpose of randomly determining the location of the new cutpoint, so that it is possible for the fine-grained sub-subseries to migrate in the time domain. The mutation threshold probability should be set quite low, so that most mutation will typically still occur within the original specialization parameters, and this migration is typically slow and not of great magnitude unless the number of generations is large. This migration is the only mechanism by which the original specialization locations can be changed, but the algorithm needs to prevent the tendency for the scale of the fine-grained subseries to simply expand over many generations towards the scale of the coarse-grained subseries, leading to favouring of individuals with fine-grained subseries almost identical

to their coarse-grained subseries. In the algorithm, the mutation rate is fixed as

$$mutRate = \frac{I}{N}, \tag{4.11}$$

so that on average just one population member per subgroup will be subject to mutation in each generation. Note that the number of subgroups I is itself a function of the length of the dataset, per Equation 4.1.

During development, experiments were conducted using an adaptive mutation rate, but this only proved effective when the number of generations was very large, making use impractical; hence the rate was fixed, as noted in Subsection 4.2.6. Feasible individuals after mutation are guaranteed by only allowing change of a single cutpoint within a range allowed by spacing conditions. This also guarantees changes to the representation through mutation are small, though the effect on fitness values may not be. In comparison to crossover, code for mutation is comparatively simple and its computation burden low.

4.3 Parameters, parallelization and the specialized nature of the MOEA

A notable feature of the construction of the EA is that it has very few choice input parameters; in fact the only choices are:

- N, the total population size;
- G, the number of generations;
- t_{min}, the minimum number of observations in any subperiod.

In practice, given limited computational resources, whilst results can be expected to improve with higher N and G, the first is limited by available memory and if parallel processing is used, the available number of cores, and the second by processor speed and the number of cores used. The third parameter t_{min} can generally be left to the minimum meaningful value of 2, as it was in all the

studies described in Chapter 3, but might need to be adjusted for very large and complex problems if it is found that too many infeasible individuals are generated. This problem did not come up in testing as described later in Chapter 5, however.

On the subject of parallelization, in general the specialized MOEA shares attributes with many other MOEAs, namely that:

- Multiple runs can be executed in parallel, with the same or different parameters;
- Many operations within a single generation can be parallelized;
- The whole set of operations on a given subgroup within a single generation can be parallelized;
- Generations themselves cannot be parallelized within a given run, by the fundamental nature of the evolutionary process.

The first and fourth aspects are standard and require no further explanation. As to the second, all of the key algorithms as set out in Algorithm 4.7 below for each generation can be parallelized. Furthermore, certain aspects, such as the evaluation of the fitness function and tournament selection, are also tractable to GPU computation by virtue of their simplified nature, though this was not implemented for final testing, as explained in Subsection A.3.3. Finally as to the third, operations on different subgroup can be entirely parallelized across generations, unless any adjustment to the groups is required from generation to generation, for example changing population sizes in an adaptive manner; such intergenerational adjustments were not implemented in the MOEA.

As noted, the algorithm permits only 3 choice input parameters, and the values used for these is generally driven by practical matters including available time and computational resources as well as the size and complexity of the data used, rather than any speculation on the part of the user about what parameter settings might produce the best results. For this reason, if multiple runs are

conducted, it should only be to increase confidence in the optimality of results, and it may well be that better results are to be obtained by using available computation time and resources to increase N and G, rather than performing more runs, with the caveat that this may also depend on the nature of the data.

Otherwise as regards selection and genetic operations, the use of permuted biobjective tournament selection (Subsection 4.2.4) means that all individuals in each generation are sampled for at least one tournament and have the same chance of being sampled for more than one tournament, depending on the number of victors produced by the initial permutation. Rather than setting an arbitrary number of participants per tournament (many convergent algorithms use $\tau = 2$ to maximize diversity), $\hat{\tau}$ is randomized, and as a consequence so is selection pressure. In crossover (Subsection 4.2.5), individuals are passed as is only if two copies are assigned as parents in a particular pairing, and all individuals passed from tournament selection are assigned as parents, so there is no crossover probability parameter. Finally, the mutation probability is determined by population size and length of the dataset (Equation 4.11).

It is worth emphasizing that the MOEA is highly specialized in several respects, most notably in the initialization, the type of fitness function and its method of calculation, the affine crossover algorithm and the design of mutation to avoid infeasible individuals. Some of these innovations can be expected to improve performance for this very specific task against other types of algorithms, but others, in particular the crossover and mutation algorithms, are designed to avoid the population being swamped with infeasible or very low fitness individuals, which is what should be expected if general purpose EAs are used for this problem. This, together with the issue of how to set the many parameters general purpose EAs generally require, makes comparison with existing EAs very tricky in practice, and for these reasons such comparisons were not conducted.

The entire specialized MOEA is summarized as Algorithm 4.7.

Algorithm 4.7 MOEA for partitioning self-affine multivariate time series

Step 0: load $(T-1)\times M$ array of log differences of the multivariate time series **S**. Generate initial population divided into subgroups using Algorithms 4.1 and 4.2;
Step 1: calculate fitness function and find subgroup PFs using Algorithm 4.3;
Step 2: perform tournament selection using Algorithm 4.4;
Step 3: perform crossover using Algorithm 4.5;
Step 4: perform mutation using Algorithm 4.6;
Step 5: loop to Step 1 until convergence conditions met or maximum number of generations reached;
Step 6: return the new population.

4.4 Implications of the use of permuted multiobjective tournament selection with randomized tournament size

In Algorithm 4.4, the selection scheme without replacement differs from Scheme C described in Chapter 2 in several ways. Firstly, since no ranks or rankings are explicitly calculated, nondominance between the participants in each tournament is calculated. However, as nondominance is calculated in the same way as that underlying Algorithm 2.2, the results in all cases will be just as if a ranking had been calculated; that is to say, the resultant PF will contain individuals only of the same rank, which is the highest rank of any participant in a given tournament. Secondly, the number of participants for all tournaments at each generation is now a random number $\hat{\tau} \in \left[\sqrt{\tilde{N}_t}, 2\cdot\sqrt{\tilde{N}_t}\right]$, so that $\hat{\tau}$ unique individuals are sampled for each tournament from the same permutation. There are never any copies of the same individual in the specialized MOEA, by design, and there are generally no individuals with all identical fitness values. Thirdly, as there is no tiebreaker used, all individuals of highest rank in a given tournament are passed to genetic operators, meaning the number of tournaments conducted $\hat{T}$ is also a random number. Finally, as the range of $\hat{\tau}$ depends on the size of the remaining population in a permutation, the expected tournament size decreases as the algorithm runs through the permutation, but then resets as a new permutation is calculated, until $2\cdot N$ individuals in total have been

passed.

Elite nonsampling is not an issue due to the use of permutations. Also, all individuals are passed to genetic operators after selection, fitness values are continuous, and the initial population does not contain multiple copies, we can assume that all individuals are unique before selection at each generation, so the individual multisampling issue does not require consideration. Furthermore, although no ranks are calculated, the in-rank multisampling issue is also not a concern, because all individuals of the highest rank in a given tournament are passed apart from in the final tournament, where some random selection may be necessary. Hence we can assume, from the analysis in Chapter 2, that the elite population can be expected to increase by a factor related to the distribution of $\hat{\tau}$.

It is also worth noting that a consequence of Theorem 2.3 is that for each rank below the first, there is an optimal tournament size that will maximize the rank size after selection, but this optimal tournament size changes at each generation. Hence if tournament size is randomized, any distributional parameters such as mean and variance would need to be chosen with care, perhaps by parameter tuning, or else by adaptation; the choice of the uniform distribution for $\hat{\tau}$. largely avoids these complications. Finally, we should also note that as with Scheme B, Equation 2.31 also applies, so that individuals of the lowest fitness levels can never win a tournament unless there are more of them than the tournament size, meaning that such individuals will never be passed to genetic operators or to the next generation.

To summarize, the selection scheme described here has the following advantages:

- the nonsampling probability is zero for all individuals, so that elite nonselection is not an issue;
- the rate of increase of elite individuals is well described for large populations;

- the scheme features neither individual nor in-rank multisampling.

However, diversity is to a greater or lesser extent decreased, and selection pressure increased, at least in comparison to Schemes A and C as described in Chapter 2, because low-ranked individuals may be automatically eliminated. Given that as coded, $\hat{\tau} \in \left[\sqrt{\check{N}_t}, 2 \cdot \sqrt{\check{N}_t}\right]$, selection pressure is deliberately high to balance the nonconvergent nature of the MOEA given that all selected individuals are passed to crossover, and the lack of explicit elitism in the algorithm (though elite nonsampling is not an issue due to the use of permuted tournament selection). An additional consequence of the average number of participants in each tournament being quite high is that low-ranked individuals are all the more likely to be automatically eliminated.

4.5 Summary

The novel approach to creating an MOEA to partition multivariate self-affine time series, as described in this chapter, makes only the simplest of assumptions about the DGP and uses only 3 input parameters, two of which, the population size N and the maximum number of generations G, relate to the computational structure of the MOEA rather than being model parameters as such, with the third, the minimum partition size t_{min}, in practice being set to the effective minimum value 2 in all experiments. The price paid is that the starting computational complexity is very high. To address this, a highly parallelizable population-based evolutionary algorithm was developed, which reduces the problem to a biobjective one using objective functions based on the correlation of realized covariances for successive coarse-grained subseries and for fine-grained sub-subseries with the coarse-grained subseries.

The summarizing approach developed in Chapter 3 significantly simplifies the problem, yet still yields a set of solutions for analysis a posteriori rather than a single solution based on a priori objective weightings, say, which would yield much less in terms of insights into the fitness landscape.The population is

split into subgroups specialized to examining fine-grained partitions in differing sections of the time series, and all possible partition sizes can be investigated.

The algorithm uses biobjective permuted tournament selection with randomized size; a crossover method that in effect applies affine transformations where necessary to fit together elements of each parent's representation; and a mutation method developed to maximize the number of feasible individuals after operation. The mathematics of the choice of tournament selection scheme imply that the nonselection probability for elite individuals is zero, obviating the need for explicit elitism.

In the next chapter, we describe testing of the specialized MOEA with simulated and real data.

Chapter 5

Testing with Simulated and Real Data

5.1 Introduction

Testing the MOEA described in Chapter 4 presents particular problems. Underlying the choice and design of the MOEA are the assumptions that training data is not available, and in particular, that it is difficult of impossible to definitively declare that a given time series should be partitioned in one clearly defined way; rather, we assume that there are many ways in which a self-affine multivariate time series can validly be partitioned, and so the MOEA is deliberately designed to be nonconvergent and to produce a cloud of valid solutions. This chapter describes the testing approach developed to cope with such a situation.

The principal hardware platform used in his project was a 3.7GHz, 4 core system with 32GB of RAM. During the development period, memory was upgraded to 64GB, to overcome problems with oversized arrays during testing, as well as to improve performance.

The rest of this Chapter is arranged as follows. Section 5.2 describes the high level algorithm used to test the MOEA and how it was implemented. Section 5.3

describes the formation of simulated self-affine multivariate time series via the technique of stitching together time series generated using fractionally integrated time series (FITS), the advantage of which is that we do know the points at which the different FITS have been joined, or stitched. The testing methodology is described, and results discussed. Section 5.4 describes the use of real financial data for testing and results are presented, with some interpretation.

5.2 Structure of the MOEA testing environment

To test the MOEA, a high level algorithm was employed to run the algorithm multiple times and create the large solution set described in Chapter 5. The Matlab code sets up cell arrays to handle the MOEA outputs - specifically, the sets of course and fine cutpoints for the final PFs from each run, and the associated objective values - and then calls the MOEA multiple times, using the specified parameters. The code for this high level testing algorithm contains a *parfor* loop, and is the only code used that calls CPU parallel processing, using a pool of 4 workers. It is generally best practice to use *parfor* loops at the highest nesting level[1], since the MOEA itself is run sequentially in each iteration on a separate CPU core, automatically allocated by Matlab. Testing cycles were very lengthy, reflecting the considerable complexity of the problem; for example, the testing of the simulated FITS data for 100 iterations took, depending on the other parameters and size of dataset used, between 47-53 hours per run of the testing.

This means that in the testing version, the MOEA itself does not explicitly employ parallel processing through use of *parfor* loops, although it still employs vectorization extensively. In the stand-alone version however, *parfor* loops are employed at the highest level of the MOEA itself, with each subgroup dealt with by a separate run of the MOEA, i.e. the MOEA is run as a PGA. An overview of the general structure of the specialized MOEA was shown in Algorithm 4.7. The

[1]This approach has the added advantage of avoiding all of the issues and pitfalls with *parfor* loops highlighted in the Appendix.

specialized MOEA in many respects conforms to the classical structure of the genetic algorithm, with initialization based on representation followed by iteratively running fitness evaluation, selection and genetic operator algorithms, with the PF recorded and the best individuals archived after fitness evaluation. One difference from the typical structure is that the initial population is segmented into subgroups, and these subgroup populations are kept separate throughout the MOEA's operation, with PFs recorded and elites archived separately for each.

The output of the testing algorithm consists of arrays of the coarse and fine cutpoints and associated objective values for the PFs at each generation. Note again that the MOEA is deliberately non-convergent; the only stopping criterion is the passing of a given number of generations, and in general since objective values are convergent and all individuals undergo crossover, all individuals are unique at each generation, which maximizes diversity but means the MOEA would not terminate in the absence of a generation limit, even if the true PF were found and archived. A combined PF from all individuals in all islands is also calculated, and this is the level at which archiving of elites takes place. Since this is strictly implicit elitism, the elites are only archived; they are not reintroduced into the population, and multiple copies cannot propagate.

5.3 Testing with simulated data

5.3.1 Formation of the simulated data

Testing with real multivariate data, especially of higher dimension, is problematic as in general we do not know how to partition the data, this being precisely the problem the techniques developed in this chapter are designed to address. A first step is to formulate a set of data we know more about, specifically one in which we know that the data inherently displays self-affinity. The approach used in testing is to "stitch" together subseries which are individually self-affine and which repeat patterns from one subseries to the next, but which are each

sufficiently different to the preceding subseries in terms of the coarse-grained metric (Equation 3.32) that the MOEA can detect a change in subseries.

The fundamental building blocks of this approach are multivariate fractionally integrated time series (FITS), which use FBM processes in their construction. The approach used is the p-model set out in [41], and the implementation is adapted from [159]. The p-model itself produces only stationary random time series and has one parameter, $p \in [0, 1]$, which is associated with increasingly peaked series as it approaches 0 or 1, and calmer series as it approaches 0.5. To create nonstationary series, the result of the p-model is filtered in Fourier space and a further slope parameter is specified; slopes flatter than -1 are called stationary in [41], whilst slopes between -1 and -3 are called nonstationary, with stationary increments. These nonstationary cases are at least continuous, but not differentiable. Slopes steeper than -3 are nonstationary and differentiable.

The approach in constructing multivariate test series with differentiated subseries is as follows. Firstly, for each of κ subseries as required, m "master" series of length $L \cdot T_k, k = 1 \ldots \kappa$ and random parameters $p \in [0.25, 0.49]$ and $slope \in [-3, -1]$ are generated, the random parameters recorded and log differences taken and the series "stitched" together so that the total length is $L \cdot T$. The parameter ranges are set so that the series are neither too peaked nor excessively smooth and are nonstationary. Next, for each subseries $k = 1 \ldots \kappa$, m FITS of length T_k are generated for each subseries, again each with independent uniformly random parameters but with the random seed reset to the same state s recorded before the first series was generated. Hence the random numbers used are the same for each FITS, but the parameters are different, such that the correlations between different FITS vary randomly. Log differences are taken and scaled down by a factor of $L^{1/\alpha}$, where α is a scaling parameter, and each t-th log difference from these series is added to the $t \cdot L$-th log difference of each of the m FITS. The effect is to add an additional shock to the beginning of each period. This might be thought of for example in the context of financial returns series, as reflecting new information available at the beginning of each

trading day, and is meant to simulate the phenomenon of price jumps at the beginning of the trading day which is commonly observed in financial data. Finally the entire system is normalized to be non-zero and have the same starting values.

