ABSTRACT

AN ANALYSIS OF TWO-POPULATION COEVOLUTIONARY COMPUTATION

Elena Popovici

George Mason University, 2006

Thesis Director: Dr. Kenneth De Jong

Coevolutionary computation (CoEC) is the subfield of evolutionary computation (EC) centered around the notion of interaction among simultaneously evolving entities. While it promises important problem-solving advantages, coevolution also brings many challenges.

A practitioner trying to use a coevolutionary algorithm (CoEA) to solve a *problem* is generally interested in how the choices made in designing the *algorithm* affect its *performance* on that particular problem. In other words, they would like to have information about the dependency: **problem properties + algorithm properties → performance**. For traditional evolutionary algorithms (EAs), our understanding of this dependency has reached reasonably satisfactory levels. For CoEAs it has not, as they have proven notoriously more complex and less intuitive. The main contribution of my dissertation is advancing the understanding of this dependency for traditional two-population coevolutionary algorithms.

The way I achieve this is through extensive analysis that *connects* algorithm, problem and performance through one key aspect: **dynamics**. While the importance of understanding the dynamics of coevolutionary systems has been pointed out by previous research, this dissertation is the first study that "glues" all four pieces together.

Additionally, an important feature of the analysis is that it spans subareas of CoEC that were previously studied independently (compositional cooperative and test-based competitive). It bridges them by identifying a problem property and introducing tools for analyzing this property that are applicable across subareas, thus providing a more holistic perspective of the field of CoEC.

The analysis is performed both for previously unstudied aspects of CoEAs and for ones that have been investigated using other techniques. The power of the new analysis approach is particularly visible in the latter case, where it is shown to explain prior "mysterious" results.

Chapter 1

Introduction

This chapter starts with a short description of what evolutionary and coevolutionary computation are and what they are used for. It then presents the motivations for the work conducted and the contributions resulted from this work. The chapter concludes with a roadmap for the contents of the dissertation.

1.1 Evolutionary Computation

Evolutionary Computation (EC) (De Jong, 2006) is the field of computer science that studies computational methods inspired by the Darwinian concept of evolution through natural selection. These computational methods, generally referred to as evolutionary algorithms (EAs), have been successfully used for a variety of purposes such as search, optimization, learning, adaptation, artificial life and even artificial creativity.

The algorithms maintain a set (*population*) of potential problem solutions (*individuals*) and refine it by repeatedly applying mechanisms of *variation* and *selection* with the goal of finding optimal (or high quality) solutions. The key features of the variation mechanism are *inheritance* ("child" individuals are similar to the "parent" individuals that generated them) and *stochasticity*. The selection mechanism is biased by the quality (*fitness*) of the solutions with respect to the problem solving goal; it can be stochastic or deterministic. The pseudocode for a generic EA is shown below:

```
Initialize population
Evaluate population (i.e. assign fitness to all individuals)
while not Termination criteria
    Select parents from population based on fitness
    Produce children from parents (variation with inheritance)
    Evaluate children
    Select survivors for next population
```

The stochastic nature of evolutionary algorithms creates difficulties for their study. However, many years of research have shed some light on the behavior of EAs, pointing out their general operation patterns, their strengths and weaknesses as well as their sensitivity to various domain features and design decisions. Additionally, this research has exposed a number of possible extensions to EC, which would broaden its area of applicability or improve its performance. One such extension, namely coevolutionary computation, is the topic of this dissertation.

1.2 Coevolutionary Computation

As the next chapter will show, the subject of coevolutionary computation is a complicated one, starting with its definition. For now we shall adopt the simple view that a coevolutionary algorithm (CoEA[1]) is an EA in which the fitness of an individual is dependent on its interactions with other evolving individuals. Coevolutionary computation (CoEC) is the subfield of computer science that is concerned with the study of CoEAs and their application to problem solving. Although, as will be conveyed in this chapter and the next, the field draws a lot from biology, the focus of this dissertation will be on the computational, problem-solving aspects. To convey this more clearly, rather than using the general term coevolution, I will use the following more specific terms:

- CoEAs – referring only to the algorithms;
- CoEC setup – referring to a CoEA being applied to a problem;
- CoEC – referring to the field, encompassing the algorithms, the way they are applied to problem solving and the analysis of CoEC setups.

[1]In the literature, coevolutionary algorithms have actually been abbreviated to CEAs. I prefer CoEA over CEA for two reasons: 1) it is consistent with CoEC; and 2) the letter "C" has been used to abbreviate the word *compact* in CGA (compact genetic algorithm).

The following chapter presents a framework for CoEC that will explain in great detail what CoEAs are and what type of problems they are applied to.

1.2.1 Advantages of CoEAs

Coevolutionary algorithms emerged at the intersection of efforts from two directions: first, the simulation of those processes taking place in nature, where reciprocal evolutionary change occurs between interacting species or populations[2]; second, the need for a computational method to tackle domains in which performance of a potential solution can only be expressed based on interactions with other potential solutions. Later on, a third direction was established by arguing that decomposing a complex problem into smaller pieces and applying an EC method based on interactions between those pieces could improve the problem solving performance.

The second direction was expanded and CoEAs were applied more generally to domains in which the problem solving goal could not be translated into a fitness function for a traditional EA to use. This could be for one of several reasons: 1) the goal was not even theoretically testable (e.g. find a chess playing program beating all other possible programs); 2) testing the goal was computationally intractable (e.g. find a sorting network that sorts all binary input sequences, when the number of inputs is high); 3) the goal was computationally testable, but it was expensive to do so for each evaluation during the algorithm (e.g. in the sorting networks domain when the number of inputs is small (Hillis, 1990)); and 4) the goal may not translate into an evolution-friendly fitness function (e.g. evaluating a game player against a set of expert opponents may provide no gradient for the search). The hope of CoEC was to have an algorithm achieve the problem solving goal without explicitly encoding it internally into fitness.

Here are a few more detailed examples of the situations described above. Consider the task of evolving game playing computer programs. It is extremely hard or even impossible to come up with a fitness function that can assess how good a player is without having it play games with other players. Similarly, in a sorting network domain, it is hard to figure out if a sorting network is correct without running it on all (relevant) input sequences. In

[2]This is the meaning of the term coevolution in biology (Ridley, 1993) and, as we shall see, it is different from the regular usage of the term in EC.

the first case, playing games with all possible players means enumerating the whole search space, which is clearly not an option. In the second case, testing a network on all input sequences is possible, but it is very computationally expensive to do so for each and every network during the evolutionary process. One alternative is to use a fixed set of interacting individuals (players, input sequences, etc.) for fitness assessment. However, it is not clear how such a set should be chosen. And even if one knew and could construct such a set, evolving against it is prone to the well-known effect of over-fitting. Another alternative is to randomly generate interacting individuals for each fitness evaluation or for each generation. In order to avoid high computation costs, the number of random individuals has to be small (compared to the size of the set of all possible interacting individuals), in which case they are no longer a representative sample. Additionally, the noise introduced may be harmful to the search process. CoEAs were sought in hope of avoiding such problems, by providing a dynamic gradient for search to follow. For example, in the sorting networks problem described above, a CoEA would use two populations, one evolving networks and one evolving input sequences. The networks would receive good fitness if they sort many of the input sequences, while the input sequences would receive good fitness if they are not sortable by many networks. For the game playing problem, one could use either this two population approach or, alternatively, use a single population and have programs be evaluated by playing games against other programs in that same population.[3]

Additional motivation for using and studying CoEAs comes from their potential role in generating complexity and diversity. There are two different trends of opinions on this subject, but in both cases the conclusion is that CoEC needs further investigation. Those that believe in its ability to generate complexity and diversity (Sims, 1994; Hillis, 1990; Hamilton, 1982) see it as a solution for chronic problems with regular EAs such as premature convergence (loss of diversity) and difficulties in evolving complex structures. Those that argue that in biology there is a relative lack of empirical evidence to support such claims (Floreano and Nolfi, 1997), see computer-based coevolution as the perfect test-bed for simulating and analyzing what would be hard to study in nature in a timely and controlled manner. The understanding coming out of this simulation-oriented type of research

[3]The debate on whether such a single-population algorithm should be called a CoEA will be discussed in the following chapter.

also bears hope for learning how to construct artificial self-adaptive systems.

1.2.2 Challenges

Thus, coevolutionary algorithms have been implemented and applied with several different goals in mind and their performance varied widely across all these goals. The efforts to improve performance discovered that, while CoEAs were a natural extension to traditional EAs, the intuitions and heuristics we have about the latter did not directly transfer to the former. Research was invested to understand why this was the case. Unfortunately, even the analysis techniques through which knowledge about EAs was acquired failed on many occasions to provide any insights into how CoEAs work. New, CoEC-specific techniques needed to be developed. Section 2.5 provides a review of CoEC analysis.

The most informative analysis techniques proved to be the ones that focused on what happened at run-time in CoEC setups, in other words, the *dynamics*. This type of analysis identified an unfortunately large number of ways in which CoEAs can fail (reviewed in section 2.4.3). These were fairly complex behaviors, sometimes even difficult to characterize, often difficult to detect, and always difficult, if not impossible, to predict.

The important consequence of this situation is that for CoEAs (unlike for traditional EAs) there are currently no design guidelines. A CoEC practitioner has little, if any, information on how to setup their system in order to obtain the desired results or how to change a system in order to improve performance.

1.3 Goal, Hypothesis, Contributions

As conveyed in the previous section, CoEAs' promise is a better approach to several different kinds of problems. Thus, it would be really useful if we could harness these algorithms. However, this has proven a difficult thing to do. Previous research has uncovered some of the mysteries behind coevolutionary failure. But, as I will argue at the end of the following chapter (section 2.6), it has only answered the question "*How can CoEAs fail?*" and not "*Why (in what circumstances) do CoEAs fail?*". Without answering this latter question, one cannot move on to determine how to design successful coevolutionary algorithms. Treating the symptom (*how* they fail) rather than the cause (*why* they fail) is an unreliable

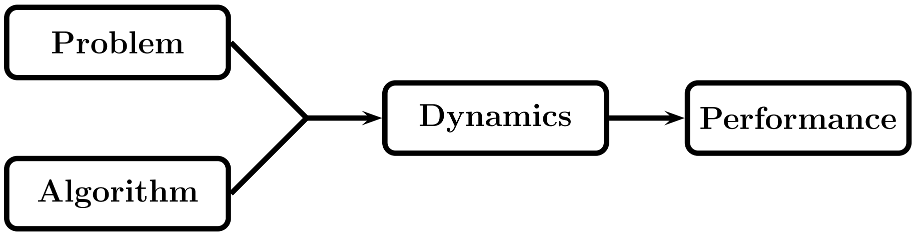


Figure 1.1: Connected approach for CoEC analysis.

strategy (they will fail in a different way on the next problem).

The ***goal*** of this dissertation is to provide insights into the dependency **problem properties** + **algorithm properties** → **performance** in a way that would answer the aforementioned "*why*" question with respect to both failure and success.

My ***hypothesis*** on how to achieve this goal is two-fold:

- in light of previous research, dynamics should hold the key to understanding CoEC setups; but, unlike previous research:

- a holistic analysis approach is required, connecting the interplay of problem properties and algorithm properties to dynamics and dynamics to performance, as shown in Figure 1.1.

Throughout the dissertation, a large amount of evidence is brought in support of the hypothesis, by performing such "connected" analysis and generating answers of the following type: because the problem had property set P and the algorithm had property set A, at run-time, dynamics set D occurred, and this is reflected in the observed performance. Algorithm properties considered include both general EA properties (e.g. population size and elitism) and CoEA-specific properties (e.g. controlling how individuals and populations interact). Some known problem properties are revisited (e.g. linearity, modality) and a new, CoEC-specific property is introduced. Performance is evaluated mainly in terms of quality of the best solution found by the algorithm, given a certain amount of time. Dynamics (time-dependent system properties) include solution-quality-related properties and search-space-exploration properties and are investigated by introducing new analysis tools.

With these means, this dissertation makes the following key contributions to the field of coevolutionary computation:

- The first **integrative, connected analysis** of traditional two-population coevolutionary computation, relating the interplay of problem and algorithm properties to performance through run-time behavior (dynamics), as depicted in Fgure 1.1. This yields **deeper insights** into the behaviors of two-population CoEAs and suggests some **heuristics** for tuning these algorithms to the problems (e.g. when to use larger population sizes and when smaller ones, or when to have populations communicate more often and when less).

- A new, **CoEC-specific problem property**, which I have named *best responses*. This property exerts a major influence on the dynamics and performance of traditional two-population CoEAs. It is orthogonal to previously known properties and this allows it to bridge subareas of CoEC that were so far studied independently (compositional cooperative and test-based competitive, defined in chapter 2).

- New **qualitative and quantitative tools** for analyzing the dynamics of traditional two-population CoEAs. These tools, based on tracking best individuals, are targeted at studying the effect of a problem's best-response curves on the algorithm's behavior and performance.

- A collection of **CoEA test suites** built around the aforementioned problem property.

In addition, the background chapter provides a framework for the maturing field of CoEC. Parts of this framework emerged in recent years, as the CoEC research community joined in a collective effort to eliminate the confusions and controversies that have been historically plaguing the field. As a member of that community, I have been a contributor. Some fragments of this framework can be found scattered in different publications, while others consist only of verbal consensus at conferences and workshops. The background chapter is my attempt to unify this knowledge in a single document.

1.4 Organization

The remainder of this dissertation is organized as follows.

Chapter 2 presents the recently emerged framework for “mature” CoEC. It is, to the best of my knowledge, the first attempt to summarize this framework in writing. In addition, it provides the reader with background material necessary to understand the remainder of the dissertation. In particular, it describes two classes of problems that will be the subject of the analysis in subsequent chapters: compositional cooperative and test-based competitive. This chapter also includes a review of analytical research in CoEC, pointing out open issues, some of which will be addressed by the analysis in this dissertation.

Chapter 3 provides a first example of how dynamics analysis provides insights into the dependency **problem properties + algorithm properties → performance**. It does so in the context of using a CoEA for optimization by means of problem decomposition. The particular algorithm properties under investigation are ones common to simple EAs as well, namely population size and elitism. New tools for analyzing the dynamics of CoEAs are introduced and their use exposes a problem property, named best-response curves (or simply best responses), that has a high influence on performance. To better study this property, a tunable family of problems is constructed and used as a test suite for further analysis both in this chapter and the next. Subsets of the material in this chapter have been published in (Popovici and De Jong, 2005c, 2006a).

Chapters 4 through 6 perform the newly introduced type of analysis for incrementally more complex compositional cooperative problems.

Chapter 4 employs the previously introduced test suite and analysis tools to investigate the performance effects of three CoEA-specific properties (parameters): collaboration method, communication frequency and communication flow. The best-response curves are shown to be influential in all three cases. Additionally, the knowledge gained from the analysis of the test suite is used to make and confirm performance predictions for other problems commonly used as test beds in the literature. In all cases, the problems consist of functions given in closed-form and with known optima. The best responses for these functions can thus be analytically computed and are uniquely defined and continuous. Previous versions of the studies in sections 4.1, 4.2 and 4.3 of this chapter have been published in (Popovici and De Jong, 2005a, 2006b,c), respectively.