It is not of course guaranteed that the stitched FITS generated by such a process will have subseries that are sufficiently distinct, in terms of the serial correlation of the realized covariances of the constructed coarse-grained subseries, to be useful in testing the MOEA. Therefore, suitable FITS were found by a Monte Carlo process, selecting FITS with sufficiently distinct subseries. With small values of T, it is not hard to find series with significantly negatively correlated subseries, but with larger values of T it becomes difficult to find series with maximum correlations between successive subseries that are any lower than a small positive number (less than 0.1, say). This means that the task faced by the MOEA will not necessarily be easy, as the original cutpoints in the construction of the simulated data are hard to identify. The actual stitched FITS generated and used in the first study study with just described below are illustrated in Figure (5.1), with actual observations plotted in the upper window, and log differences in the lower window.

5.3.2 Results from the MOEA for simulated data

In an initial experiment the generated FITS comprised $m = 8$ series with 2 central partitions and a further 2 incomplete subseries at the beginning, so that $\kappa = 4$, with a total length of $T = 512$ periods, each comprising $L = 32$ observations, for 16,384 high frequency observations in all. Because of the difficulties in constructing test data with significantly negative serial correlations between subseries, especially with larger numbers of periods, correlation coefficients observed in data used tend to be slightly negative and close to zero (typically in $[0, -0.1]$. The MOEA was run over 15 generations with a population size of 10,000 individuals for each of 11 subgroups. All testing was conducted on a 3.7GHz, 4 core system with 32GB of RAM. Earlier experimentation had shown

that such population sizes could lead to relatively rapid convergence with reasonable results, whilst smaller population sizes tend to lead to convergence after more generations but with poorer results, though possibly with lower overall runtime. The number of series in the multivariate system was chosen so that sufficient covariances were generated to make correlations meaningful but not so many that in particular the clarity of graphical representations is compromised. Computational complexity is also an issue but for reasons explained in Subsection 4.2.3, the choice of m affects only the $\kappa-1$ correlation calculations, generally in a sub-quadratic manner, and does not affect the underlying calculations of realized covariance at each generation. Hence this approach is potentially well suited to high-dimensional problems.

Two measures were considered to assess the success of the MOEA in finding suitable coarse-grained partitions. The first looks at the errors between the individuals with the best (i.e. lowest) values for the first objective f_1, i.e. the solutions with the highest dissimilarity between regimes, as assessed by the following formula:

$$\theta^{(1)}_{i,r,g} = \frac{1}{T \cdot (\kappa - 1)} \sum_{k=1}^{\kappa-1} \left| c^{EA}_{k,i,r,g} - c^{FITS}_{k} \right|, \tag{5.1}$$

where $c^{EA}_{k,i,r,g}$is the k-th cut point for a given subgroup, partition and generation, and c^{FITS}_{k} is the actual cut point used to generate the FITS; lower scores are better. The second is derived from our coarse-grained summation function (Equation 3.32) as follows:

$$\theta^{(2)}_{i,r,g} = \left[\sqrt{\frac{1}{4}\left(\overline{\rho^{FITS}} + \sup\left\{ \rho^{FITS}\left(s, s-1\right) \right\} + 2 \right)} - \sqrt{\frac{1}{4}\left(\overline{\rho^{*EA}_{i,r,g}} + \sup\left\{ \rho^{*EA}_{i,r,g}\left(s, s-1\right) \right\} + 2 \right)} \right], \tag{5.2}$$

that is, the difference between the coarse-grained summation function values

for the best individual generated by the MOEA and for the FITS cut points; again, lower scores are better. Note that these measures consider only the coarse-grained partitioning for a given solution.

The best results in terms of the first measure had a maximum error of 5 periods, i.e. an "error" of less than 1%, for any cut point (there were several individuals, distributed across different subgroups, which met this standard). The results demonstrate that the MOEA can find the "correct" cutpoints with a good degree of accuracy, although larger T and κ will lead to slower convergence or equivalently, lower accuracy for a given computational budget. One solution is illustrated in Figure 5.1[2], where the solid lines indicate the cutpoints in the original data and the dashed lines the coarse-grained cutpoints found by the algorithm.

A second simulated FITS dataset was then generated for the main experiment with $m = 8$ and $n = 32$ as before but with a much larger number of periods, $T = 4096$, or 131,072 observations in all, and $\kappa = 6$, so 4 internal partitions instead of 2; constructing the test data was significantly more computationally expensive, and this time the correlations of successive subseries tend to to be close to zero but slightly positive, typically in $[0, 0.1]$. The MOEA was then run for 100 generations with a total of 32 subgroups, each with a population of 10,000, and a total of 30 runs, and the results were aggregated by subgroup across the runs. Typical running times for this data were around 22 seconds per generation but this dropped to below 7 seconds when parallelization of subgroups across 4 cores was implemented, showing that parallelization is highly effective in reducing computation time.

Figure 5.2 shows an example result. In the upper plot, the solid black lines represent the coarse-grained cutpoints used to set up the actual dataset, whilst the dashed lines represent the cutpoints found in the MOEA solution. The plots of the data use log differences rather than the untransformed data, and this makes the distinctions between the partitions much clearer. In the lower plot of

[2]All figures are to be found at the end of this chapter.

Figure 5.2, only a zoomed-in section of the data is shown, and the dashed lines represent the cutpoints of the fine-grained partitioning found in this particular solution. Note that the relative lengths of the subseries in the fine-grained partitioning are quite different to those in the coarse-grained partitioning. It is also quite difficult to detect by eye, even with a system with just 8 different series, that the subseries in the fine-grained partitions correlate quite closely with those in the coarse-grained partitions, even though this is in fact the case; for a higher-dimensional system, visual interpretation becomes impossible, but for the MOEA, only a relatively modest increase in computational complexity is involved.

Figure 5.3 shows scatter plots of solutions for selected subgroups in terms of their values for $\theta^{(1)}_{i,r,g}$ on the x-axis and $\theta^{(2)}_{i,r,g}$ on the y-axis. The lines represent Pareto fronts for each of the subgroups shown; these PFs are not to be confused with the ones constructed during the running of the MOEA, which use Equations 3.32 and 3.33 instead of Equations 5.1 and 5.2, and so have no knowledge of the partitioning used to set up the simulated data. The best results from the MOEA show an error (as measured by $\theta^{(1)}$) of less than 2%, and manage to find reasonable fine-grained partitionings as well.

There is a clear trade-off between $\theta^{(1)}$ and $\theta^{(2)}$, but notice also that most individuals have a negative value for $\theta^{(2)}$, indicating that in terms of Equation 3.32, the MOEA finds coarse-grained partitions of subseries that are actually better differentiated than those used in the original construction of the test data. Also note that, although the MOEA has no knowledge of the partitioning used to set up the simulated data but which is used in the ex post calculation of $\theta^{(1)}_{i,r,g}$ and $\theta^{(2)}_{i,r,g}$ after the MOEA has run, at the last generation the MOEA has in fact found many more solutions close to the bottom left extremes of the plot, that is, solutions with, in particular, better solutions in terms of $\theta^{(1)}$, than was the case at the first generation. Note also that no single subgroup's PF completely dominates that of all others, though some are completely dominated; indeed, just 6 of the original 32 subgroups (only 10 of which in total are shown in the

plots, for clarity) have PFs containing points which are non-dominated with respect to all other points from all subgroups.

It is simple to extract from the several subgroup PFs for any given partition into κ subseries a single global PF. However, each subgroup PF is a valid subset of solutions on its own, and if not completely dominated by another subgroup PF, has valuable information about the solution space, given that each subgroup is optimizing over a different location of the fine-grained partition and potentially also a different scale. Similarly, with real data where the actual number of partitions is unknown, for each partition size $\kappa = 3 \ldots \kappa_{MAX}$, that solution subset is also valid in its own right, and in many cases we cannot say that one number of subseries κ forms a superior partition to another number; it is again operating at a different scale. Taken together, all the subgroup PFs for all the partition sizes κ may be thought of as forming a single solution set over a number of different scalings and fine-grained subseries locations; in other words an optimized sample of a much larger optimal solution set.

5.4 Testing with real data

5.4.1 Data source

We used the public access CRSP data[3] for second-by-second calculations of 9 capitalization and market indices calculated using intraday prices of US stocks over a 12-month period ending March 2016. As examples, CRSPSCT is an index of the total return of smaller stocks, whilst CRSPTMT is an index of total market return. To ensure greater stability of the realized covariance matrices and reduce noise, the second-by-second measurements were first aggregated into 5-minute bars and the daily covariance matrices were then calculated from these. The system is highly correlated, with correlations between series over the period varying between 0.0072 and 0.9985 but averaging 0.7718.

[3] Available at: https://wrds-web.wharton.upenn.edu/wrds/about/index.cfm

5.4.2 Results from the MOEA for real data

A total of 100 runs were effected for the data and the results, i.e. the sets of coarse-grained and fine-grained cutpoints with objective values on the PFs produced after 100 generations for each run, were recorded for each subgroup. This produced a large number of partitionings, all of which are potentially valid (as they are non-dominated). The MOEA was used to investigate partitionings with 3 and with 4 cutpoints (i.e. 2 or 3 regimes), and was run with a smaller population per subgroup (1000) and a smaller number of subgroups (11) than used in the experiment with simulated data, and 100 generations per run as before.

The progress of of the combined PFs for the whole population, aggregated from all runs and subgroups, is illustrated in Figure 5.4, with the objective values for $f_1$and $f_2$on the x-axis and y-axis, respectively. The algorithm shows progress generation by generation, with PFs from later generations often completely dominating those produced at earlier generations; this was also the case for the simulated data. Note that in later generations the f_2 values are extremely small (and so close to the axis) and this phenomenon is seen more quickly for the runs with 3 regimes; this provides strong evidence of the self-affinity of the real data set.

In order to cluster the sets of coarse-grained cutpoints obtained from all the PFs obtained from the MOEA, the k-means clustering algorithm was used[152, 116], with $k = \{2, 3, 4, 5\}$, running the algorithm 1000 times for each k and retaining the results with the lowest sums of in-cluster distances from centroids. We then calculated silhouette numbers [143] for each subgroup; 5.5 shows the means and variances for each subgroup, for both 2 and 3 regimes.The subgroups generally show similar patterns. For 2 regimes, the silhouette means do not vary greatly but variances increase with the number of clusters, indicating more low silhouette values. For 3 regimes, means decrease significantly for more than 4 clusters and variances also increase with the number of clusters. This indicates that a low number of clusters is most supported by the data, with in all

likelihood, just 2 clusters being best of all in the case of 2 regimes, whilst for 3 regimes, results are very similar for 2-4 clusters. Overall this implies that after 100 generations, results were already tightly grouped into a small number of clusters. In particular, the number of clusters is much smaller than the number of subgroups, indicating some convergence of results between subgroups.

The silhouette plots for different numbers of clusters for 2 regimes and for 3 regimes are shown in Figures 5.7 and 5.6; note that whatever the number of clusters, with 2 regimes there are 2 clusters of results that contain most of the results, whilst with 3 regimes a single cluster contains most of the results. Figures 5.8 and 5.9 for partitionings with 3 and 4 cutpoints respectively show the various CRSP index levels plotted against time, together with vertical lines indicating the centroids of the 4 clusters; each cluster has 3 cutpoints with the line colour and line type the same, which may be taken as representative of a particular type of solution with two defined partitions (which have corresponding fine-grained partitions) plus "incomplete" partitions before the first and after the last cutpoint.

With real data, we do not know the "true" partitioning, though with small systems we might be guided by visual cues, standard multivariate statistical techniques or some other known facts in guessing one or more valid partitionings; with large systems, such guesses may be impossible. If we look at the results as represented in the two figures without any prior knowledge, several observations can nonetheless be made regarding the representative partitionings. Firstly, the "incomplete" final partition of the data lying after each final cutpoint is in all cases quite large, with almost all final cutpoints indicated before the end of 2015 and some much earlier. This tells us either that later data the system is considerably different from earlier sections in terms of our coarse-grained summation function (Equation 3.32) or that it is easier to find similar fine-grained partitions in terms of our fine-grained summation function (Equation 3.33) for earlier data, or both. With this relatively low-dimension and highly correlated dataset, it is perhaps possible to note visually that there is a change

in the system in late 2015, but it is not feasible to check the similarity of coarse-grained and fine-grained partitions visually even with such a relatively small system. Secondly, the partitionings vary considerably in spread, i.e. the distance between the first and last cutpoints, so some solutions involve much larger "incomplete" first and last partitions than others, or looked at differently, the partitionings in effect operate on different time scales to one another; this is arguably a desirable feature in the context of our assumption that the system has scalable, fractal properties and is likely maintained by the island approach to segregating the EA population. Finally, the greatest concentration of cutpoints is in the August-November period, this being particularly noticeable in the second figure, and this is perhaps indicative of a greater change in the system covariances during that period.

5.5 Conclusion

Testing was conducted using both simulated data and real stock market data. The simulated data was constructed using generation processes known to be self-affine and designed to have as clear a partitioning as possible in terms of the main metric used by the MOEA to assess differentiation of successive subseries. Initial results have indicated that the MOEA may be able to come close to the partitioning used in the simulated data whilst simultaneously finding reasonable self-affinity, and indeed in limited testing was able to find partitions with better differentiation than the ones used to set up the test data. It was also observed that there is a clear trade-off between closeness to the original partition and the measured power of the differentiation between successive coarse-grained subseries, but that the overall solution set improved with successive generations. For the testing using real data, although it is not possible to comment directly on the accuracy of results as the "true" partitions are unknown, we were able to make several useful observations regarding the operation of the MOEA on the specific data set.

Figure 5.1: A typical close fit to the original subseries cut points

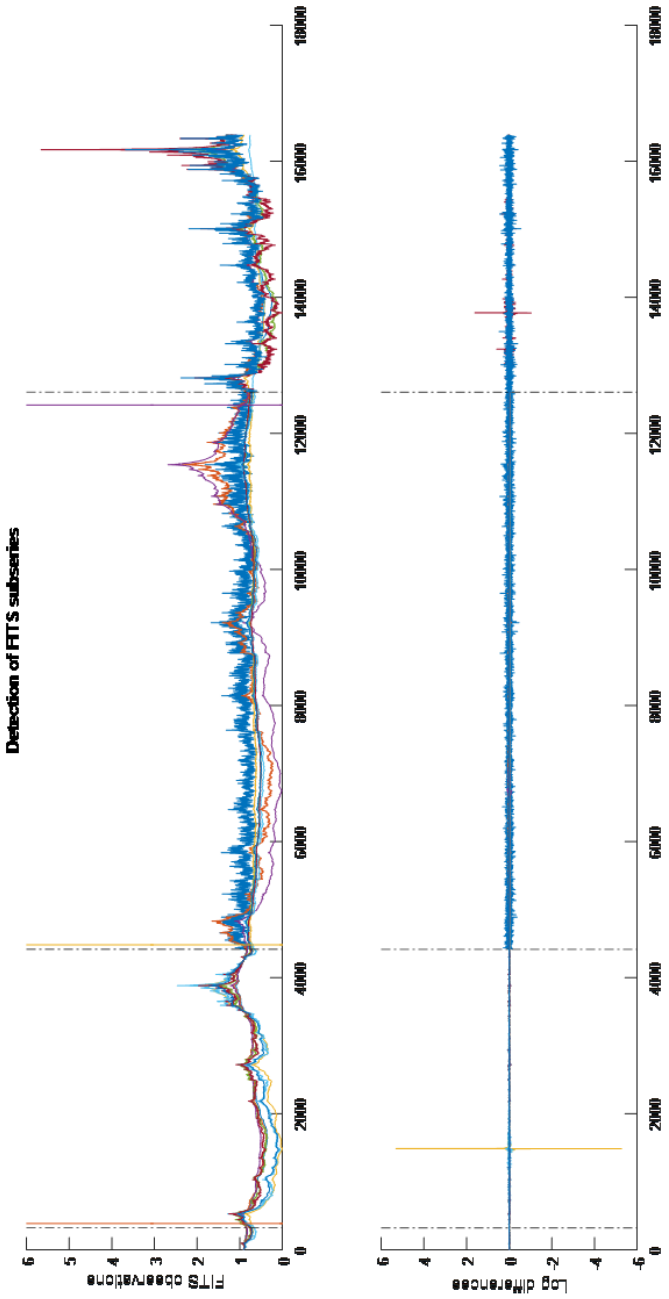


Figure 5.2: Coarse-grained and fine-grained partitionings

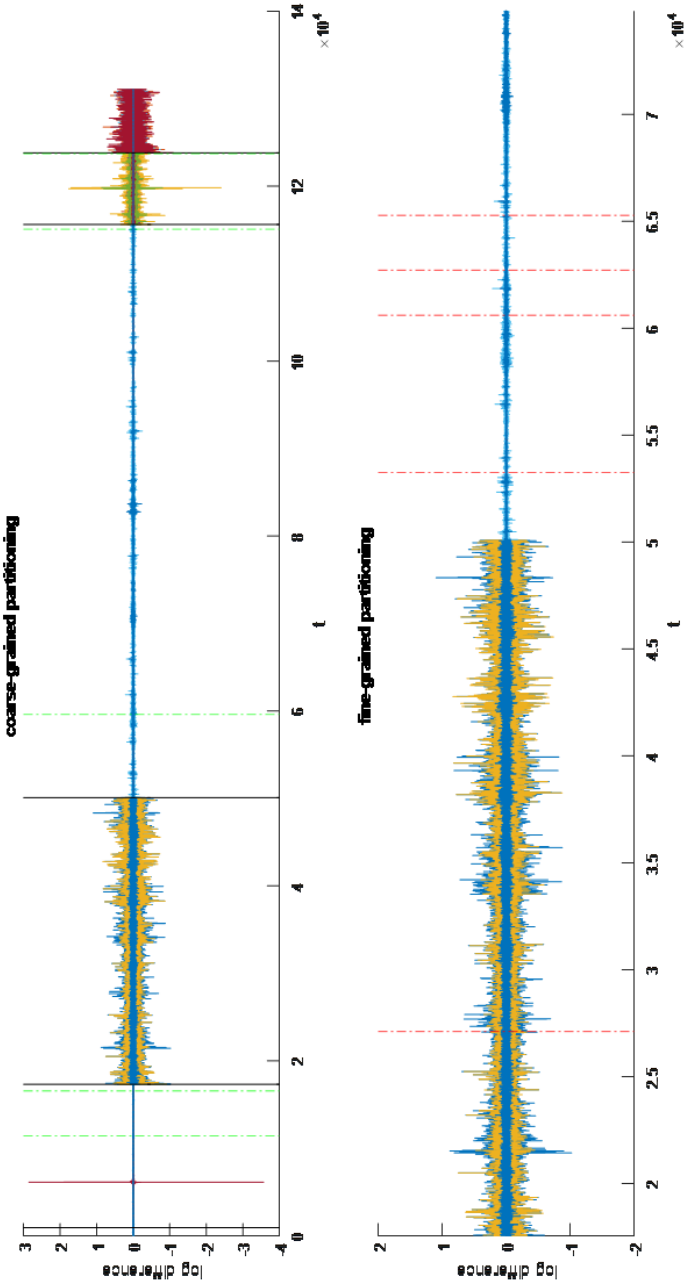


Figure 5.3: trade-off of theta scores

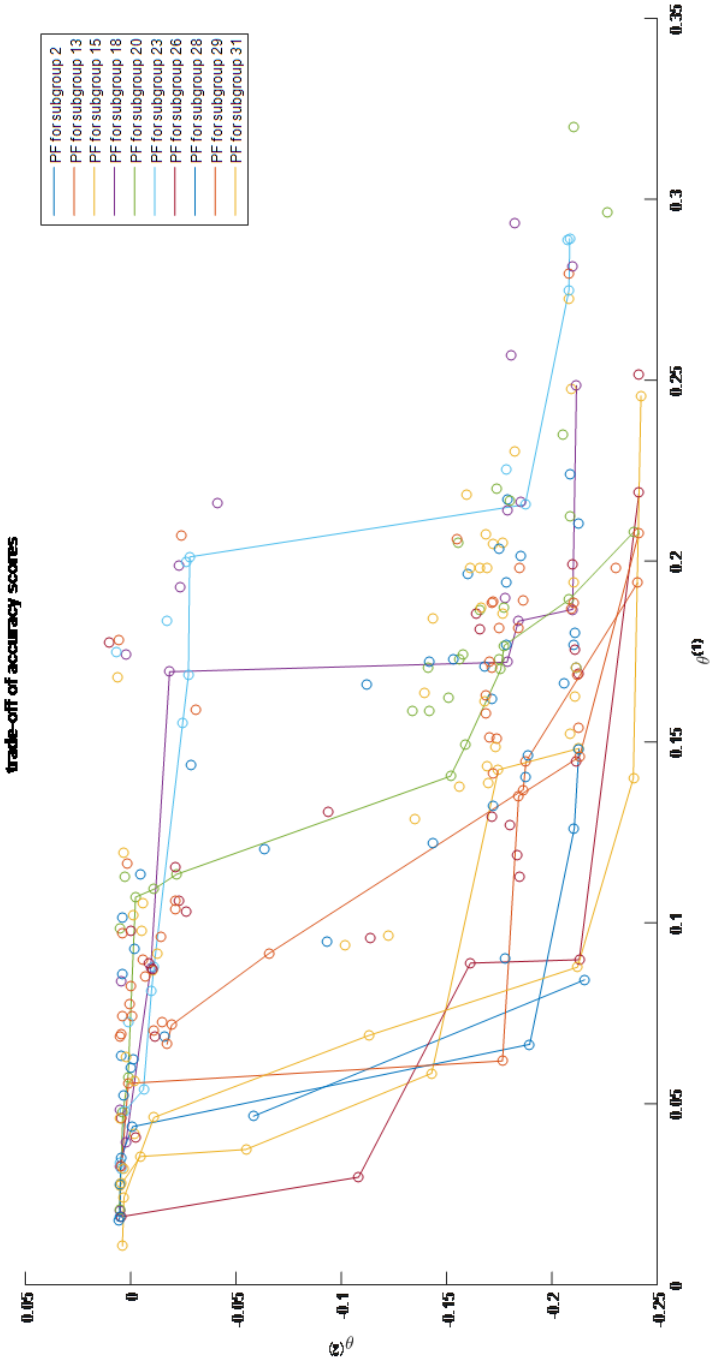


Figure 5.4: Progress of PFs by generation

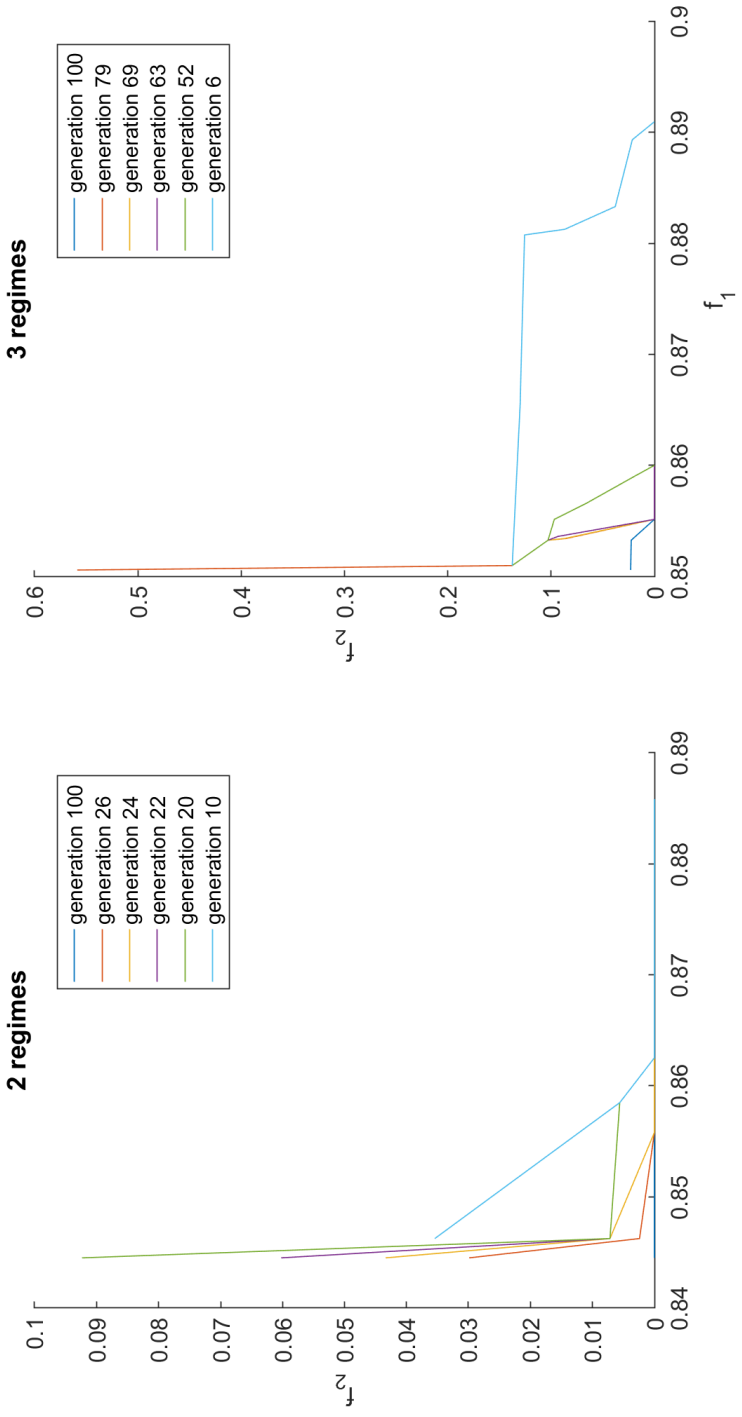


Figure 5.5: Means and variances of silhouette numbers

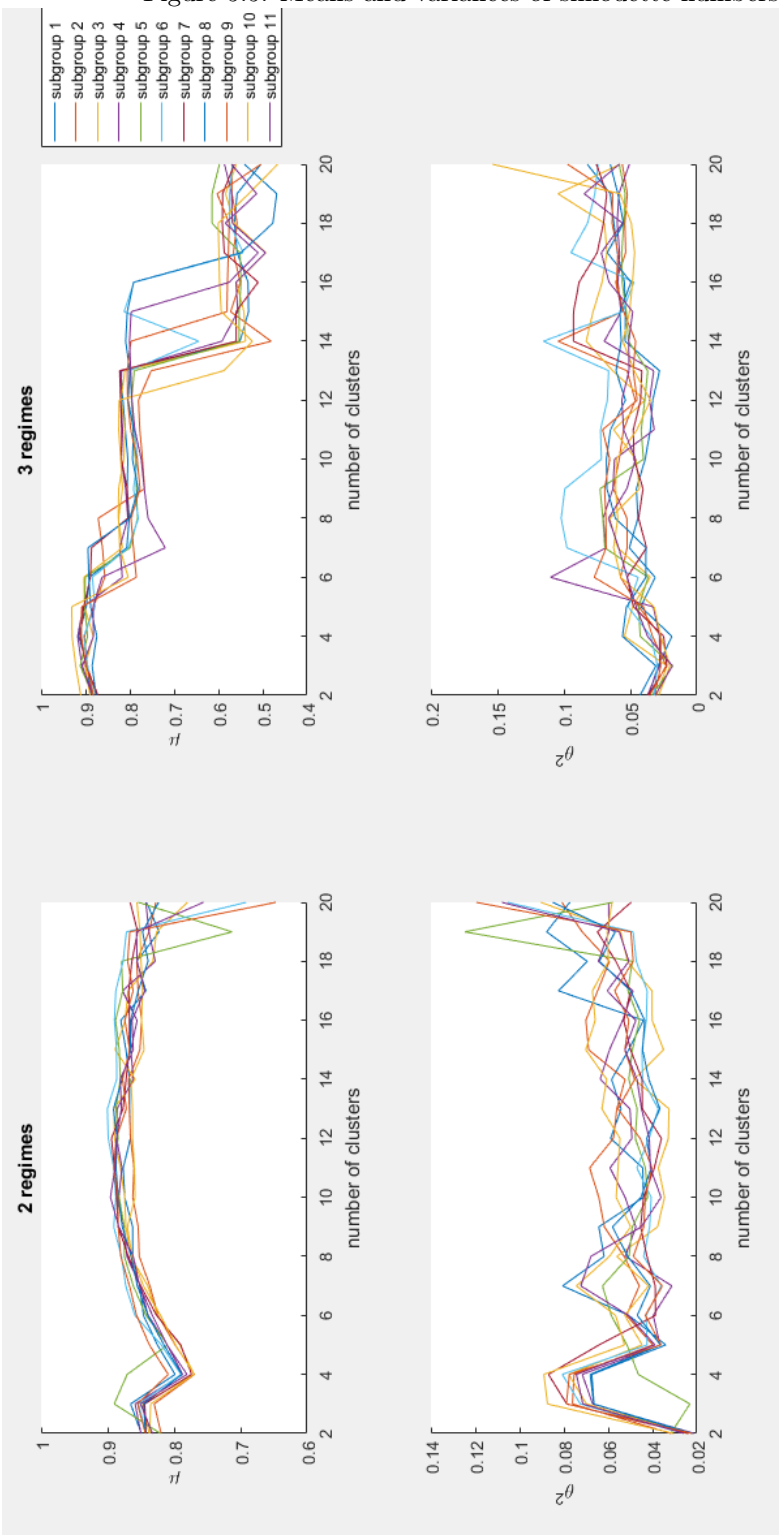


Figure 5.6: Silhouette plots for clusters of results with 3 regimes

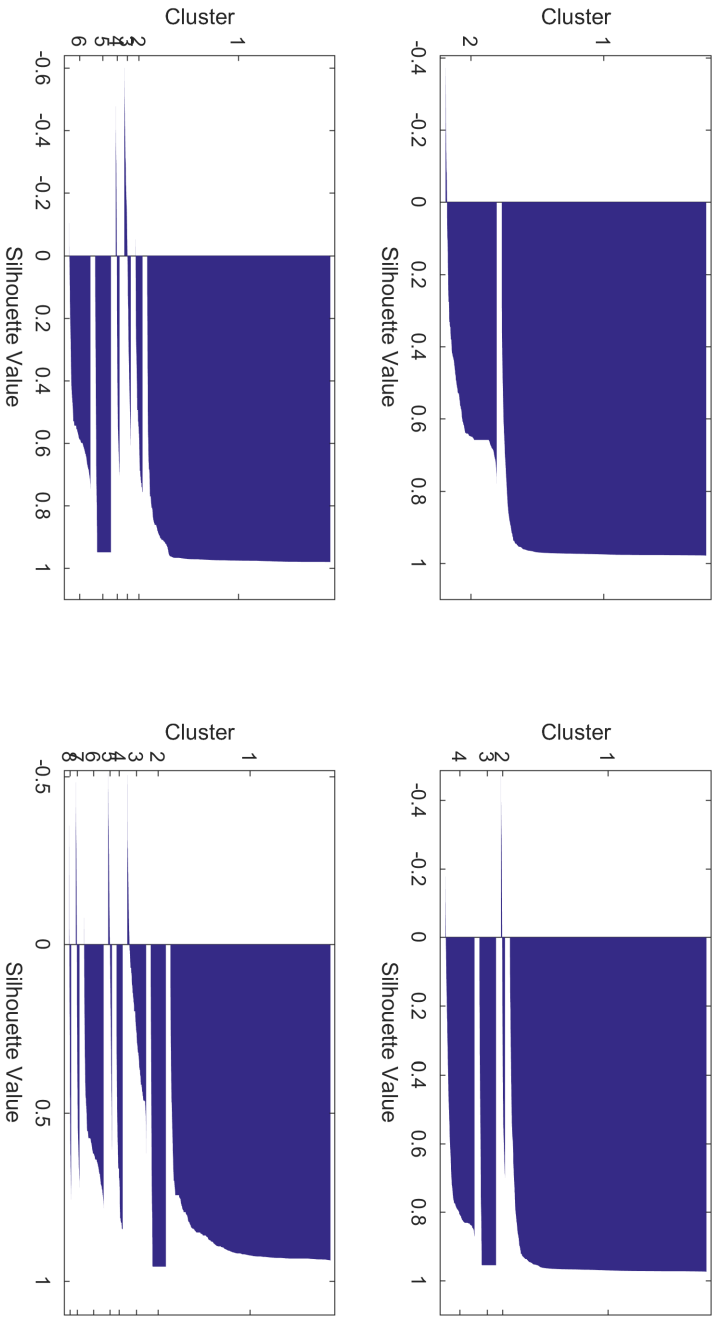


Figure 5.7: Silhouette plots for clusters of results with 2 regimes

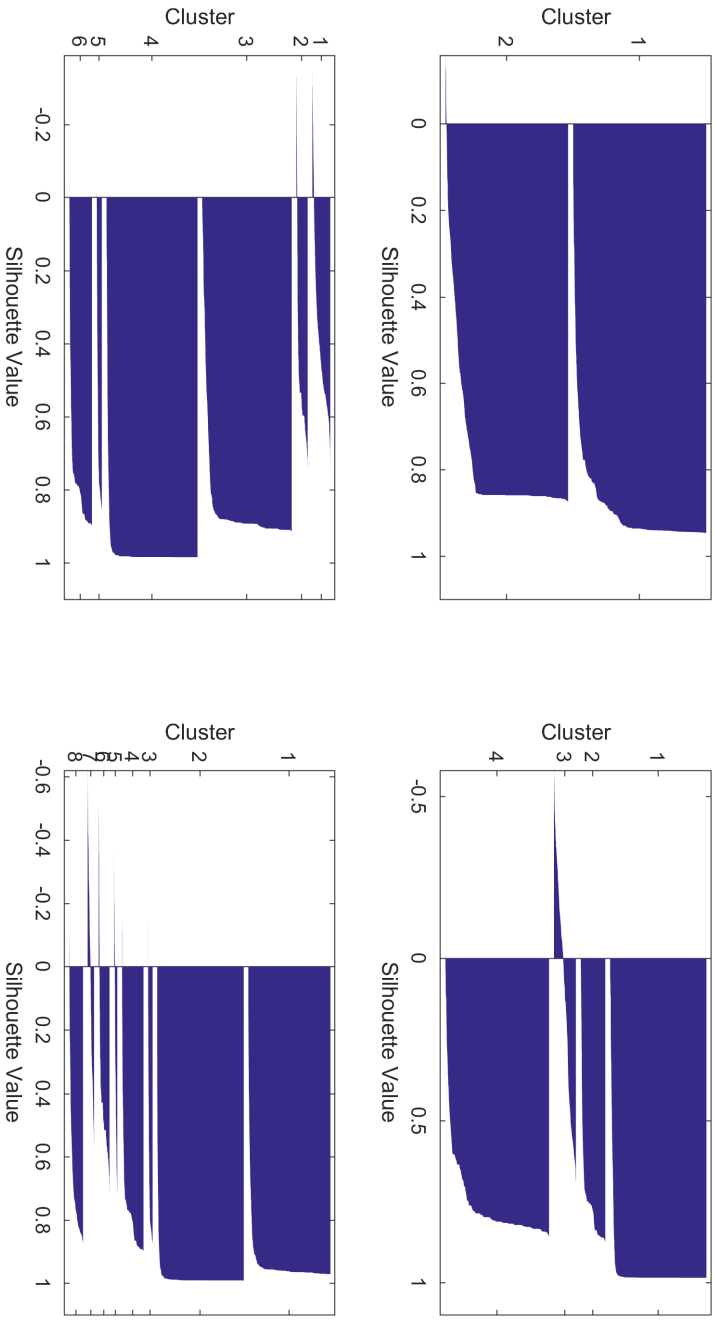


Figure 5.8: CRSP data with representative partitionings: 3 cutpoints

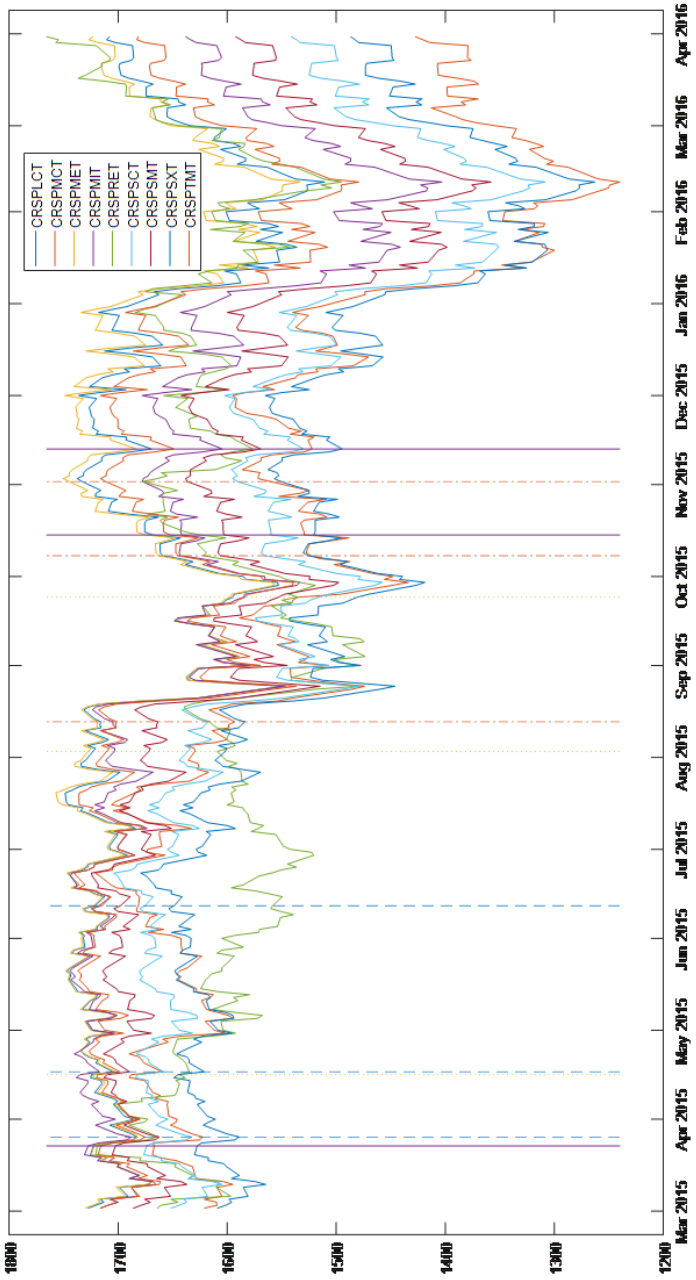
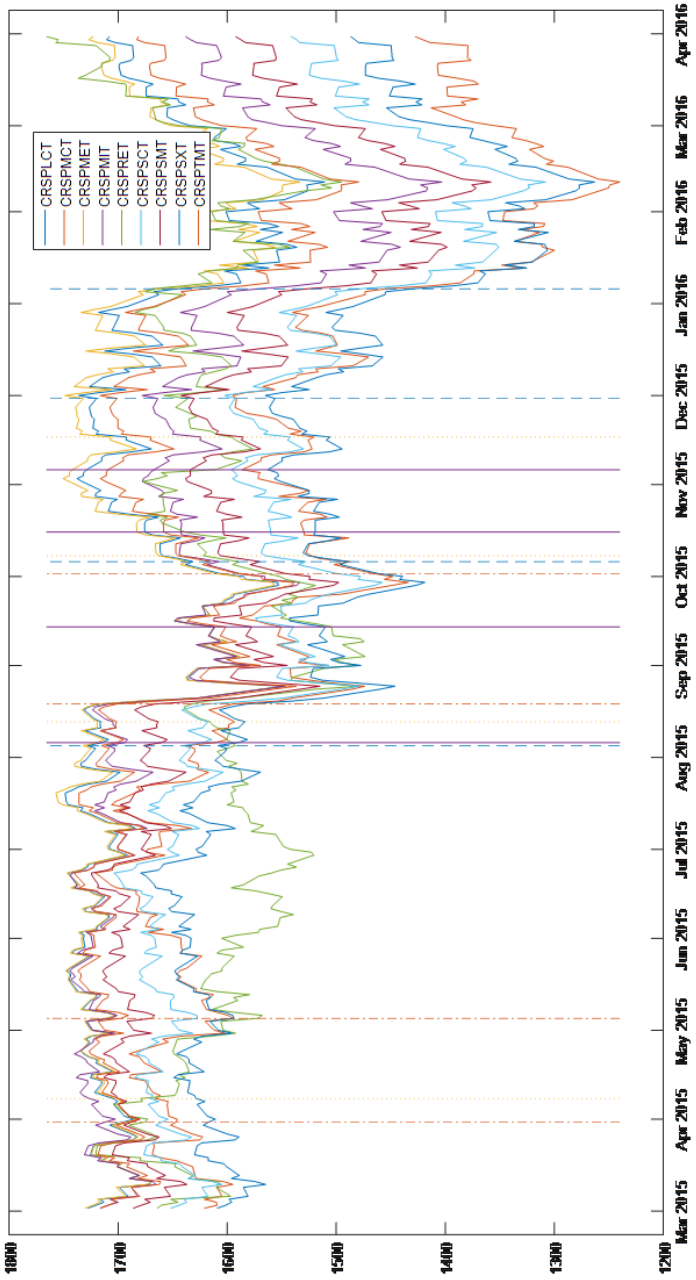


Figure 5.9: CRSP data with representative partitionings: 4 cutpoints



Chapter 6

Conclusion

The main contributions of the research detailed in this thesis are in specific areas of the theory, design, execution and testing of MOEAs. In particular, several advances in the theory of multiobjective tournament selection have been detailed; an optimization model has been developed to address a hard multiobjective problem, namely the partitioning of multivariate self-affine time series; a specialized MOEA has been designed to solve this problem with several unique design elements, including the choice of tournament selection scheme, in line with theory; and specific testing paradigms were developed, with promising results for performance of the MOEA.

In Section 6.1 below, the main contributions of and findings contained in this thesis are summarized, followed by comments on possible future directions of research in Section 6.2.

6.1 Summary of main contributions and findings

In Chapter 2, the main contribution was a set of results was developed concerning both the operation of ranking schemes under multiobjective tournament selection and the effect on both the development of the rank or fitness distribution; these findings underpin the design of the tournament selection algorithm which

forms a key part of the specialized MOEA described in subsequent chapters. Firstly, results regarding the operation of Pareto rankings were shown, based in turn on axioms derived from the operation of a simple ranking algorithm. Then it was shown that there exists a general limiting expression for the probability of nonsampling under multiobjective tournament selection without replacement using Pareto rankings, and hence for the nonselection of elite individuals. This analysis can be adapted to selection without replacement as well as to the effect of the addition of elitism, and the effects of these variations of the tournament selection algorithms were examined, with the conclusion that completely permuted schemes are of interest as they guarantee all individuals will be sampled yet remains tractable to analysis of the expected rank or fitness distribution.

It was also shown shown that there is a limiting expression for the expected number of generations to convergence for an iterative search algorithm using only tournament selection. This expression could be used as a useful check in developing or choosing algorithms. For example, if an algorithm converges close to or even faster than this limiting prediction, it could be considered likely that the algorithm is demonstrating premature convergence. It was shown furthermore that one can find a tournament size that maximizes the expected rank size for any given rank below the first rank. Finally, it was shown that there exists a limit to the nonsampling probability across multiple generations that has a negative double exponential form, implying that archiving can be a partial solution to the elite nonsampling problem.

Whilst it has long been recognized that altering tournament size will tend to increase selection pressure, it was shown that for all but the individuals in rank 1, there exists a value τ^*_{ig} for tournament size, at a given generation, beyond which selection pressure will in fact start to decrease. It was also shown that the value of τ partially determines the nonsampling limit in some selection schemes, that the effect is doubly exponential under iterated tournament selection, and that it plays a role in expected time to convergence, which implies a trade-off between algorithm efficiency and the potential for premature convergence.