Chapter 5 raises the complexity of the problems one level up in terms of possible relationships between the best responses. The first section focuses on discontinuous best responses, the second section on locally-best responses. Two new test suites are introduced that feature these properties. The third section uses a test-suite from the literature to illustrate possible effects of asymmetric best responses. The functions are still given in closed-form, the best responses are computable and the optima are known. The same type of analysis as before is performed for the effect of collaboration methods on the first test suite, the effect of population size on the second test suite and the effect of the starting population on the third test suite.

Chapter 6 raises the complexity level once more, by featuring a problem that is not available in closed form, the best responses are not analytically computable and the optima are not known. Instead, the values of the function to be optimized are the result of a simulation. Some landscape-probing techniques are applied and they reveal that this problem actually combines the features of the test suites from the previous chapter. The performance effects of collaboration methods and population size are once again analyzed, and the results align to a reasonable degree with those obtained on the test suites.

Chapter 7 switches to test-based competitive problems, and shows that best responses are still an influential problem property and the analysis tools introduced are still applicable, thus providing a link between two areas previously studied in isolation. The work in this chapter has been published in (Popovici and De Jong, 2005b).

The dissertation concludes in chapter 8 with a review of contributions and a discussion of limitations and potential extensions of the presented work.

Chapter 2

Background

The field of coevolutionary computation is a fairly young one. While the first papers on the subject date back in the early 1990s, dedicated venues for publication, in the form of workshops and tracks appeared only after 2000. As CoEC researchers began interacting at these venues, a lot of ideological and terminological controversies were raised, painting a fairly confusing picture for the field. Starting in 2004, with a terminology discussion held at the Genetic and Evolutionary Computation Conference (GECCO), a collective effort was set in motion to attempt to improve the situation. Out of this effort, to which I myself contributed, a framework for CoEC has emerged, and a first tutorial on CoEC, authored by myself and Paul R. Wiegand, was presented at the Congress on Evolutionary Computation in 2005. Two more tutorials followed in 2006 at GECCO.

This chapter is the first attempt to summarize this framework in a single document. Additionally, a review of CoEC analysis is included, presented from the perspective of the framework and pointing out open issues that this dissertation addresses. The chapter also serves the purpose of providing the reader with background material needed to understand the remainder of the thesis. Namely, it describes what makes a problem a coevolutionary problem and how coevolutionary algorithms approach such problems. While several classes of coevolutionary problems are described, only two such classes are addressed by the analysis in the dissertation, namely compositional cooperative with ideal team as a solution concept (used in chapters 3 through 6) and test-based competitive with optimum expected utility as a solution concept (used in chapter 7).[1]

[1]The reader interested only in the minimum amount of information may skip subsections *Role Symmetry* and *Infinite (Variable) Number of Roles* from 2.2.1, read only the two solution concepts mentioned above from 2.2.4 and skip sections 2.4 and 2.5 all together.

2.1 Framework Overview

CoEC is a currently maturing field whose boundaries are still being defined. What does or does not deserve the name of coevolution within EC has been (and occasionally still is) a debatable issue among researchers. At the beginnings of the field, there was work that was not labeled as coevolution when first published, but assigned the name later on (Axelrod, 1989; Angeline and Pollack, 1993; Holland, 1985). Additionally, there has been work called coevolution by its authors at publication time, which does not fit any of the current views on CoEC (Juillé, 1995; Angeline and Pollack, 1994).

This confusing situation is becoming more and more a thing of the past. However, it is useful to review its causes, as it was understanding them that led to solutions for constructing the framework described in this chapter.

The first reason was ambiguous cross-field terminology. CoEC and EC itself have drawn inspiration from biology, but during the import process, the ideas and the terms have been altered to suit computational goals. There is a tradeoff to be made between facilitating communication with other sciences (mainly biology) and facilitating research in computer science. Some biological aspects may not be useful for computational purposes and some computational approaches may not be biologically realistic or plausible, yet still be useful. In recent years, the balance seems to have leaned in favor of using those terms that most closely fit the concepts needed to reason about algorithms. Still, it is a good practice, for terms highly overloaded with meanings, to state which semantics are used in a particular context.

While this is especially important when presenting / publishing work for multidisciplinary audiences, with CoEC there are many terms which are have multiple meanings within the field. Thus, the second and, in my opinion, more important reason for past confusion was ambiguous terminology within CoEC. As mentioned in the introduction, CoEC has multiple roots, which for a while developed in parallel, without much exchange of information between them. As a result, separate (and sometimes conflicting) terminologies were introduced. By the time it became apparent that the different subareas had many things in common and could benefit from exchanging information, terms like cooperation, competition or Red Queen each had at least three different uses. Moreover, the meanings associated with the different uses were often defined loosely rather than formally. Additionally, CoEAs turned out to exhibit many phenomena that were not encountered in traditional EAs, and thus they needed to be named. Unfortunately, these phenomena were quite complex, thus difficult to understand and even to describe, causing all the more terminology problems.

The effort to reconcile terminology within CoEC resulted in a major, yet somewhat basic realization that any meaning of any term would fall in one of three separate categories: problem-related, algorithm-related or system-run-time-related.

Indeed, at the highest level of generality, the important and distinct aspects of any computational setup are as follows:

- static aspects – those that are predefined and / or explicitly designed into the system;
  - problem properties;
  - algorithm properties:
- dynamic aspects – those that vary with time while the system is running and may be emerging (there was nothing specifically implemented to generate them) – I shall refer to these as:
  - time-dependent system properties (or, in short, *dynamics*).

The newly emerged CoEC framework is thus structured around these three aspects: problem, algorithm and dynamics. Each of them is discussed in detail in a separate section. Terms are defined in their appropriate category (or categories). In certain cases, it was discovered that a term is really difficult to define in a category in which it had been previously *loosely* used. The recommendation made in such cases is that the term no longer be used in that category.

The presentation in the coming section 2.2 is largely mathematically formalized. This was deliberate: while I realize it may slow down the reader, I felt it was necessary, given the above mentioned terminological issues and the potential for ambiguity of natural language.

2.2 Coevolutionary Problems

Perhaps the most important aspect of a problem specification is some sort of description or characterization of what constitutes a solution to that problem. Although it may sound strange, often times in the past coevolutionary algorithms have been used without such a description of what the solutions sought were. Acknowledging this issue and addressing it was an important part of the effort leading to the CoEC framework, therefore the notion of solution is central to this section.

One aspect specific of CoEA-approachable problems is that, either by nature or by approach, they involve multiple elements that interact in some way. Examples of such elements were used in the previous chapter: players of a game; sorting networks and input sequences; components of a design. A problem specification will include descriptions of these elements, the way they interact *and* how they relate to solutions. This section presents categorizations of problems based on the various aspects of problem specifications.

As a basis for the vocabulary of this presentation, I use a subset of the formalisms proposed by Ficici in chapter 2 of his PhD dissertation (2004), yet I diverge from it.

2.2.1 Domains

An interactive domain $\mathcal{D}$ is a tuple $(n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f)$ with $n \geq 2$.

- $i = 1, 2, ..., n$ are called **roles**;
- $\mathcal{B}_i$ is the **behavior set** associated with role i;
- $b \in \mathcal{B}_i$ is called a **behavior**;
- a tuple $e = (b_1, b_2, ..., b_n) \in \mathcal{E} = \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n$ is called an **event**; behaviors obtain meaning only in the context of some event;
- $\mathcal{O}$ is a set of **outcomes** for events;
- $f : \mathcal{E} \to \mathcal{O}$ is a function defining the outcomes of events; sometimes $\mathcal{O}$ will be equal to $\mathcal{E}$ and f will be the identity function, yet in general they may be different and there may be some "effort" associated with transforming elements of $\mathcal{E}$ into elements of $\mathcal{O}$ through f.

A non-interactive domain is simply a behavior set $\mathcal{B}$ (i.e. $n = 1$ and there is no $\mathcal{O}$ and no f, as these make no sense for $n = 1$). Some non-interactive domains may be reformulated as interactive domains (e.g. through decomposition of the set $\mathcal{B}$ into $\mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n$). Some domains are intrinsically interactive (e.g. games). Using coevolutionary algorithms only makes sense for domains formulated as interactive.

A domain $\mathcal{D}$ becomes measurable when it has some metrics associated with it: $\mathcal{D}_m = (n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f, \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n)$

- $\mathcal{M}_i$ is the **metric set** associated with role i;
- $m_i \in \mathcal{M}_i$ is called a **metric** of behavior; $m_i : \mathcal{E} \times \mathcal{O} \to \mathbb{R}$; for $e = (b_1, b_2, ...b_i, ..., b_n)$ and $o = f(e)$, $m_i(e, o)$ denotes the success of behavior b_i in the context of event e that generated outcome o.

Given e and o, there is usually no effort associated with computing $m_i(e, o)$. Additionally, computing some $m_i(e, o)$ and $m_j(e, o)$ should require only one call to the function f in order to compute $o = f(e)$. This understood, I will define metrics as functions of a single parameter, $m_i : \mathcal{E} \to \mathbb{R}$, $m_i(e) = m_i(e, f(e))$.

The cardinality of $\mathcal{M}_i$ sets and the way the metrics are used in the definition of a problem's solution concept will determine whether whether we are dealing with a single- or multi-objective problem.

Role Symmetry

Role symmetry is a domain property that must be taken into consideration when constructing a CoEA to solve a problem in that domain. I define it here and will return to it later on in section 2.3.1.

Consider a domain $\mathcal{D}_m = (n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f, \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n)$. Roles i and j, $i \neq j$ (for simplicity, assume $i < j$) are called **symmetric** if:

- $\mathcal{B}_i = \mathcal{B}_j = \mathcal{B}$;
- $\exists g : \mathcal{M}_i \to \mathcal{M}_j$ a bijection such that:

- $\forall(b_1, b_2, ..., b_n) \in \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n, \forall m_i \in \mathcal{M}_i : m_i(b_1, b_2, ..., b_i, ..., b_j, ..., b_n) = g(m_i)(b_1, b_2, ..., b_j, ..., b_i, ..., b_n)$.

In other words, roles i and j share the same behavior set and for any two behaviors in this set, their corresponding successes in an event, while possibly different from each other, do not depend on which one is used in role i and which in role j. These successes will still generally depend on the event's behaviors for the other roles.

As defined, the role symmetry relationship is transitive, i.e. if roles i and j are symmetric and roles j and k are symmetric then roles i and k are symmetric. It is also clearly reflexive and symmetric, thus equivalency classes can be defined. Any two roles that do not belong to the same equivalency class are **asymmetric**.

If two roles share the same behavior set, but the success of a (common) behavior depends on which role uses it, I shall argue that equality of the roles' behavior sets is really irrelevant (i.e. the nature of the domain would not change if we simply renamed the behaviors in one of the sets).

Two types of domains are most common in CoEC practice:

- all roles are asymmetric (more accurately, any two roles are asymmetric or, equivalently, all equivalency classes have size 1); or

- all roles are symmetric (more accurately, any two roles are symmetric or, equivalently, there is a single equivalency class which contains all the roles).

The remaining domains are less common and can be described as ones having $n \geq 3$ roles and at least one equivalency class of size $k, 2 \leq k < n$.

When all roles are symmetric, events are multisets of size n rather than tuples:

$$\mathcal{E} = \mu^n(\mathcal{B}) = \{\eta : \mathcal{B} \to \mathbb{N} | \sum_{b \in \mathcal{B}} \eta(b) = n\},$$

and all metric sets can be collapsed into a single one whose elements are of type $m : \mathcal{B} \times \mu^n(\mathcal{B}) \to \mathbb{R}$, $m(b, \eta)$ defined iff $\eta(b) \geq 1$, in which case $m(b, \eta)$ denotes the success of behavior b when participating in the event defined by behavior-multiset η.

Games and multi-agent domains often feature symmetric roles. Games where there is a notion of starting player are by default asymmetric, but they can be reformulated as symmetric by averaging over events that only vary the starting player. Examples of such domains can be found in: (Rosin and Belew, 1995) – Tic-Tac-Toe, Nim, Go; (Angeline and Pollack, 1993) – Tic-Tac-Toe; (Reynolds, 1994) – Tag. Other games are symmetric by nature: numbers games (Watson and Pollack, 2001); robot competitions (Stanley and Miikkulainen, 2002); virtual creatures competitions (Sims, 1994); Tron (Funes and Pollack, 2000); prisoner's dilemma (Miller, 1996; Axelrod, 1989). All of the above involve only two roles.

Examples of domains with two asymmetric roles include: sorting networks and input sequences (Hillis, 1990); cellular automata and initial conditions (Juillé and Pollack, 1998; Pagie and Hogeweg, 2000; Pagie and Mitchell, 2002; Williams and Mitchell, 2005); pursuit and evasion behaviors (Cliff and Miller, 1994, 1995; Floreano and Nolfi, 1997; Wahde and Nordahl, 1998); finite two-player games given by trees (Koza, 1991), attack and defense scenarios (Skolicki et al., 2005). The classic example of domains with more than two asymmetric roles is multi-parameter function optimization (Potter and De Jong, 1994; Potter, 1997; Wiegand, 2004).

Infinite (Variable) Number of Roles Role symmetry allows for an interesting type of domains that can be formalized as $\mathcal{D}_m = (n = \infty, \mathcal{B}_1 = \mathcal{B}_2 = ... = \mathcal{B}, \mathcal{O}, f, \mathcal{M}_1 = \mathcal{M}_2 = ... = \mathcal{M})$, with:

- $\mathcal{E} = \mu(\mathcal{B}) = \{\eta : \mathcal{B} \to \mathbb{N} | \sum_{b \in \mathcal{B}} \eta(b) \geq 1\}$; and
- $\mathcal{M} = \{m : \mathcal{B} \times \mu(\mathcal{B}) \to \mathbb{R} | m(b, \eta) \text{ defined } iff\ \eta(b) \geq 1\}$

In other words, there is a single behavior set, and there is no constraint on the number of behaviors that participate in an event (as long as there is at least one). Order of behaviors is not important and repetitions of the same behavior are allowed (which is why events are modeled as multisets).

Examples of domains that can be viewed in this form include ones involving neural networks with variable number of neurons (Potter, 1997) or multi-agent teams of variable size. Often however, they are the result of reformulation of non-interactive domains of type $(\mathcal{B} = \mu(\mathcal{B}_0), \mathcal{M} = \{m : \mathcal{B} \to \mathbb{R}\})$. This is done by extending m to $m' : \mathcal{B}_0 \times \mu(\mathcal{B}_0) \to \mathbb{R}$,

$\forall b \in \mathcal{B}_0, \eta \in \mu(\mathcal{B}_0)$ such that $\eta(b) \geq 1, m'(b, \eta) = c(m(\eta))$, where c is a *credit assignment* function. Using the identity function for c is a common choice, although it may not always be appropriate. For more on credit assignment, see sections 2.2.3 and 3.3.3 of (Potter, 1997).

2.2.2 Problems

A problem $\mathcal{P}$ is a tuple $(\mathcal{D}_m, \mathcal{S}, \mathcal{C})$ as follows:

- $\mathcal{D}_m$ is a domain with a set of metrics;
- $\mathcal{S}$ is the **space of potential solutions** to the problem; it is some sort of aggregate over some behavior sets of $\mathcal{D}_m$; the definition of $\mathcal{S}$ makes no use of the metric sets;
- $\mathcal{C} : \mathcal{S} \rightarrow \{true, false\}$ is the **solution concept** that partitions the space of potential solutions into solutions and non-solutions; it may be *extensional* or *intensional* (Ficici, 2004); an extensional solution concept specifies which elements of $\mathcal{S}$ are solutions and which are not without stating any properties or reason for the clustering; an intensional solution concept describes what properties a potential solution must have in order to be a solution; this description may depend on behavior sets which do not directly contribute to the space of potential solutions; the definition of $\mathcal{C}$ generally uses some of the available metrics.