Finally, it was also shown that varying τ has a predictable effect on the growth rate of the elite population. Lemma 2.9 and Theorem 2.2, Equation 2.50 and Theorem 2.4 all reveal important effects of the choice of tournament size. Hence the value of τ must be carefully chosen, balancing these various considerations.

In Chapter 3 we saw that the problem of how to best partition a multivariate time series into subseries with different distributional properties is an important one with many potential uses in different areas of research, but is also at root a difficult combinatorial problem with high computational complexity. The problem becomes more complex still if we assume self-affinity in the underlying DGP; yet many real data types, including but by no means limited to financial data, display this attribute. Identifying valid partitions on this basis may not only be the best way to identify time-varying features of the data but may also open the door to prediction of future states, or at least identification of the current state on some scale. However, statistical techniques currently available are not well-suited to high-dimensional multivariate analysis of time series showing time-varying, self-affine distributional attributes. Furthermore, all rely on assumptions about the underlying DGP and often on large numbers of model parameters.

The problem was formulated so as to minimize the similarity between successive coarse-grained subseries and maximize the similarity between this coarse-grained partitioning and some fine-grained partitioning at a smaller time scale, using functions that summarize each objective, based on realized covariance. The main contribution of this summarizing approach is that it significantly simplifies the problem, yet still yields a set of solutions for analysis a posteriori rather than a single solution based on a priori objective weightings, say, which would yield much less in terms of insights into the fitness landscape.

In Chapter 4, the main contribution is the novel approach to creating a specialized MOEA to partition multivariate self-affine time series. The MOEA, makes only the simplest of assumptions about the DGP and uses only 3 input parameters, two of which, the population size N and the maximum number of

generations G, relate to the computational structure of the MOEA rather than being model parameters as such, with the third, the minimum partition size t_{min}, in practice being set to the effective minimum value 2 in all experiments. The price paid is that the starting computational complexity is very high. To address this, a highly parallelizable population-based evolutionary algorithm was developed, which reduces the problem to a biobjective one using objective functions based on the correlation of realized covariances for successive coarse-grained subseries and for fine-grained sub-subseries with the coarse-grained subseries.

The summarizing approach developed in Chapter 3 significantly simplifies the problem, yet still yields a set of solutions for analysis a posteriori rather than a single solution based on a priori objective weightings, say, which would yield much less in terms of insights into the fitness landscape. The population is split into subgroups specialized to examining fine-grained partitions in differing sections of the time series, and all possible partition sizes can be investigated.

The algorithm uses biobjective permuted tournament selection with randomized size, a choice supported by the findings in Chapter 2. The mathematics of the tournament selection scheme chosen imply that the nonselection probability for elite individuals is zero, obviating the need for explicit elitism.

The specialized MOEA also uses a crossover method that in effect applies affine transformations where necessary to fit together elements of each parent's representation, and a mutation method developed to maximize the number of feasible individuals after operation.

In Chapter 5 the main contribution is the development of techniques to test the specialized MOEA using both simulated data and real stock market data. The simulated data was constructed using generation processes known to be self-affine and designed to have as clear a partitioning as possible in terms of the main metric used by the MOEA to assess differentiation of successive subseries. Initial results indicated that the MOEA may be able to come close to the partitioning used in the simulated data whilst simultaneously finding

reasonable self-affinity, and indeed in limited testing was able to find partitions with better differentiation than the ones used to set up the test data. It was also observed that there is a clear trade-off between closeness to the original partition and the measured power of the differentiation between successive coarse-grained subseries, but that the overall solution set improved with successive generations. For the testing using real data, although it is not possible to comment directly on the accuracy of results as the "true" partitions are unknown, several useful observations regarding the operation of the MOEA on the specific data set were made. Although parallelism was implemented in the MOEA described in Chapter 4, in testing, it was rather the high-level testing algorithm that implemented parallel processing.

6.2 Direction of future research

The work presented in this in this thesis has addressed certain key questions regarding the theory, design and execution of MOEAs, but has also raised a number new questions, or highlighted existing ones that remain to be adequately addressed. In the course of the work on the mathematics of tournament selection, it was found that far from being a topic that is largely understood in all its key aspects, this area still has much worth researching. Throughout the work, there are specific assumptions about the selection schemes used, as well as the precise method for determining rankings; hence all findings are dependent on these assumptions and particular methods. This work is in many ways still its early stages, and many questions remain. The effect of elitism of different types in different ranking and selection schemes is one area for future research. The effect of different tiebreakers in Pareto ranking schemes on subsequent fitness distributions is another area of interest. Furthermore, researchers may wish to investigate the effect of genetic operators in conjunction with tournament selection of the fitness landscape. This is perhaps the hardest area to address, since tournament selection operates on the fitness space, whilst genetic operat-

ors have their effect in the representation space, and the mapping between the representation and fitness spaces is generally unknown; but progress in this area could lead to a much deeper understanding of the mathematics of evolutionary algorithms in general.

To progress further with the work on partitioning multivariate self-affine time series , it is anticipated that it will be necessary to develop new techniques and extensions to the MOEA framework to test whether the partitionings produced are useful for example in minimizing variance of a portfolio of assets in out-of-sample testing when the current state of the asset market is unknown. It will then be possible to address the feasibility of probabilistically assessing the current state. It would also be useful to develop suitable metrics for quality of the PF, with attention to the particular nature of the problem. Although the motivation for this research comes from financial applications, it can be seen more widely as the first steps on the road to a broader framework for threat analysis, detection and mitigation, and it would be useful to extend testing to other types of data. For example, the work presented in this thesis could be used to detect abnormal patterns temporarily present at different time scales in multivariate data of many types. Future work might allow optimization to mitigate the adverse impact of such patterns, and even detect the onset of such patterns in real time, allowing re-optimization.

Papers and conferences

Taylor, C. M. (2014). *A framework for production of robust portfolios using a portfolio multiobjective optimization evolutionary algorithm and a K-means genetic algorithm*. University of Essex.

Taylor, C. M. (2018). *Use of GPU computing in MOEAs*.

Taylor, C. M., & Salhi, A. (2017). On Partitioning Multivariate Self-Affine Time Series. *IEEE Transactions on Evolutionary Computation*, *21*(6), 845–862. https://doi.org/10.1109/TEVC.2017.2688521

Taylor, C. M., & Salhi, A. (2018a). The Mathematics of Pareto Rankings and Tournament Selection. In *EGL'2018 Workshop on Optimisation, Applied and Numerical Mathematics*.

Taylor, C. M., & Salhi, A. (2018b). The Mathematics of Pareto Rankings and Tournament Selection. *Submitted*.

Appendix A

Appendix - programming and parallel processing

This appendix contains additional information regarding the choice and use of Matlab as a programming environment, and on the use of GPU computing in the development of the MOEA, although for reasons discussed below, GPU computing was not used in the final version as tested; only CPU parallel computation was deployed.

Much of the material Matlab in this appendix has been gleaned from the programme's own extensive online documentation and online forums, both Mathworks' own and others. For Subsection A.3, [92] is useful but technical and now quite outdated. For Subsection A.3.3, [136] is a relatively recent study, whilst [153] is less recent but still useful. However, the most important source material is extensive experience of iterative bug fixing and performance tweaking in the course of developing the specialist MOEA, for which there is no substitute.

A.1 Hardware

As noted in Chapter 5, the principal hardware platform used in his project was a 3.7GHz, 4 core system with 32GB of RAM upgraded during testing to

64GB to overcome problems with oversized arrays during testing, as well as to improve performance. In addition, two high performance GPUs were added at different times; an Nvidia GTX Titan Black with 6GB memory and 2880 CUDA cores, and an Nvidia GTX 1080 Ti with 11GB memory and 3584 CUDA cores. Whilst these are both consumer graphics cards, they are both enabled for Nvidia's CUDA parallel processing platform and have high performance as GPU processors, used with Matlab's Parallel Computing Toolbox. The availability of two GPU cards and more than two CPU cores makes it possible to operate the GPUs in parallel, as well as distributing tasks over the cores of the GPU, though as explained in Section A.3. However, the system only supports 16 lanes, meaning that only 8 lanes are available for each GPU when both are used; a more powerful CPU/motherboard combination could provide 16 lanes to each GPU, increasing computational throughput.

A.2 Choice of programming language

Which programming language and environment to use is a critical choice, typically made very early on in a project and difficult to change once started. Choice of language can affect, inter alia:

1. Ease of programming, including interface, debugging, reuse and making changes;

2. Speed of execution, analysis of computational efficiency and making improvements to execution time;

3. Availability of ready-made code, whether third party or proprietary,and ability to audit and override code;

4. Implementation of specific computational features pertinent to the specific task a researcher is attempting;

5. Extent and ease of use of parallel processing;

6. Quality and ease of use of graphics, in particular visualization tools;

7. Stability, reliability and upgradabilty.

The specialized MOEA described in Chapter 4, as well as many investigations done behind the scenes to investigate and verify theoretical results, was developed entirely in Matlab Versions 2014b - 2018a, as published by Mathworks Inc. In addition, Maple Versions 18 and 2016-2018 was used for algebraic investigation and generation of certain graphics. Maple has the advantage that one can work directly with algebraic formulae, and generate results and graphics therefrom, making it useful for investigation, verification and visualization, particularly in Chapter 2. Although Maple does have programming capabilities, they are rudimentary compare to Matlab, so Maple was only used in conjunction with the MOEA with respect to investigations of more theoretical aspects.

With regard to points 1-7 above, below is a brief summary of how, with the benefit of extensive use on this project, Matlab fares as a development environment for a specialized MOEA:

1. Matlab is a high level programming language with a helpful programming interface, extensive documentation, and reasonable debugging tools. It is, with certain exceptions noted later, relatively tolerant and informal, in the sense that many different syntaxes can be used to achieve a desired result, and indeed object oriented code can be mixed with standard or legacy code fairly freely. Changes can be made on he fly without the need to recompile code, as Matlab uses a just-in-time compiler, and it is easy to build up a library of specialized function which can freely call each other in a nested manner. Memory allocation and garbage collection are automatic, and in general there is little need to worry about the interface with hardware (other than keeping within memory limitations); the only exception being parallel processing, as discussed in Section A.3.

2. Perhaps the greatest argument against using Matlab historically has been computation speed, in particular compared to lower level languages such

as C. However, since the new Matlab execution engine was implemented in Release 2015b, code has run noticeably faster - informally, some key routines used in the MOEA have been observed to run several times faster - and in any case, many low-level functions in Matlab are reputedly compiled in C/C++ anyway (and many other languages - the interface is reportedly written in Java, CUDA is implemented for GPU computing, and use of other languages has been reported for specialized use[1]. Many routes exist to improving execution time in Matlab, including, but not limited to, exploitation of its strength in manipulation of large matrices; vectorization; use of Matlab executable (MEX) files to call routines written in C/C++; parallel processing; and use of object-oriented programming. All of these topics are discussed in varying degrees of detail in following sections. Matlab also features Profiler, a tool that highlights the most computationally intensive parts of a user's code, at any level from highest (for example, in the context of the specialized MOEA, that calculation of fitness functions is consuming a large portion of CPU time) to the lowest, i.e. specific subroutines.

3. Matlab uses a wide variety of toolboxes to provide specialized functions to the user, which have the advantage of having been extensively tested and optimized by Mathworks Inc. yet are generally written themselves in Matlab, meaning they can be examined by the user and if necessary, overridden in various ways to meet specific needs. A less desirable side-effect is that useful functions are often contained in toolboxes that a researcher in a given field might not think of looking at, or have access to (given that each toolbox must be bought separately); for example, graph theory tools are to be found in the Bioinformatics Toolbox. Beyond the toolboxes, many users share stand-alone functions for specific purposes or develop entire toolboxes that are shared online. However, such third party software is certainly not guaranteed to be bug free, accurate (in performing as

[1] http://www.walkingrandomly.com/?p=4333

specified) or efficient. The MOEA avoided using such third party software, but extensive use was made in testing of a third party tool to generate the FITS, as described in Section 5.3.

4. Matlab started life as a matrix manipulation library, and remains particularly good at this, in particular fast execution of many matrix manipulation and calculation functions, including very large and sparse matrices. Given that, as can be seen in the explanations in Chapter 3, the MOEA relies heavily on matrix functions, Matlab is ideally suited to development and execution of some of the core algorithms within the specialized MOEA, especially on large datasets with high dimensionality.

5. Matlab implements parallel processing across multiple CPUs and GPUs using the Parallel Computing Toolbox and for larger implementations, the Distributed Computing Server. Parallel computing in Matlab is discussed in more detail in Section A.3.

6. Whilst lacking the range of visualization tools available from specialist programmes such as Tableau, Matlab nonetheless has considerable flexibility in graphics production tools, as can be seen from the variety of visualizations reproduced in Chapter 5; the graphics in Chapter 2 were generated using both Matlab and Maple. Most of these graphics were produced not by using one-off commands, but by writing customized, reusable functions, often themselves incorporating specialized graphics functions contained in Matlab toolboxes; for example, the silhouette visualizations in Section 5.4 use functions contained in the Statistics and Machine Learning toolbox.

7. As a commercial product developed, refined, extended and updated over decades, Matlab is very stable, and few crashes were experienced during development and testing of the specialized MOEA or in other preparatory work, though mistakes in programming can cause non-terminating loops which occasionally cannot be stopped other than by exiting the programme. Runtime errors are handled by terminating the user programme

within Matlab and returning control of the interface, usually with a (more or less) useful error message. Matlab is upgraded twice a year, generally with a high degree of backwards compatibility.

Hence, whilst no doubt other programming languages could have been used successfully, Matlab fulfilled the requirements for the development and implementation of the specialized MOEA well. For a high-usage or commercial application, after further development it might be preferable to convert the programme to another language, but note that Matlab also has a range of add-on Coder products that can automate a good deal of this process.

A.3 Use of parallel processing in Matlab

Matlab's Parallel Computing Toolbox provides tools for the use of multiple CPUs and/or multi-core CPUs to run programmes in parallel, but also tools to use a GPU with many cores - modern GPUs have thousands - for parallel processing, and more recently, has introduced specific features to handle the use of multiple GPUs, themselves running in parallel, as well as support for distributed clusters of CPUs and for cloud computing platforms including Amazon Web Services and others. What follows below is a description of the deployment of a local system with multiple CPU cores and multiple GPUs, as this was the type of system used in the development and testing of the specialized MOEA .

A.3.1 Implicit parallel processing

Matlab uses parallel CPU processing "under the hood" for many low level functions, in particular operations on arrays. Continuous improvements to the way Matlab uses multiple CPU cores without explicit instruction from the user have increased performance, but also mean that in many circumstances (but by no means all), programmes run as fast or faster without explicitly coding parallel CPU loops. It is difficult or impossible to predict what will work best; often, he only way is to try both and measure performance. On the other hand, GPUs if

present are never used by Matlab unless specifically instructed.

The programmer can also often significantly increase execution speed by use of vectorization, which has been a key feature of Matlab since its early days. Vectorization is a large topic, a proper treatment of which is beyond the scope of this chapter, but the basic idea is that by putting variables into arrays and operating on the whole array, rather than single variables sequentially, one can speed up execution substantially, in part by Matlab's use of multiple CPU cores in parallel. The basic way to do this is by using vector valued functions, but there are more advanced methods, using specific Matlab constructs such as *arrayfun*, together with anonymous functions (very simple compact functions that are not stored separately in a function file). Vectorization is always advised in the documentation as the first step in speeding up code; however, the sophisticated optimization in recent versions of the execution engine has reduced the relative advantage of this technique, the exception being in GPU processing, as we shall discuss in Subsection A.3.3.

A.3.2 The use of the *parfor* syntax for CPU parallel processing

The implementation of CPU parallel processing in Matlab is superficially very simple. A pool of "workers" (objects presenting CPU cores) is initialized using the *parpool* command, or by certain other means, and parallel processing is invoked by substituting the *parfor* command for other types of loop commands (*for* or *while*). Matlab automatically divides processing of iterations of the loop between workers, in theory cutting loop execution time substantially. There are two important caveats however. Firstly, one may use *parfor* at only one level of nesting, whether within a particular function or where functions are nested. Whilst Matlab will allow nested *parfor* commands, tasks can only be divided amongst multiple workers at one level, so nesting will not improve performance and may detract from performance as Matlab wastes time dealing with redundant allocations. Secondly, in general dependency between different iterations is

not allowed Matlab deals with code within a loop in very particular ways that require much tighter programming following particular rules which do not apply to regular execution.

In Matlab programmes generally, one does not have to explicitly declare a type for a variable, and in normal (i.e. non-parallel) Matlab programming one has to spend little time worrying about what role a variable will pay in programme execution, as opposed to its type (structure, matrix etc.) and one does not always even have to explicitly declare a type. In *parfor* loops, however, there are several different roles played by variables that are handled differently, namely:

- Loop variables which are used to index the loop execution;
- Temporary variables which are created and used solely inside the loop;
- Reduction variables whose values change across loops but without reference to loop execution order;
- Broadcast variables, which are defined before and whose values are used within the loop but not assigned or changed within the loop;
- Sliced variables, which are arrays or other similar constructs where different parts of the variable are operated on by different iterations within he loop.

It is typically the last two types of variable that cause problems. Broadcast variables can impose a heavy communication burden which can slow down execution, possibly to below that of a normal loop. However, such code will typically run. The trickiest to deal with are sliced variables, which can prevent code from running for many reasons, often difficult to analyse even given error messages from Matlab. The most common issue is that Matlab cannot work out whether there are dependencies between different parts of the same variable, and therefore cannot assign different workers to different iterations.

However, there are many other wrinkles. For example direct assignment to arrays indexed by the *parfor* loop within a nested *for* loop is not generally possible, and the same can apply to other types of multilevel assignment, for example assignment to objects using dot notation. Frequently a work-around can be found by first assigning to a temporary variable and then to a sliced variable, or else by writing a new function and calling that function within the *parfor* loop, rather than the underlying code, but these measures still have to avoid cross-iteration dependencies to be allowed within a *parfor* loop.

A.3.3 GPU processing

The use of GPUs to speed up execution is a relatively recent addition to Matlab, and effective (or at least easy) use of multiple GPUs is more recent still. GPUs have a fundamentally different architecture which means that they can contain many more cores in a single chip than a CPU; however, the limited instruction set means typically that they can only execute very simple operations, or combinations thereof, and typically do not handle logic operations, in particular branching, well, if at all. When well implemented in very suitable applications however, for example the calculation of realized covariance on large arrays as described in Chapter 4, the speed improvements can be very large, depending on the relative power of GPUs v CPUs in the system. Whilst in this project speed increases of the order of many tens or even hundreds of times which are sometimes reported from testing were rarely is ever seen, improvements of several times or even more than 10x were observed in some cases. However, the degree of speed improvement depends critically on the problem addressed and the techniques employed; in many cases, use of the GPU can slow execution, possibly substantially. This is because any data and variables to be used, including empty arrays, must first be moved from computer general-use memory to the dedicated GPU memory, and if results are required for execution of code by the CPU, then the necessary data must be moved into general memory. These processes can create a communication overhead that can more than offset any

speed gains from the portions of code executed on the GPU.