I will say an algorithm solved a problem if it identified (at least) one solution.

One important thing to note is that the cost of search for any algorithm attempting to solve such a problem will be measured in number of events assessed (i.e. number of calls to the function f), and not number of points of $\mathcal{S}$ assessed. In general, determining whether an element of $\mathcal{S}$ is a solution will require assessing multiple events.

2.2.3 Clustering Criteria

With such an intricate definition of what a problem is, one can obviously think of numerous criteria for clustering problems. Two of them have recently been devised as important and qualitatively different. They have been introduced as a replacement for the historical yet fuzzy division of the CoEC field into “cooperative coevolution” and “competitive coevolution”. The first is truly a criteria for clustering problems, while the second is actually a criteria for clustering domains.

The first criteria is based on the relationship between $\mathcal{S}$ and $\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n$. Based on this criteria, two main categories of problems have been identified: *test-based* problems (de Jong and Pollack, 2004) and *compositional* problems (Wiegand and Potter, 2006). They are not complementary, however practical instances of problems that do not fit in either category are very rare.

The second criteria distinguishes based on relationships between metrics in $\mathcal{M}_1, \mathcal{M}_2, ...,$ $\mathcal{M}_n$. The two main classes are denoted (whether for the sake of history, reuse, correlation or just for lack of imagination) as “cooperative” domains and “competitive” domains. Yet I urge the reader to withheld for now from making the connection to the historical division of CoEC. This connection will be discussed at the end of this section, as it is more holistic and not referring just to metrics. Note also that these two classes are not complementary either.

The remainder of this subsection discuses these two criteria. However, neither of them is concerned with solution concepts. Therefore, the following subsection, 2.2.4, discusses solution concepts and how they relate to the above clusterings.

Compositional vs. Test-based Problems

Compositional problems are those for which $\mathcal{S} = Agg(\mathcal{B}_1) \times Agg(\mathcal{B}_2) \times ... \times Agg(\mathcal{B}_n)$, where if X is a set, then $Agg(X)$ is a set whose elements are some sort of aggregation of elements of X. Examples include (but are not limited to):

- $Agg(X) = X$;
- $Agg(X) = \wp(X) = \{Y | Y \subseteq X\}$, i.e. the set of all subsets of X, also called the power set of X;
- $Agg(X) = \mu(X) = \{\eta : X \to \mathbb{N}\}$, i.e. the set of multisets over X;
- $Agg(X) = X^p = \{(x_1, x_2, ..., x_p) | x_i \in X, \forall i \in 1..p\}$, i.e. the set of all words of length p using X as an alphabet;

- $Agg(X) = \triangle X = \{p : X \to \mathbb{R} | \int_{x \in X} p(x)\, dx = 1\}$, i.e. the set of all probability distributions over X.

Therefore, the simplest example is $\mathcal{S} = \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n$. Other cases of interest are reviewed in section 2.2.4. These problems are often, although not necessarily, the result of reformulating a non-interactive domain as an interactive one. Examples can be found both in domains with all roles asymmetric, such as multi-parameter function optimization (Potter and De Jong, 1994; Potter, 1997; Wiegand, 2004) and in domains with all roles symmetric, in particular those with an infinite (variable) number of roles, such as variable size neural networks (Potter, 1997). In the first case, roles are parameters, behaviors are parameter values, and potential solutions are tuples of values, one for each parameter. In the second case, behaviors are neurons and potential solutions are multisets of neurons defining complete networks.

Test-based problems are those for which $\mathcal{S} = Agg(\mathcal{B}_i)$ for some role i. Of course, the simplest example is $\mathcal{S} = \mathcal{B}_i$, for some role i. Another is $\wp(\mathcal{B}_i)$. Again, other cases of interest are reviewed in section 2.2.4. Recall that while elements of $\mathcal{S}$ are some sort of aggregates over $\mathcal{B}_i$ alone, the solution concept $\mathcal{C}$ will be described in a way that involves the other roles as well. Thus, the reason for the name test-based is that the roles which do not contribute to the definition of $\mathcal{S}$ but contribute to the definition of $\mathcal{C}$ can be viewed as used for *testing* which elements of $\mathcal{S}$ are solutions and which are not.

Most often, test-based problems have only two roles, say $\mathcal{B}_1$ and $\mathcal{B}_2$, with $\mathcal{S}$ being some aggregate over $\mathcal{B}_1$ and the elements of $\mathcal{B}_2$ being called *tests* (de Jong and Pollack, 2004). Should there be three or more roles in the domain, it is unclear whether elements of all behavior sets not involved in the definition of $\mathcal{S}$ would be worthy to be labeled as tests or tuples of such elements should be labeled as tests.

Test-based problems can also occur both in domains with symmetric roles and in domains with asymmetric roles. The first case is mainly represented by two-player games; the roles correspond to the players, behaviors are strategies and potential solutions are also strategies. Examples can be found in: (Rosin and Belew, 1995) – Tic-Tac-Toe, Nim, Go; (Angeline and Pollack, 1993) – Tic-Tac-Toe; (Reynolds, 1994) – Tag; (Watson and

Pollack, 2001) – numbers games; (Stanley and Miikkulainen, 2002) – robot competitions; (Sims, 1994)- virtual creatures competitions; (Funes and Pollack, 2000) – Tron; (Miller, 1996; Axelrod, 1989) – prisoner's dilemma.

Examples of domains with two asymmetric roles for test-based problems include: sorting networks ($\mathcal{S}$) and input sequences (tests) (Hillis, 1990); cellular automata ($\mathcal{S}$) and initial conditions (tests) (Juillé and Pollack, 1998; Pagie and Hogeweg, 2000; Pagie and Mitchell, 2002; Williams and Mitchell, 2005); pursuit and evasion behaviors (Cliff and Miller, 1994, 1995; Floreano and Nolfi, 1997; Wahde and Nordahl, 1998); finite two-player games given by trees (Koza, 1991). In the last two examples, problems may define $\mathcal{S}$ over either of the roles (tests will correspond to the remaining role).

As previously mentioned and easily noticeable, the compositional and test-based problem classes are not complementary. The most obvious example of problems that do not fit in the above categories are those for which $\mathcal{S} = Agg(\mathcal{B}_{i_1}) \times Agg(\mathcal{B}_{i_2}) \times ... \times Agg(\mathcal{B}_{i_k})$, where $1 < k < n$, $i_j \neq i_l, \forall j, l \in 1..k, j \neq l$. For this to be possible, n must be ≥ 3. To some extent, such problems can be viewed as exhibiting both compositional aspects ($\mathcal{S}$ composes over multiple roles) and test-based aspects (roles not participating in the definition of $\mathcal{S}$ serve for testing $\mathcal{C}$).

Even more generally, one could define $\mathcal{S}$ as some sort aggregate that involves more than one $\mathcal{B}_i$, but does not involve tuple-ing (taking the cross product), e.g. $\mathcal{S} = \mathcal{B}_i \cap \mathcal{B}_j$. I am not aware of any such problems being approached with coevolutionary algorithms.

As a side note, some researchers (Bucci, Wiegand - personal communication) believe we need a better term as a name for the first class. Their argument is that one could regard as composition the act of aggregating over the behavior set of a single role, as is the case for example with $\wp(\mathcal{B}_i)$ for test-based problems. Unfortunately, to date a replacement term has not been proposed.

Cooperative vs. Competitive Domains

Consider $\mathcal{D}_m = (n = 2, \mathcal{B}_1, \mathcal{B}_2, \mathcal{O}, f, \mathcal{M}_1 = \{m_1\}, \mathcal{M}_2 = \{m_2\})$. In other words, the domain has only two roles and the metric sets associated with the roles each have a single element.