The main methods for moving data from main memory to the GPU and back are the function *gpuArray* and its mirror function *gather*. However, where possible it can be more efficient to initiate data on the GPU. A substantial subset of Matlab built-in functions, including those in toolboxes, have an overridden version that works with the GPU, an option to work with the GPU, or in some cases, will work with no change to coding with data on the GPU and will return data also on the GPU. In particular, it is possible to initiate arrays - blank, filled with ones or zeros, random numbers or defined sequences - on the GPU with no transfer from main memory, saving substantially on computation time. Best practice is as far as possible to initiate variables this way, and to transfer any other data from main memory just once at the beginning of a programme or subroutine and once back when processing is complete. This requires programming everything on the GPU, as far as possible.

The issue is that many commonplace Matlab functions either do not work at all on the GPU, or have some restrictions on how they are used or available options. This means it is often necessary to programme in a different way, or to find ways of producing the same results s a given function but using simpler functions that will work on the GPU. An advantage however is that one does not generally face the kinds of difficult-to-trace errors that *parfor* throws - the issue is rather finding a way to produce a desired result with more limited tools. Finally, another option is to run MEX files containing CUDA code, but this requires additional knowledge of CUDA and how the language controls the GPU.

Loops and branching are difficult to perform on the GPU as common constructs like *if...then* statements and *for* loops cannot be directly executed on the GPU. Loops however execute reasonably well in many cases with the loop itself run on the CPU and all the rest of the code on the GPUs as long as data is not transferred between GPU and main memory during the loop. However, *if...then* statements present more of a challenge as data transfer is usually un-

avoidable. One workaround for both loops and conditional statements is the use of the GPU versions of *arrayfun*, *bsxfun* and *pagefun*, which can replicate the effect of loops and can also use conditional commands within the body of user-defined anonymous functions, though careful programming is required, and actual results in terms of speed are not always as expected.

In recent versions of Matlab, use of multiple GPUs has become quite simple, as Matlab will now automatically assign workers within a pool created by *parpool* to GPUs in an optimal way, though it is also possible to assign them manually. Efficient use of multiple GPUs with automated worker allocation requires use of *parfor* loops or of other functions that explicitly support multiple GPUs. The only additional complication is that one must specify which GPU data resides on and keep track of this. Moving data between GPUs is best avoided for similar reasons to those for avoiding moving data between GPUs and main memory.

Another complication in using GPUs is that some degree of explicit memory management is often required. In general with CPU computing in Matlab, one has to do little more than ensure one does not throw out of memory errors. In GPU computing, firstly the dedicated memory available to the GPU is often rather smaller than main memory, meaning care must be taken when transferring large amounts of data. Secondly, if using multiple GPUs, one must keep track of memory usage on each GPU. Finally, an irksome feature of GPU computing is that the GPU memory tends to fragment over time leading to slower execution speed, less available memory and potentially, out of memory errors even when memory used appears to be within limits. The only solution to this appears to be to periodically take all the data off the GPU, reset it and move the data back, which can add substantially to execution time.

GPU-enabled versions of several key functions, including fitness evaluation and tournament selection, were developed and tested successfully in single runs of the MOEA, but not used in the version of the MOEA with multiple runs in parallel on the CPU used for testing in Chapter 5, despite working well on single runs with significant speed increases over CPU-only versions. The main

reason was that the memory fragmentation issue over many runs in testing made the MOEA unstable and prone to crash when available memory on the GPU reduced to below requirement. Further development would be necessary to solve this issue, unless future updates to Matlab take care of fragmentation automatically.

.

Bibliography

[1] Patrice Abry and Gustavo Didier. Wavelet estimation of operator fractional Brownian motions. *Bernoulli*, 24(2):895–928, 2018. URL http://www.e-publications.org/ims/submission/BEJ/user/submissionFile/21616?confirm=cffb2b8c.

[2] I. Alberto, C. Azcarate, F. Mallor, and P.M. Mateo. Multiobjective Evolutionary Algorithms . Pareto Rankings. *Monogr. Semin. Mat. Garcia Galdeano*, 27:23–25, 2003. URL https://documat.unirioja.es/servlet/articulo?codigo=867106.

[3] L Altenberg. The schema theorem and Price's theorem. In *Foundations of Genetic Algorithms Vol. 3*, pages 23–49. Elsevier, 1995. ISBN 1558603565. doi: 10.1007/BF02080569.

[4] Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, 2017. ISSN 0219-1377. doi: 10.1007/s10115-016-0987-z. URL http://link.springer.com/10.1007/s10115-016-0987-z.

[5] Torben G Andersen, Tim Bollerslev, Francis X. Diebold, and Paul Labys. Realized Volatility and Correlation. Working paper. 1999. URL https://www.sas.upenn.edu/{~}fdiebold/papers/paper29/temp.pdf.

[6] Torben G Andersen, Tim Bollerslev, Francis X. Diebold, and Paul Labys. The Distribution of Realized Exchange Rate Volatility. *Journal of the*

American Statistical Association, 96(453):42–55, 2001. ISSN 0162-1459. doi: 10.1198/016214501750332965. URL http://www.nber.org/papers/w6961.

[7] Alexander Aue, Siegfried Hörmann, Lajos Horváth, and Matthew Reimherr. Break detection in the covariance structure of multivariate time series models. *Annals of Statistics*, 37(6 B):4046–4087, 2009. ISSN 00905364. doi: 10.1214/09-AOS707.

[8] AD Back and AS Weigend. A first application of independent component analysis to extracting structure from stock returns. *International Journal of Neural Systems*, 8(4):473–484, 1997. URL http://www.worldscientific.com/doi/abs/10.1142/S0129065797000458.

[9] Thomas Bäck. Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms. In *IEEE World Congress on Computational Intelligence (1994)*, pages 57–62. IEEE, 1994. ISBN 0-7803-1899-4. doi: 10.1109/ICEC.1994.350042. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=350042.

[10] Ole E. Barndorff-Nielsen and Neil Shephard. Estimating quadratic variation using realized variance. *Journal of Applied Econometrics*, 17(5): 457–477, 2002. ISSN 08837252. doi: 10.1002/jae.691.

[11] Ole E. Barndorff-Nielsen and Neil Shephard. Econometric Analysis of Realised Covariation: High Frequency Covariance, Regression and Correlation in Financial Economics. *Econometrica*, 72(3):885–925, 2004. ISSN 1556-5068. doi: 10.2139/ssrn.305583.

[12] Jozef Barunik, Tomaso Aste, T. Di Matteo, and Ruipeng Liu. Understanding the source of multifractality in financial markets. *Physica A: Statistical Mechanics and its Applications*, 391(17):4234–4251, sep 2012. ISSN 03784371. doi: 10.1016/j.physa.2012.03.037. URL http://linkinghub.elsevier.com/retrieve/pii/S0378437112002890.

[13] Barbara Bedowska-Sojka and Agata Kliber. Realized Volatility versus GARCH and Stochastic Volatility Models. The Evidence From The WIG20 Index and the EUR/PLN Foreign Exchange Market. *Przeglad Statystyczny*, 2010. URL http://keii.ue.wroc.pl/przeglad/Rok2010/Zeszyt4/2010{_}57{_}4{_}105-127.pdf.

[14] A. Belegundu, E. Constans, R. Salagame, and D. Murthy. Multi-objective optimization of laminated ceramic composites using genetic algorithms. In *5th Symposium on Multidisciplinary Analysis and Optimization*, Reston, Virigina, 1994. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1994-4363. URL http://arc.aiaa.org/doi/10.2514/6.1994-4363.

[15] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies A comprehensive introduction. *Natural Computing*, 1(1):3 – 52, 2002. ISSN 1572-9796. doi: 10.1023/A:1015059928466. URL http://www.cs.bham.ac.uk/{~}pxt/NIL/es.pdf.

[16] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009. ISSN 15677818. doi: 10.1007/s11047-008-9098-4.

[17] Fischer Black. Studies in Stock Price Volatility Changes. In *Proceedings of the 1976 Business and Economic Studies Section*, pages 177–181. American Statistical Association, Alexandria,Va., 1976.

[18] Tobias Blickle. *Theory of Evolutionary Algorithms and Application to System Synthesis*. PhD thesis, Swiss Federal Institute of Technology, Zürich, 1996. URL https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/142835/eth-40490-01.pdf.

[19] Tobias Blickle and Lothar Thiele. A Mathematical Analysis of Tournament Selection. In *Genetic Algorithms: Proceedings of the 6th Interna-*

tional Conference (ICGA95), pages 9–16, San Francisco, 1995. Morgan Kaufmann. doi: 10.1.1.52.9907. URL https://pdfs.semanticscholar.org/e1cc/08dd2f2d622dd7c54e76aacd853101fe3451.pdf.

[20] Tobias Blickle and Lothar Thiele. A Comparison of Selection Schemes used in Genetic Algorithms. *Evolutionary Computation*, 4(4):361–394, 1996. ISSN 1063-6560. doi: 10.1162/evco.1996.4.4.361.

[21] Tim Bollerslev. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986. ISSN 03044076. doi: 10.1016/0304-4076(86)90063-1.

[22] George E. P. Box. Evolutionary Operation: A Method for Increasing Industrial Productivity. *Applied Statistics*, 6(2):81, 1957. ISSN 00359254. doi: 10.2307/2985505. URL http://www.jstor.org/stable/10.2307/2985505?origin=crossref.

[23] Jürgen Branke, Hartmut Schmeck, and Hans Christian Andersen. Global Selection Methods for SIMD Computers. In *Workshop on Evolutionary Computing*, pages 6–17. Morgan Kaufmann, 1996. URL https://pdfs.semanticscholar.org/f48a/2b6f474866493f44196c684d4ca7cce4a326.pdf.

[24] Hans J Bremermann. Optimization through Evolution and Recombination. In *Self-organizing Systems - 1962*. Spartan Books, Washington DC, 1962. ISBN 978-1258715038.

[25] Hans J Bremermann, M. Rogson, and S. Salaff. Search by evolution. In *Biophysics and Cybernetic Systems - Proc. 2nd Cybernetic Sciences Symp.*, pages 157–67, Washington DC, 1965. Spartan.

[26] Anne Brindle. *Genetic Algorithms for Function Optimization*. Phd thesis, University of Alberta, 1980. URL http://www.citeulike.org/group/712/article/431776.

[27] Jeff Calder, S Esedoglu, and A O Hero. A continuum limit for non-dominated sorting. In *Proceedings of the Information Theory and Applications Workshop (ITA), 2014*, 2014. doi: 10.1109/ITA.2014.6804207.

[28] Laurent E. Calvet and Adlai J. Fisher. Forecasting Multifractal Volatility. *Journal of Econometrics*, 105(1):27–58, 2001. ISSN 03044076. doi: 10.1016/S0304-4076(01)00069-0.

[29] Laurent E. Calvet and Adlai J. Fisher. How to Forecast Long-Run Volatility: Regime Switching and the Estimation of Multifractal Processes. *Journal of Financial Econometrics*, 2(1):49–83, 2004. ISSN 1479-8409. doi: 10.1093/jjfinec/nbh003. URL `http://jfec.oupjournals.org/cgi/doi/10.1093/jjfinec/nbh003`.

[30] Laurent E. Calvet and Adlai J. Fisher. *Multifractal Volatility: Theory, Forecasting, and Pricing*. Academic Press, 2008. ISBN 0080559964. URL `http://books.google.com/books?id=ARuiiaog0IgC{&}pgis=1`.

[31] Laurent E. Calvet, Adlai J. Fisher, and Samuel B. Thompson. Volatility comovement: a multifrequency approach. *Journal of Econometrics*, 131 (1-2):179–215, 2006. ISSN 03044076. doi: 10.1016/j.jeconom.2005.01.008.

[32] Domingos Moreira Cardoso and Jorge Freire De Sousa. A multi-attribute ranking solutions confirmation procedure. *Annals of Operations Research*, 138(1):127–141, 2005. ISSN 02545330. doi: 10.1007/s10479-005-2449-y. URL `http://link.springer.com/10.1007/s10479-005-2449-y`.

[33] Gagari Chakrabarti and Chitrakalpa Sen. *Anatomy of Global Stock Market Crashes: An Empirical Analysis*. Springer India, India, 2012. ISBN 978-81-322-0462-6. doi: 10.1007/978-81-322-0463-3. URL `http://link.springer.com/10.1007/978-81-322-0463-3`.

[34] M. Chakraborty and U. K. Chakraborty. An analysis of linear ranking and binary tournament selection in genetic algorithms. In *Inter-*

national Conference on Information, Communications and Signal Processing, volume 1, pages 407–411, 1997. ISBN 0-7803-3676-3. doi: 10.1109/ICICS.1997.647128.

[35] Fei Chen, Francis X. Diebold, and Frank Schorfheide. A Markov-switching multifractal inter-trade duration model, with application to US equities. *Journal of Econometrics*, 177(2):320–342, 2013. ISSN 03044076. doi: 10.1016/j.jeconom.2013.04.016. URL `http://dx.doi.org/10.1016/j.jeconom.2013.04.016`.

[36] Luca Citi, Giulia Guffanti, and Luca Mainardi. Rank-based Multi-Scale Entropy Analysis of Heart Rate Variability. In *Computing in Cardiology*, pages 597–600, Boston MA, 2014. URL `http://cinc.org/archives/2014/pdf/0597.pdf`.

[37] Carlos Artemio Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, New York, USA, second ed. edition, 2007. ISBN 9780387310299. URL `http://books.google.com/books?id=rXIuAMw3lGAC{&}pgis=1`.

[38] Adriana S. Cordis and Chris Kirby. Discrete stochastic autoregressive volatility. *Journal of Banking and Finance*, 43(1):160–178, 2014. ISSN 03784266. doi: 10.1016/j.jbankfin.2014.03.020.

[39] Madalena Costa, Ary Goldberger, and C.-K. Peng. Multiscale Entropy Analysis of Complex Physiologic Time Series. *Physical Review Letters*, 89(6):068102–1 – 068102–4, 2002. ISSN 0031-9007. doi: 10.1103/PhysRevLett.89.068102. URL `http://link.aps.org/doi/10.1103/PhysRevLett.89.068102`.

[40] M.C. Cowgill, R.J. Harvey, and L.T. Watson. A Genetic Algorithm Approach to Cluster Analysis. *Computers & Mathematics with Applications*, 37(7):99–108, 1999. ISSN 08981221. doi: 10.1016/S0898-1221(99)

00090-5. URL http://www.sciencedirect.com/science/article/pii/S0898122199000905.

[41] Anthony Davis, Alexander Marshak, Robert Cahalan, and Warren Wiscombe. The Landsat Scale Break in Stratocumulus as a Three-Dimensional Radiative Transfer Effect: Implications for Cloud Remote Sensing. *Journal of the Atmospheric Sciences*, 54(2):241–260, 1997. ISSN 0022-4928. doi: 10.1175/1520-0469(1997)054<0241:TLSBIS>2.0.CO;2.

[42] Richard A. Davis, Thomas C.M. Lee, and Gabriel A. Rodriguez-Yam. Structural Break Estimation for Nonstationary Time Series Models. *Journal of the American Statistical Association*, 101(473):223–239, 2006. ISSN 01621459. doi: 10.1198/016214505000000745.

[43] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, New York, 2001. ISBN 9814126853. doi: 10.1109/TEVC.2002.804322. URL http://dl.acm.org/citation.cfm?id=559152.

[44] Kalyanmoy Deb and Amrit Pratap. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=996017.