Such a domain is called **strictly competitive** if:

$$\begin{gathered}\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2 : \\ m_1(b_1, b_2) > m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) < m_2(b_1', b_2') \\ \wedge \\ m_1(b_1, b_2) = m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) = m_2(b_1', b_2')\end{gathered}$$

Of course, this is equivalent to:

$$\begin{gathered}\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2 : \\ m_1(b_1, b_2) < m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) > m_2(b_1', b_2') \\ \wedge \\ m_1(b_1, b_2) = m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) = m_2(b_1', b_2')\end{gathered}$$

and also to:

$$\begin{gathered}\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2 : \\ m_1(b_1, b_2) > m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) < m_2(b_1', b_2') \\ \wedge \\ m_1(b_1, b_2) < m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) > m_2(b_1', b_2')\end{gathered}$$

One can also define a domain to be **weakly competitive** if:

$$\begin{gathered}\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2 : \\ m_1(b_1, b_2) < m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) > m_2(b_1', b_2')\end{gathered}$$

or alternatively (but not equivalent), if:

$$\begin{gathered}\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2 : \\ m_1(b_1, b_2) > m_1(b_1', b_2') \Leftrightarrow m_2(b_1, b_2) < m_2(b_1', b_2')\end{gathered}$$

So-called constant-sum games from game theory (Osborne and Rubinstein, 1994; Hofbauer and Sigmund, 1998) are examples of strictly competitive domains. A domain $\mathcal{D}_m = (n = 2, \mathcal{B}_1, \mathcal{B}_2, \mathcal{O}, f, \mathcal{M}_1 = \{m_1\}, \mathcal{M}_2 = \{m_2\})$ is a **constant-sum** domain if:

$$\exists c \in \mathbb{R} : \forall b_1 \in \mathcal{B}_1, \forall b_2 \in \mathcal{B}_2, m_1(b_1, b_2) + m_2(b_1, b_2) = c$$

Alternatively, this can be written as:

$$\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2 :$$
$$m_1(b_1, b_2) + m_2(b_1, b_2) = m_1(b_1', b_2') + m_2(b_1', b_2')$$

We could extend the notions of competitiveness to domains where $\mathcal{M}_1$ and $\mathcal{M}_2$ each have more than one metric, if there exists a bijection between $\mathcal{M}_1$ and $\mathcal{M}_2$ and all pairs of metrics in this bijection obey the corresponding set of conditions, as described above for (m_1, m_2). Of course, the existence of a bijection between $\mathcal{M}_1$ and $\mathcal{M}_2$ implies they have the same cardinality.

Consider now a domain $\mathcal{D}_m = (n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f, \mathcal{M}_1 = \{m_1\}, \mathcal{M}_2 = \{m_2\}, ..., \mathcal{M}_n = \{m_n\})$. Such a domain is called **strictly cooperative** if:

$$\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2, ..., \forall b_n, b_n' \in \mathcal{B}_n :$$
$$m_1(b_1, b_2, ..., b_n) < m_1(b_1', b_2', ..., b_n') \Leftrightarrow m_2(b_1, b_2, ..., b_n) < m_2(b_1', b_2', ..., b_n') \Leftrightarrow ... \Leftrightarrow$$
$$m_n(b_1, b_2, ..., b_n) < m_n(b_1', b_2', ..., b_n')$$
$$\wedge$$
$$m_1(b_1, b_2, ..., b_n) = m_1(b_1', b_2', ..., b_n') \Leftrightarrow m_2(b_1, b_2, ..., b_n) = m_2(b_1', b_2', ..., b_n') \Leftrightarrow ... \Leftrightarrow$$
$$m_n(b_1, b_2, ..., b_n) = m_n(b_1', b_2', ..., b_n')$$

An example of such a domain is one for which:

$$\forall b_1, b_1' \in \mathcal{B}_1, \forall b_2, b_2' \in \mathcal{B}_2, ..., \forall b_n, b_n' \in \mathcal{B}_n :$$
$$m_1(b_1, b_2, ..., b_n) = m_2(b_1, b_2, ..., b_n) = m_n(b_1, b_2, ..., b_n)$$

This is the equivalent of what in game theory is called a **variable-sum game with symmetric payoffs**.[2] The term payoff has occasionally been used in CoEC as a synonym for metric of behavior.

Just as for the competitive case, there are two alternative (and equivalent) definitions (one using $<$ and $=$, and another using $<$ and $>$). Also, weak cooperation and extension to more than one metric per metric set can be defined in a manner similar to competition.

Now that cooperative domains have been defined, it should be apparent why it is difficult to define competition for more than two roles. As in life, "the enemy of my enemy is my friend".

[2]Note that symmetric payoffs and symmetric roles, as discussed in section 2.2.1, are two different concepts.

It should also be apparent that there are domains that are neither weakly/strongly competitive, nor weakly/strongly cooperative (even when there are just two roles).

Role symmetry is orthogonal to this clustering as well. The main four combinations are:

- cooperative with symmetric roles: neural networks with variable number of nodes (Potter, 1997);
- cooperative with asymmetric roles: multi-parameter function optimization (Potter and De Jong, 1994; Potter, 1997; Wiegand, 2004);
- competitive with symmetric roles: mainly games (Rosin and Belew, 1995; Angeline and Pollack, 1993; Reynolds, 1994; Watson and Pollack, 2001; Funes and Pollack, 2000; Miller, 1996; Axelrod, 1989) and robot simulations (Stanley and Miikkulainen, 2002; Sims, 1994);
- competitive with asymmetric roles: sorting networks and input sequences (Hillis, 1990), cellular automata and initial conditions (Juillé and Pollack, 1998; Pagie and Hogeweg, 2000; Pagie and Mitchell, 2002; Williams and Mitchell, 2005), pursuit and evasion behaviors (Cliff and Miller, 1994, 1995; Floreano and Nolfi, 1997; Wahde and Nordahl, 1998), finite two-player games given by trees (Koza, 1991).

The reader may notice that the domains listed as cooperative correspond to those listed for compositional problems and domains listed as competitive correspond to those listed for test-based problems. This is because historically, compositional problems have only been defined in cooperative domains and test-based problems only in competitive domains. Note however that there is nothing in the definition of these clusterings requiring such a correspondence.

2.2.4 Solution Concepts

Neither of the two clustering criteria above is based on solution concepts. However, the opposite holds: solution concepts are dependent (to a certain degree) on the type of domain and the type of problem. This is natural, because a solution concept is basically a predicate over the space $\mathcal{S}$, that uses the metrics to decide its values. This section reviews currently known and used solution concepts. It also points out some other solution concepts that could be envisioned. The presentation is driven by the nature of the set $\mathcal{S}$. Some of these solution concepts have been described in (de Jong, 2005) with notations adjusted (simplified) either for two-role test-based problems or for multi-role compositional problems. Here, I define all of them using the general language defined in 2.2.1 and 2.2.2. For simplicity and without loss of generality, when $\mathcal{S}$ is defined over a single role, I will use role 1 rather than role i.