[45] Gustavo Didier and Vladas Pipiras. Integral representations and properties of operator fractional Brownian motions. *Bernoulli*, 17(1):1–33, 2011. ISSN 1350-7265. doi: 10.3150/10-BEJ259. URL http://projecteuclid.org/euclid.bj/1297173831.

[46] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1 (1):19–31, 2011. ISSN 22106502. doi: 10.1016/j.swevo.2011.02.001. URL http://dx.doi.org/10.1016/j.swevo.2011.02.001.

[47] A. E. Eiben, R Hintering, and Z Michalevicz. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[48] A. E. Eiben, Zbigniew Michalewicz, Marc Schoenauer, and James E. Smith. Parameter Control in Evolutionary Algorithms. In Fernando G Lobo, Claudio F. Lima, and Zbigniew Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer, Berlin, 2007. ISBN 3540694315. doi: 10.1007/978-3-540-69432-8_2.

[49] Ralph Nelson Elliott. *Nature's Law*. Snowball Publishing, 2010.

[50] Ralph Nelson Elliott. *The Wave Principle*. Snowball Publishing, 2012.

[51] Robert Engle. Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of UK Inflation. *Econometrica: Journal of the Econometric Society*, 50(4):987–1007, 1982.

[52] Eugene F. Fama. Mandelbrot and the Stable Paretian Hypothesis. *The Journal of Business*, 36(4):420–429, 1963. ISSN 00219398; 15375374. URL `http://www.jstor.org/stable/2350971`.

[53] Hongbing Fang, Qian Wang, Yi-Cheng Tu, and Mark F Horstemeyer. An Efficient Non-dominated Sorting Method for Evolutionary Algorithms. *Evolutionary Computation*, 16(3):355–384, 2008. ISSN 1063-6560. doi: 10.1162/evco.2008.16.3.355. URL `https://pdfs.semanticscholar.org/d506/d6241ec2b70c8144d6e8a5a5a41f67ed5be7.pdf`.

[54] Iztok Fister, Xin She Yang, Janez Brest, and Dušan Fister. A brief review of nature-inspired algorithms for optimization. *Elektrotehniski Vestnik/Electrotechnical Review*, 80(3):116–122, 2013. ISSN 00135852. doi: 10.4249/scholarpedia.1461.

[55] P J Flynn, Anil K. Jain, M N Murty, P J Flynn, Azriel Rosenfeld, K Bow-

yer, and N Ahuja. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):264–323, 1999. ISSN 03600300. doi: 10.1145/331499.331504.

[56] David B. Fogel. *Evolutionary Computation: The Fossil Record*. IEEE Press, 1998. ISBN 9780780334816. URL https://dl.acm.org/citation.cfm?id=521840.

[57] Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, New York, 1966. URL https://cds.cern.ch/record/107769?ln=en.

[58] Carlos M Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Icga*, 93(July):416–423, 1993. ISSN 14639076. doi: citeulike-article-id:2361311. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.9077{&}rep=rep1{&}type=pdf.

[59] Eric S. Fraga and Oluwamayowa Amusat. Understanding the impact of constraints: A rank based fitness function for evolutionary methods. In Zhigljavsky A. Pardalos P. and Žilinskas J., editors, *Advances in Stochastic and Deterministic Global Optimization*, chapter 1, pages 243–254. Springer, Cham, Zurich, 2016. ISBN 19316828 (ISSN). doi: 10.1007/978-3-319-29975-4_13. URL http://link.springer.com/10.1007/978-3-319-29975-4{_}13.

[60] AS Fraser. Simulation of Genetic Systems by Automatic Digital Computers I. Introduction. *Australian Journal of Biological Sciences*, 10 (4):484, 1957. ISSN 0004-9417. doi: 10.1071/BI9570484. URL http://www.publish.csiro.au/?paper=BI9570484.

[61] R. M. Friedberg. A Learning Machine: Part I. *IBM Journal of Research and Development*, 2(1):2–13, 1958. ISSN 0018-8646. doi: 10.1147/rd.21.0002. URL http://ieeexplore.ieee.org/document/5392654/.

[62] R. M. Friedberg, B. Dunham, and J. H. North. A Learning Machine: Part II. *IBM Journal of Research and Development*, 3(3):282–287, jul 1959. ISSN 0018-8646. doi: 10.1147/rd.33.0282. URL http://ieeexplore.ieee.org/document/5392566/.

[63] Rene Garcia and Pierre Perron. An Analysis of the Real Interest Rate under Regime Shifts. *The Review of Economics and Statistics*, 78(1): 111–125, 1996. URL http://www.jstor.org/stable/2109851.

[64] S. García, Alberto Fernández, Julian Luengo, F. Herrera, and Æ J Æ Luengo. A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Computing*, 13(10):959–977, 2009. ISSN 14327643. doi: 10.1007/s00500-008-0392-y.

[65] David E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989. URL https://pdfs.semanticscholar.org/2e62/d1345b340d5fda3b092c460264b9543bc4b5.pdf.

[66] David E Goldberg and Kalyanmoy Deb. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Foundations of Genetic Algorithms*, 1:69–93, 1991. doi: 10.1.1.101.9494.

[67] David E Goldberg and Robert Lingle. Alleles, Loci, and the Traveling Salesman Problem. In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159, Hillsdale, NJ, 1985. L. Erlbaum Associates.

[68] Weiyu Guo and Mark E. Wohar. Identifying Regime Changes in Market Volatility. *Journal of Financial Research*, 29(1):79–93, mar 2006. ISSN 0270-2592. doi: 10.1111/j.1475-6803.2006.00167.x. URL http://doi.wiley.com/10.1111/j.1475-6803.2006.00167.x.

[69] Donatien Hainaut. A fractal version of the Hull-White interest rate model. *Economic Modelling*, 31(1989):323–334, 2013. ISSN 02649993. doi: 10.1016/j.econmod.2012.11.041. URL http://dx.doi.org/10.1016/j.econmod.2012.11.041.

[70] David Hallac, Peter Nystrup, and Stephen Boyd. Greedy Gaussian Segmentation of Multivariate Time Series. arXiv preprint. 2016. URL http://arxiv.org/abs/1610.07435.

[71] James Douglas Hamilton. Rational-Expectations Econometric Analysis of Changes in Regime. *Journal of Economic Dynamics and Control*, 12 (2/3):385–423, 1988. ISSN 0165-1889.

[72] James Douglas Hamilton. A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica: Journal of the Econometric Society*, 57(2):357–384, 1989. URL http://www.jstor.org/stable/1912559.

[73] James Douglas Hamilton. Analysis of Time Series Subject to Changes in Regime. *Journal of Econometrics*, 45:39–70, 1990. URL http://www.sciencedirect.com/science/article/pii/0304407690900939.

[74] Julia Handl and Joshua Knowles. An Evolutionary Approach to Multiobjective Clustering. *IEEE Transactions on Evolutionary Computation*, 11 (1):56–76, 2007. ISSN 1089778X. doi: 10.1109/TEVC.2006.877146.

[75] Georges R Harik and Fernando G Lobo. A parameter-less genetic algorithm. In *GECCO'99 Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, pages 258–265, San Francisco, 1999. Morgan Kaufmann. ISBN 1-55860-611-4. doi: 10.1016/j.ins.2003.03.029. URL https://pdfs.semanticscholar.org/78f6/0aecd400fe57652d3ef01834a0865ab8289c.pdf.

[76] Kassel Hingee and Marcus Hutter. Equivalence of Probabilistic Tournament and Polynomial Ranking Selection. In *IEEE Congress on Evolution-*

ary Computation (CEC 2008), pages 564–571, 2008. ISBN 9781424418237. doi: 10.1109/CEC.2008.4630852.

[77] Ekatarina A. Holdener. *The Art of Parameterless Evolutionary Algorithms*. Phd thesis, Missouri University of Science and Technology, 2008.

[78] John Henry Holland. Outline for a Logical Theory of Adaptive Systems. *Journal of the ACM*, 9(3):297–314, 1962. ISSN 00045411. doi: 10.1145/321127.321128. URL `http://portal.acm.org/citation.cfm?doid=321127.321128`.

[79] John Henry Holland. Nonlinear Environments Permitting Efficient Adaptation. In *Proceedings of the Symposium on Computer and Information Sciences II*, pages 147–164, 1966.

[80] John Henry Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor MI, 1975. ISBN 0262581116. doi: 10.1137/1018105.

[81] Yi Hong, Sam Kwong, Qingsheng Ren, and Xiong Wang. A Comprehensive Comparison Between Real Population Based Tournament Selection and Virtual Population Based Tournament Selection. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 445–452, 2007. ISBN 1424413400.

[82] Harold Edwin Hurst. Long-term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers*, 116:770–808, 1951.

[83] Julien Idier. (Re)correlation: a Markov switching multifractal model with time varying correlations. Working paper. 2009. URL `https://papers.ssrn.com/sol3/papers.cfm?abstract{_}id=1580075`.

[84] Julien Idier. short-term comovements in stock markets: the use of Markov-switching multifractal models. *The European Journal of Finance*, 17(1): 27–48, 2011. ISSN 1351-847X. doi: 10.1080/13518470903448440.

[85] Carla Inclan and GC Tiao. Use of Cumulative Sums of Squares for Retrospective Detection of Changes of Variance. *Journal of the American Statistical Association*, 89(427):913–923, 1994. URL `http://www.tandfonline.com/doi/abs/10.1080/01621459.1994.10476824`.

[86] Nor Jaini and Sergey Utyuzhnikov. Trade-off ranking method for multi-criteria decision analysis. *Journal of Multi-Criteria Decision Analysis*, 24 (3-4):121–132, may 2017. ISSN 10991360. doi: 10.1002/mcda.1600. URL `http://doi.wiley.com/10.1002/mcda.1600`.

[87] Mikkel T. Jensen. Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503–515, oct 2003. ISSN 1089778X. doi: 10.1109/TEVC.2003.817234. URL `http://ieeexplore.ieee.org/document/1237166/`.

[88] Chiang Kao. Weight determination for consistently ranking alternatives in multiple criteria decision analysis. *Applied Mathematical Modelling*, 34 (7):1779–1787, 2010. ISSN 0307904X. doi: 10.1016/j.apm.2009.09.022. URL `http://dx.doi.org/10.1016/j.apm.2009.09.022`.

[89] Giorgos Karafotias, Mark Hoogendoorn, and A. E. Eiben. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015. ISSN 1089778X. doi: 10.1109/TEVC.2014.2308294.

[90] Graham Kendall. Is Evolutionary Computation Evolving Fast Enough? *IEEE Computational Intelligence Magazine*, 13(2):42–51, 2018. ISSN 1556-603X. doi: 10.1109/MCI.2018.2807019.

[91] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An Online Algorithm for Segmenting Time Series. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 289–296, San Jose, CA, 2001. IEEE. ISBN 0-7695-

1119-8. doi: 10.1109/ICDM.2001.989531. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=989531.

[92] Jeremy Kepner. *Parallel MATLAB for Multicore and Multinode Computers*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009. ISBN 978-0-89871-673-3. doi: 10.1137/1.9780898718126. URL http://epubs.siam.org/doi/book/10.1137/1.9780898718126.

[93] Padmavathi Kora and Priyanka Yadlapalli. Crossover Operators in Genetic Algorithms: A Review. *International Journal of Computer Applications*, 162(10):34–36, 2017. ISSN 09758887. doi: 10.5120/ijca2017913370. URL http://www.ijcaonline.org/archives/volume162/number10/kora-2017-ijca-913370.pdf.

[94] John R. Koza. Hierarchical Genetic Algorithms Operating on Populations of Computer Programs. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, pages 768–774, Detroit, MI, 1989. Morgan Kaufmann. ISBN 9780470544600. doi: 10.1109/9780470544600.ch22.

[95] John R. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992. ISBN 0262111705.

[96] John R. Koza. *Genetic programming II : Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994. ISBN 0262111896.

[97] Saku Kukkonen and Kalyanmoy Deb. A Fast and Effective Method for Pruning of Non-dominated Solutions in Many-Objective Problems. In *Parallel Problem Solving from Nature - PPSN IX*, pages 553–562. Springer, Berlin, Heidelberg, 2006. doi: 10.1007/11844297_56. URL http://link.springer.com/10.1007/11844297{_}56.

[98] H T Kung, F Luccio, and F P Preparata. On Finding the Maxima of a Set of Vectors. *Journal of the ACM*, 22(4):469–476, 1975. ISSN 00045411. doi:

10.1145/321906.321910. URL http://www.eecs.harvard.edu/{~}htk/publication/1975-jacm-kung-luccio-preparata.pdf.

[99] William B. Langdon. *Genetic Programming and Data Structures*. Kluwer Academic Publishers, Boston, 1998. ISBN 9780792381358.

[100] William B. Langdon and Riccardo Poli. Fitness causes bloat. In *Soft Computing in Engineering Design and Manufacturing*, pages 13–22. Springer, London, 1998. ISBN 3540643605. doi: 10.1007/BFb0055926.

[101] William B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer-Verlag, Berlin, 2002. URL http://discovery.ucl.ac.uk/124583/.

[102] M. Laumanns, Lothar Thiele, and E. Zitzler. Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182, 2004. ISSN 1089-778X. doi: 10.1109/TEVC.2004.823470.

[103] Shane Legg, Marcus Hutter, and Akshat Kumar. Tournament versus Fitness Uniform Selection. In *Congress on Evolutionary Computation (CEC 04)*, pages 2144–2151, 2004. ISBN 0-7803-8515-2. doi: 10.1109/CEC.2004.1331162. URL http://arxiv.org/abs/cs/0403038.

[104] Ke Li, Kalyanmoy Deb, Qingfu Zhang, and Qiang Zhang. Efficient Nondomination Level Update Method for Steady-State Evolutionary Multiobjective Optimization. *IEEE Transactions on Cybernetics*, 47(9):2838–2849, 2017. ISSN 21682267. doi: 10.1109/TCYB.2016.2621008.

[105] Miqing Li, Shengxian Yang, and Xiaohui Li. A Performance Comparison Indicator for Pareto Front Approximations in Many-Objective Optimization. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, 2015. doi: 10.1145/2739480.2754687.

[106] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[107] Ruipeng Liu. *Multivariate Multifractal Models : Estimation of Parameters and Applications to Risk Management*. PhD thesis, Kiel University, 2008.

[108] Ruipeng Liu and Thomas Lux. Higher Dimensional Multifractal Processes : A GMM Approach. In *ICCEF 2010*. Society for Computational Economics, 2012.

[109] Ruipeng Liu and Thomas Lux. Non-homogeneous volatility correlations in the bivariate multifractal model. *The European Journal of Finance*, (May 2015):1–21, 2014. ISSN 1351-847X. doi: 10.1080/1351847X.2014.897960. URL `http://www.tandfonline.com/doi/abs/10.1080/1351847X.2014.897960`.

[110] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013. ISSN 08936080. doi: 10.1016/j.neunet.2013.01.012. URL `http://arxiv.org/abs/1203.0453`.

[111] Farhad Hosseinzadeh Lotfi, Mohsen Rostamy-Malkhalifeh, Nazila Aghayi, Zahra Ghelej Beigi, Kobra Gholami, Farhad Hosseinzadeh Lotfi, Mohsen Rostamy-malkhalifeh, Nazila Aghayi, Zahra Ghelej Beigi, and Kobra Gholami. An improved method for ranking alternatives in multiple criteria decision analysis. *Applied Mathematical Modelling*, 37(1-2):25–33, jan 2013. ISSN 0307904X. doi: 10.1016/j.apm.2011.09.074. URL `https://www.sciencedirect.com/science/article/pii/S0307904X11006330`.

[112] Alexandre Lung-yut fong, Céline Lévy-Leduc, and Olivier Cappé. Homogeneity and change-point detection tests for multivariate data using rank statistics. arXiv preprint. 2011. URL `http://arxiv.org/abs/1107.1971`.