Test-based Problems With $\mathcal{S} = \mathcal{B}_1$

For these problems the domain has the form:

$$\mathcal{D}_m = (n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f, \mathcal{M}_1 \supseteq \{m_1\}, \mathcal{M}_2, ..., \mathcal{M}_n).$$

$\mathcal{C}$: maximum cumulative (or expected/average) utility As the name suggests, this solution concept denotes as solutions those elements of the search space $\mathcal{S}$ that have maximum cumulative utility over all elements of $\mathcal{S}$. Note that maximizing cumulative utility is equivalent to maximizing expected utility and also equivalent to maximizing average utility. I exemplify this solution concept using cumulative utility as defined below:

$$cumulUtil(b_1) = \sum_{(b_2,...,b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n} m_1(b_1, b_2, ..., b_n), \forall b_1 \in \mathcal{B}_1.$$

Then the maximum cumulative utility solution concept is defined as:

$$\mathcal{C} : \mathcal{B}_1 \to \{true, false\}, \forall b_1 \in \mathcal{B}_1 :$$
$$C(b_1) = \forall b_1' \in \mathcal{B}_1, cumulUtil(b_1) \geq cumulUtil(b_1')$$

Test-based problems that explicitly specify maximum expected utility as the solution concept are the previously mentioned sorting networks (Hillis, 1990) and cellular automata (Juillé and Pollack, 1998; Pagie and Hogeweg, 2000; Pagie and Mitchell, 2002; Williams and Mitchell, 2005). An algorithm targeted at this solution concept for the case of two roles has been devised by de Jong (2005) and called MaxSolve. It guarantees monotonic progress towards the solution concept if the space is finite. It has been tested in the context of number games.

For a variation of this solution concept and more than two roles, an algorithm has been proposed in (Schmitt, 2003b). Monotonic progress towards the solution concept is guaranteed if the solution set defined by it is not empty.

$\mathcal{C}$: **Pareto-optimality** The maximum utility solution concept defined above sums up the success of a behavior (for role 1) over all events that behavior can be part of and then compares different behaviors based on this sum. By contrast, the Pareto-optimality solution concept first compares different behaviors for role 1 based on their success in the same context (i.e. using events that only differ on role 1) and then aggregates over events. Here is the formal definition. Let *dominates* and *non-dominated* be two predicates:

$$\begin{gathered} \textit{dominates} : \mathcal{B}_1 \times \mathcal{B}_1 \rightarrow \{true, false\}, \forall b_1, b_1' \in \mathcal{B}_1 : \\ \textit{dominates}(b_1, b_1') = \forall (b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n, m_1(b_1, b_2, ..., b_n) \geq m_1(b_1', b_2, ..., b_n) \\ \wedge \exists (b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n, m_1(b_1, b_2, ..., b_n) > m_1(b_1', b_2, ..., b_n) \end{gathered}$$

Note that $\neg \textit{dominates}(b_1, b_1')$ does not imply $\textit{dominates}(b_1', b_1)$.

$$\begin{gathered} \textit{non-dominated} : \mathcal{B}_1 \rightarrow \{true, false\}, \forall b_1 \in \mathcal{B}_1 : \\ \textit{non-dominated}(b_1) = \nexists b_1' \in \mathcal{B}_1, \textit{dominates}(b_1', b_1) \end{gathered}$$

Then the Pareto-optimality solution concept is defined as:

$$\mathcal{C} : \mathcal{B}_1 \rightarrow \{true, false\}, \forall b_1 \in \mathcal{B}_1 : \mathcal{C}(b_1) = \textit{non-dominated}(b_1)$$

The set of all $b_1 \in \mathcal{B}_1 : \mathcal{C}(b_1) = true$ is called the Pareto-front. An element $b_1 \in \mathcal{B}_1 : \mathcal{C}(b_1) = true$ is said to be Pareto-optimal.

The Pareto-optimality solution concept is less restrictive than the maximum cumulative utility solution concept. This can be easily proven by showing that if an element $b_1 \in \mathcal{B}_1$ has maximum cumulative utility than it must be Pareto-optimal. The proof is by contradiction.

Let $b_1 \in \mathcal{B}_1$ such that $\forall b_1' \in \mathcal{B}_1, \textit{cumulUtil}(b_1) \geq \textit{cumulUtil}(b_1')$. Assume b_1 is not Pareto-optimal, i.e. $\exists b_1' \in \mathcal{B}_1, \textit{dominates}(b_1', b_1)$. This means

$$\begin{gathered} \forall (b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n, m_1(b_1', b_2, ..., b_n) \geq m_1(b_1, b_2, ..., b_n) \\ \wedge \exists (b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n, m_1(b_1', b_2, ..., b_n) > m_1(b_1, b_2, ..., b_n). \end{gathered}$$

By summation, we get

$$\sum_{(b_2,...,b_n)\in\mathcal{B}_2\times...\times\mathcal{B}_n} m_1(b_1', b_2, ..., b_n) > \sum_{(b_2,...,b_n)\in\mathcal{B}_2\times...\times\mathcal{B}_n} m_1(b_1, b_2, ..., b_n),$$

which is equivalent to $cumulUtil(b_1') > cumulUtil(b_1)$. This contradicts the fact that b_1 has maximum cumulative utility, thus the assumption that b_1 is not Pareto-optimal must be false, q.e.d.

This property of being less restrictive can be a caveat for the Pareto-optimality solution concept. In particular, the Pareto-front may be very large or infinite and thus less interesting for practical purposes.

$\mathcal{C}$: simultaneous maximization of all outcomes This solution concept also compares different behaviors for role 1 based on their success in the same context, but it is more restrictive than the maximum cumulative utility solution concept. A behavior b_1 for role 1 is considered a solution if for any (fixed) context provided by roles $2, ..., n$, b_1 is more successful in that context than any other behavior for role 1. Formally, this is described by:

$$\mathcal{C} : \mathcal{B}_1 \rightarrow \{true, false\}, \forall b_1 \in \mathcal{B}_1 :$$
$$C(b_1) = \forall b_1' \in \mathcal{B}_1, \forall (b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n, m_1(b_1, b_2, ..., b_n) \geq m_1(b_1', b_2, ..., b_n)$$

Here is the proof that simultaneous maximization of all outcomes is more restrictive than maximum cumulative utility. Let $b_1 \in \mathcal{B}_1$ be a solution for the simultaneous maximization of all outcomes solution concept, i.e. $\forall b_1' \in \mathcal{B}_1, \forall (b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n, m_1(b_1, b_2, ..., b_n) \geq m_1(b_1', b_2, ..., b_n)$. Summing over all $(b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n$, we get:

$$\sum_{(b_2,...,b_n)\in\mathcal{B}_2\times...\times\mathcal{B}_n} m_1(b_1, b_2, ..., b_n) \geq \sum_{(b_2,...,b_n)\in\mathcal{B}_2\times...\times\mathcal{B}_n} m_1(b_1', b_2, ..., b_n),$$

which is equivalent to $\forall b_1' \in \mathcal{B}_1, cumulUtil(b_1) \geq cumulUtil(b_1')$. Thus b_1 is also a solution for the maximum cumulative utility solution concept, q.e.d.

The caveat of the simultaneous maximization of all outcomes solution concept is that in some domains it may be too restrictive (e.g. describe the empty set). An algorithm targeted at this solution concept for the case of two roles has been developed by Rosin (1997) and called the Covering Competitive Algorithm. Monotonic progress towards the solution concept is guaranteed if the solution set defined by it is not empty.

Test-based Problems With $\mathcal{S} = \wp(\mathcal{B}_1)$

The domain for these problems has the same form as before, namely:

$$\mathcal{D}_m = (n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f, \mathcal{M}_1 \supseteq \{m_1\}, \mathcal{M}_2, ..., \mathcal{M}_n).$$

For each solution concept $\mathcal{C} : \mathcal{S} \to \{true, false\}$, a corresponding solution concept $all(\mathcal{C}) = \mathcal{C}'$ can be defined as follows:

$$\mathcal{C}' : \wp(\mathcal{S}) \to \{true, false\}, \forall \text{ S} \in \wp(\mathcal{S}) \, (i.e. \text{ S} \subseteq \mathcal{S}) :$$
$$\mathcal{C}'(\text{S}) = \forall s \in \text{S}, \mathcal{C}(s) \wedge \forall s' \in \mathcal{S} \setminus \text{S}, \neg\mathcal{C}(s')$$

In other words, a solution for the current problem is the set of *all* solutions to our previous problem. Clearly, for the solution concept $\mathcal{C}'$ there exists a single solution.