[113] Thomas Lux. The Markov-switching multifractal model of asset returns: GMM estimation and linear forecasting of volatility. *Journal of Business & Economic Statistics*, 26(2):194–210, 2008. ISSN 07350015. doi: 10.1198/073500107000000403. URL http://wrap.warwick.ac.uk/1749/.

[114] Thomas Lux. Flexible and Robust Modelling of Volatility Comovements : A Comparison of Two Multifractal Models. Kiel Working Papers. 2010.

[115] Thomas Lux and Taisei Kaizoji. Forecasting volatility and volume in the Tokyo Stock Market: Long memory, fractality and regime switching. *Journal of Economic Dynamics and Control*, 31(6):1808–1843, 2007. ISSN 01651889. doi: 10.1016/j.jedc.2007.01.010.

[116] James MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967. ISBN 1595931619. doi: citeulike-article-id:6083430.

[117] Benoit B. Mandelbrot. The variation of certain speculative prices. *The Journal of Business*, 36(4):394–419, 1963. ISSN 00219398; 15375374. URL http://link.springer.com/chapter/10.1007/978-1-4757-2763-0{_}14.

[118] Benoit B. Mandelbrot. The variation of some other speculative prices. *Journal of Business*, 40(4):393–413, 1967. URL http://www.jstor.org/stable/2351623.

[119] Benoit B. Mandelbrot. How long is the coast of Britain? Statistical self-similarity and fractional dimension. *Science*, 156:636–638, 1967. ISSN 00368075. doi: 10.1126/science.156.3775.636. URL https://users.math.yale.edu/users/mandelbrot/web{_}pdfs/howLongIsTheCoastOfBritain.pdf.

[120] Benoit B. Mandelbrot and John W. Van Ness. Fractional Brownian Mo-

tions, Fractional Noises and Applications. *SIAM Review*, 10(4):422–437, 1968. ISSN 0036-1445. doi: 10.1137/1010093.

[121] Benoit B. Mandelbrot, Adlai J. Fisher, and Laurent E. Calvet. A Multifractal Model of Asset Returns. Cowles Foundation Discussion Paper #1164. 1997. URL `https://users.math.yale.edu/{~}bbm3/web{_}pdfs/Cowles1164.pdf`.

[122] Rosario N. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B*, 11:193–197, 1999. URL `http://link.springer.com/article/10.1007/s100510050929`.

[123] David S. Matteson and Nicholas A. James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 109(505):334–345, 2014. ISSN 1537274X. doi: 10.1080/01621459.2013.849605.

[124] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33:1455–1465, 2000. ISSN 00313203. doi: 10.1016/S0031-3203(99)00137-5.

[125] Angelo Melino and Stuart M. Turnbull. Pricing Foreign Currency Options with Stochastic Volatility. *Journal of Econometrics. Jul/Aug90*, 45(1/2):239–265. 27p. 7 Charts, 1990. ISSN 0304-4076. URL `http://www.sciencedirect.com/science/article/pii/0304407690901008`.

[126] Brad L Miller and David E Goldberg. Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems*, 9(3):193–212, 1995.

[127] Lars Black Mogensen. *Models of Changing Volatility : A Multifractal Approach*. PhD thesis, Aarchus School of Business, 2011.

[128] Raffaello Morales, T. Di Matteo, and Tomaso Aste. Dependency structure and scaling properties of financial time series are related. *Scientific reports*, 4:4589, 2014. ISSN 2045-2322. doi: 10.1038/

srep04589. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3980463{&}tool=pmcentrez{&}rendertype=abstract.

[129] Tatsuya Motoki. Calculating the expected loss of diversity of selection schemes. *Evolutionary Computation*, 10(4):397–422, 2002. ISSN 1063-6560. doi: 10.1162/106365602760972776. URL https://pdfs.semanticscholar.org/6a15/670db41fa9a398454cb9dfef77bf4ba4eea4.pdf.

[130] Heinz Mühlenbein and Dirk Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evolutionary Computation*, 1(1):25–49, 1993. ISSN 1063-6560. doi: 10.1162/evco.1993.1.1.25. URL http://www.mitpressjournals.org/doi/10.1162/evco.1993.1.1.25.

[131] Noemi Nava, T. Di Matteo, and Tomaso Aste. Anomalous volatility scaling in high frequency financial data. arXiv preprint. 2015. URL https://arxiv.org/pdf/1503.08465.pdf.

[132] M. Nowostawski and Riccardo Poli. Parallel genetic algorithm taxonomy. In *1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings*, pages 88–92, 1999. ISBN 0-7803-5578-4. doi: 10.1109/KES.1999.820127.

[133] Peter Nystrup, Bo William Hansen, Henrik Madsen, and Erik Lindström. Detecting change points in VIX and S&P 500: A new approach to dynamic asset allocation. *Journal of Asset Management*, 17(5):361–374, 2016. ISSN 1470-8272. doi: 10.1057/jam.2016.12. URL http://link.springer.com/10.1057/jam.2016.12.

[134] Author E S Page. Continuous Inspection Schemes. *Biometrika*, 41(1):100–115, 1954. ISSN 1708-0428; 0960-8923. doi: 10.1007/s11695-010-0205-0[doi].

[135] Jon D. Pelletier and Donald L. Turcotte. Self-affine time series: II. Applications and models. In Renata Dmowska and Barry Saltzman, editors, *Advances in Geophysics Vol. 40*, pages 91–166. Academic Press, Cambridge, MA, 1997. URL http://arxiv.org/abs/physics/9705038.

[136] Nikolaos Ploskas and Nikolaos Samaras. *GPU programming in MATLAB*. Morgan Kaufmann, 2016. ISBN 9780128051320. URL https://uk.mathworks.com/support/books/gpu-programming-in-matlab-ploskas.html.

[137] Riccardo Poli and William B. Langdon. Backward-chaining evolutionary algorithms. *Artificial Intelligence*, 170(11):953–982, 2006. ISSN 00043702. doi: 10.1016/j.artint.2006.04.003.

[138] Riccardo Poli, William B. Langdon, and N.F. McPhee. A Field Guide to Genetic Programing. E-book, 2008. URL http://www.essex.ac.uk/wyvern/2008-04/WyvernApril087126.pdf.

[139] Riccardo Poli, Nicholas Freitag McPhee, and Leonardo Vanneschi. Elitism Reduces Bloat in Genetic Programming. *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 1343–1344, 2008. doi: 10.1145/1389095.1389355. URL http://cswww.essex.ac.uk/staff/poli/papers/PoliMcPheeVanneschiGECCO2008{_}Elitism{_}Reduces{_}Bloat.pdf.

[140] Pier Francesco Procacci and Tomaso Aste. Forecasting market states. arXiv preprint. 2018. URL http://arxiv.org/abs/1807.05836.

[141] I. Rechenberg. Cybernetic solution path of an experimental problem. *Journal of Theoretical Biology*, 215:441–448, 1965. ISSN libr. transl. 1122. doi: 10.1109/9780470544600.ch8. URL https://ci.nii.ac.jp/naid/10000137330/http://www.citeulike.org/group/1662/article/1505176.

[142] Lewis Fry Richardson. The problem of contiguity : An appendix to statistics of deadly quarrels. *General systems: Yearbook of the Society for the Advancement of General Systems Theory*, 6:139–187, 1961.

[143] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, nov 1987. ISSN 03770427. doi: 10.1016/0377-0427(87)90125-7. URL `http://linkinghub.elsevier.com/retrieve/pii/0377042787901257`.

[144] Martin Rypdal and Ola Løvsletten. Multifractal modeling of short-term interest rates. arXiv preprint. 2011. URL `http://arxiv.org/abs/1111.5265`.

[145] Abdellah Salhi and Eric S. Fraga. Nature-inspired optimisation approaches and the new plant propagation algorithm. In *The International Conference on Numerical Analysis and Optimization*, pages K2 1–8, Jakarta, 2011. URL `http://repository.essex.ac.uk/9974/1/paper.pdfhttp://repository.essex.ac.uk/9974/`.

[146] A Sansó, V Aragó, and JL Carrion. Testing for changes in the unconditional variance of financial time series. *Revista de Economía Financiera*, 4:32–53, 2004. URL `http://www.aefin.es/AEFIN{_}data/articulos/pdf/A4-2{_}443809.pdf`.

[147] Kumara Sastry and David E Goldberg. Modeling Tournament Selection with Replacement using Apparent Added Noise. In *Proceedings of ANNIE 2001*, pages 129–134, St. Lois, MO, 2001. ASME Press.

[148] Lothar M. Schmitt. Theory of genetic algorithms. *Theoretical Computer Science*, 259(1-2):1–61, 2001. ISSN 03043975. doi: 10.1016/S0304-3975(00)00406-0.

[149] Artem Sokolov and Darrell Whitley. Unbiased Tournament Selection. In *Proceedings of the 2005 Conference on Genetic and Evolutionary*

Computation (GECCO 05), pages 1131–1138, 2005. ISBN 1595930108. doi: 10.1145/1068009.1068198. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.1575{&}rep=rep1{&}type=pdf`.

[150] Won Min Song, T. Di Matteo, and Tomaso Aste. Hierarchical Information Clustering by Means of Topologically Embedded Graphs. *PLoS ONE*, 7 (3):1–14, 2012. ISSN 19326203. doi: 10.1371/journal.pone.0031929.

[151] N Srinivas and Kalyanmoy Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221—-248, 1995. ISSN 1063-6560. doi: 10.1162/evco.1994.2.3.221. URL `http://www.mitpressjournals.org/doi/abs/10.1162/evco.1994.2.3.221`.

[152] H Steinhaus. Sur la division des corps materiels en parties. *Bulletin of the Polish Academy of Sciences*, IV:801–804, 1956. ISSN 0001-4095. doi: citeulike-article-id:3738255.

[153] Jung W. Suh and Youngmin Kim. *Accelerating MATLAB with GPU computing : a primer with examples*. Morgan Kaufmann, 2013. ISBN 9780124079168.

[154] Makram Talih and Nicolas Hengartner. Structural learning with time-varying components: Tracking the cross-section of financial time series. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67(3):321–341, 2005. ISSN 13697412. doi: 10.1111/j.1467-9868.2005.00504.x.

[155] Suqin Tang, Zixing Cai, and Jinhua Zheng. A Fast Method of Constructing the Non-dominated Set: Arena's Principle. In *2008 Fourth International Conference on Natural Computation*, volume 1, pages 391–395. IEEE, oct 2008. ISBN 9780769533049. doi: 10.1109/ICNC.2008.823. URL `http://ieeexplore.ieee.org/document/4666875/`.

[156] Michele Tumminello, Fabrizio Lillo, and Rosario N. Mantegna. Correlation, hierarchies, and networks in financial markets. *Journal of Economic Behavior & Organization*, 75(1):40–58, 2010. URL `http://www.sciencedirect.com/science/article/pii/S0167268110000077`.

[157] Alan M. Turing. Computing Machinery and Intelligence. *Mind*, 59(236):433–460, 1950. ISSN 0026-4423. doi: http://dx.doi.org/10.1007/978-1-4020-6710-5_3. URL `papers2://publication/uuid/E74CAAC6-F3DD-47E7-AEA6-5FB511730877`.

[158] Tea Tusar and Bogdan Filipic. Visualization of Pareto Front Approximations in Evolutionary Multiobjective Optimization: A Critical Review and the Prosection Method. *IEEE Transactions on Evolutionary Computation*, 19(2):225–245, 2015. ISSN 1089778X. doi: 10.1109/TEVC.2014.2313407.

[159] Victor Venema. p-model multifractal time series. Web resource, 2006. URL `http://www2.meteo.uni-bonn.de/staff/venema/themes/surrogates/pmodel/`.

[160] Darrell Whitley, Soraya Rana, and Robert B Heckendorn. The Island Model Genetic Algorithm: On Separability, Population Size and Convergence. *Journal of Computing and Information Technology*, 7:33–47, 1999. doi: 10.1.1.36.7225. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.7225{&}rep=rep1{&}type=pdf`.

[161] L Darrell Whitley. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In *Proceedings of the Third International Conference on Genetic Algorithms*, volume 89, pages 116–123, 1989. ISBN 1-55860-006-3. doi: 10.1.1.18.8195.

[162] David H. Wolpert and William G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1): 67–82, 1997. ISSN 1089778X. doi: 10.1109/4235.585893.

[163] Huayang Xie and Mengjie Zhang. Parent Selection Pressure Auto-tuning for Tournament Selection in Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 17(1):1–19, 2013. ISSN 1089778X. doi: 10.1109/TEVC.2011.2182652.

[164] Huayang Xie, Mengjie Zhang, Peter Andreae, and Mark Johnson. An Analysis of Multi-Sampled Issue and No-Replacement Tournament Selection. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 1323–1330, 2008. ISBN 9781605581309. doi: 10.1145/1389095.1389347.

[165] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. URL `http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4358754`.

[166] Jian Zhong and Xin Zhao. Modeling Complicated Behavior of Stock Prices Using Discrete Self-Excited Multifractal Process. *Systems Engineering Procedia*, 3(2011):110–118, 2012. ISSN 22113819. doi: 10.1016/j.sepro.2011.11.015.

[167] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011. ISSN 22106502. doi: 10.1016/j.swevo.2011.03.001. URL `http://linkinghub.elsevier.com/retrieve/pii/S2210650211000058`.

Nomenclature

α	proportion of the population sampled for tournaments in a generation
β	expected growth rate of the elite population: $E[r_{1g}] = \beta \cdot r_{1g}$
$\mathbf{\Omega}$	set of constraint parameters
$\check{N}_t$	total population available across all permutations after the t-th tournament in Scheme C
$\mathbf{F}_\kappa$	fine-grained partitioning
$\mathbf{K}_\kappa$	coarse-grained partitioning
$\mathbf{S}_k$	coarse-grained subseries
$\mathbf{W}_j$	fine-grained subseries
$\mathrm{R_g}$	vector of rank sizes: $E[\mathrm{R_g}] = \left[E[r_{1g}], E[r_{2g}], \ldots, E[r_{ng}]^{\mathrm{T}}\right]$.
ω	number of objectives
ρ_g	ratio of the number of individuals in rank 1 to the overall population after selection at generation g: $\rho_g = r_{1g}/N$
τ	Tournament size parameter
$\tilde{N}$	number of unique fitness vectors in the population
C_T	value of the solution obtained from an algorithm
G	total number of generations in a run

G^* expected number of generations to convergence

I number of subgroups (islands)

N population size

n number of ranks or fitness values

O set of vectors of fitness values

p^{samp} selection probability

p^{sel} sampling probability

p^{vic} victory probability

R_{ig}^{+} total number of individuals of rank i and above at generation g

T_C computational cost of an algorithm

K_κ coarse-grained partitioning

ARCH(p) p-th order autoregressive conditionally heteroskedastic model

CPD Change point detection

CUDA NVIDIA's language for GPU computing

DBHT Directed Bubble Hierarchical Tree

DGP data-generating process

DM decision-maker

DSAV discrete autoregressive tochastic volatility

EA Evolutionary algorithm

ES evolutionary strategies

FBM fractionally integrated Brownian motion

FITS fractionally integrated time series

GA genetic algorithms

GARCH(p, q) generalized autoregressive conditionally heteroskedastic model

GMM Generalized Method of Moments

GP genetic programming

ICSS iterated cumulative sum of squares

L Number of observations per day

MC Monte Carlo simulation

MCDA multi-criteria decision analysis

MDL Minimum description length

MEX Matlab executable file type

ML maximum likelihood

MMAR Multifractal Model of Asset Returns

MOEA multiobjective evolutionary algorithm

MSE multiscale entropy

MSM Markov-Switching Multifractal

NFL no free lunch theorems

PF Pareto front

PF Pareto front

PGA parallel genetic algorithm

PMX partially-mapped crossover

PSO particle swarm optimization

SV Stochastic volatility

T	number of samples in time series
TSP	travelling salesman problem