Solving a problem with $\mathcal{S} = \wp(\mathcal{S}_0)$ and solution concept $\mathcal{C}' = all(\mathcal{C})$ also solves the corresponding problem with $\mathcal{S} = \mathcal{S}_0$ and solution concept $\mathcal{C}$. In general, one would expect that solving the latter problem should be easier than solving the former, yet there are also cases when both problems are in the same big-O complexity class.

$\mathcal{C}$: Pareto-optimal equivalence set This is an example of a solution concept that has $\mathcal{S} = \wp(\mathcal{B}_1)$, yet is not of type $all(\mathcal{C}_0)$, for some solution concept $\mathcal{C}_0 : \mathcal{B}_1 \to \{true, false\}$. Instead, this $\mathcal{C} : \wp(\mathcal{B}_1) \to \{true, false\}$ is defined as follows:

$$\forall \mathcal{B} \in \wp(\mathcal{B}_1) \, (i.e. \mathcal{B} \subseteq \mathcal{B}_1) :$$
$$\mathcal{C}(\mathcal{B}) = \forall b_1 \in \mathcal{B}_1, \textit{non-dominated}(b_1) \Rightarrow$$
$$\exists b_1' \in \mathcal{B}, \forall (b_2, ..., b_n) \in \mathcal{B}_2 \times ... \times \mathcal{B}_n, m_1(b_1, b_2, ..., b_n) = m_1(b_1', b_2, ..., b_n)$$

For $n = 2$ and $m_1(b_1, b_2) \in \{0, 1\}, \forall (b_1, b_2) \in \mathcal{B}_1 \times \mathcal{B}_2$, this is the solution concept S_4 described in (de Jong, 2005).

With the Pareto-optimal equivalence set solution concept, one is not looking for the whole Pareto-front, but for any subset of the front that captures all distinctions between behaviors in terms of success (as measured by m_1). There may be multiple such subsets. In particular, one could also define a Pareto-optimal minimal equivalence set, by replacing $\exists b_1'$ with $\exists! b_1'$ (exists unique).

de Jong (2004a) introduced an algorithm called IPCA (Incremental Pareto Coevolution Archive) that guarantees monotonic progress towards the Pareto-optimal equivalence set solution concept, if given infinite memory. de Jong (2004c) also proposed LAPCA (LAyered Pareto Coevolution Archive) as an algorithm that makes reliable progress while using a limited memory.

While all solution concepts described so far have been used primarily in competitive domains, clearly there is nothing in their definitions to require that. In fact, the definitions use a single metric m_1 for the role on which $\mathcal{S}$ is based, while the distinction between cooperative and competitive domains requires the existence of at least two non-empty metric sets. Also, all these solution concepts have been used mainly in domains with only two roles.

Note also that one could apply the *all* operation as described above and obtain new solution concepts applicable to $\wp(\wp(\mathcal{B}_1))$.

Compositional Problems With $\mathcal{S} = \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n$

For these problems the domain needs to specify at least one metric for each role, i.e.:

$$\mathcal{D}_m = (n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f, \mathcal{M}_1 \supseteq \{m_1\}, \mathcal{M}_2 \supseteq \{m_2\}, ..., \mathcal{M}_n \supseteq \{m_n\}),$$

and solutions will be tuples specifying a behavior for each role.

$\mathcal{C}$: ideal team This solution concept refers to such tuples as *teams* and requires maximization of team success, defined as the sum of the successes for the behaviors composing the team in the context of that team. Formally, let:

$$team(b_1, b_2, ..., b_n) = m_1(b_1, b_2, ..., b_n) + m_2(b_1, b_2, ..., b_n) + ... + m_n(b_1, b_2, ..., b_n),$$
$$\forall (b_1, b_2, ..., b_n) \in \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n.$$

Then the ideal team solution concept is defined as:

$$\mathcal{C} : \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n \rightarrow \{true, false\}, \forall (b_1, b_2, ..., b_n) \in \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n :$$
$$\mathcal{C}(b_1, b_2, ..., b_n) = \forall (b'_1, b'_2, ..., b'_n) \in \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n, team(b_1, b_2, ..., b_n) \geq team(b'_1, b'_2, ..., b'_n).$$

The ideal team solution concept has been used almost exclusively in cooperative domains, under the name of ideal collaboration. However, there is nothing in the definition to restrict it to such domains. Of course, problems featuring an ideal team solution concept and a constant sum domain aren't very interesting, as all elements of $\mathcal{S}$ are solutions.

This solution concept was believed to be particularly difficult to be achieved using coevolutionary algorithms (Wiegand, 2004), but Panait (2006) showed that a CoEA can be designed that converges to the ideal team solution concept if given infinite populations.

$\mathcal{C}$: pure Nash-equilibria This is a solution concept inspired from game theory (Osborne and Rubinstein, 1994; Hofbauer and Sigmund, 1998). By analogy, roles can be viewed as the equivalent of players and behaviors as pure strategies. A tuple of behaviors (one for each role) is considered a solution, if every behavior in the tuple is the "best response" (i.e. most successful behavior) for that role, given the context formed by the other behaviors in the tuple.

Formally, for any $i \in 1..n$ let *best-response*$_i$ be a function defined as follows:

$$\begin{gathered}
\textit{best-response}_i : \mathcal{B}_1 \times ... \times \mathcal{B}_{i-1} \times \mathcal{B}_{i+1} \times ...\mathcal{B}_n \rightarrow \wp(\mathcal{B}_i) \\
\forall (b_1, ..., b_{i-1}, b_{i+1}, ..., b_n) \in \mathcal{B}_1 \times ... \times \mathcal{B}_{i-1} \times \mathcal{B}_{i+1} \times ...\mathcal{B}_n : \\
\textit{best-response}_i(b_1, ..., b_{i-1}, b_{i+1}, ..., b_n) = \{b_i \in \mathcal{B}_i | \forall b'_i \in \mathcal{B}_i, m_i(b_1, ..., b_{i-1}, b_i, b_{i+1}, ..., b_n) \geq \\
m_i(b_1, ..., b_{i-1}, b'_i, b_{i+1}, ..., b_n)\}.
\end{gathered}$$

Then the pure Nash equilibria solution concept is defined as follows:

$$\begin{gathered}
\mathcal{C} : \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n \rightarrow \{true, false\}, \forall (b_1, b_2, ..., b_n) \in \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n : \\
\mathcal{C}(b_1, b_2, ..., b_n) = \forall i \in 1..n, b_i \in \textit{best-response}_i(b_1, ..., b_{i-1}, b_{i+1}, ..., b_n)
\end{gathered}$$

The solution concept corresponding to the game theoretic notion of mixed Nash-equilibria can also be defined, as described below, with $\mathcal{S}$ having a different form.

Compositional Problems With $\mathcal{S} = \triangle\mathcal{B}_1 \times \triangle\mathcal{B}_2 \times ... \times \triangle\mathcal{B}_n$

The domain is still of type:

$$\mathcal{D}_m = (n, \mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n, \mathcal{O}, f, \mathcal{M}_1 \supseteq \{m_1\}, \mathcal{M}_2 \supseteq \{m_2\}, ..., \mathcal{M}_n \supseteq \{m_n\}).$$

$\mathcal{C}$: mixed Nash-equilibria Remember that for a set X, $\triangle X$ is the set of all probability distributions over X, i.e. $\{p : X \to \mathbb{R} | \int_{x \in X} p(x)\,dx = 1\}$. Let *exp* be a function defined as follows:

$$exp : 1..n \times \triangle\mathcal{B}_1 \times ... \times \triangle\mathcal{B}_n \to \mathbb{R}$$
$$\forall i \in 1..n, \forall(p_1, p_2, ..., p_n) \in \triangle\mathcal{B}_1 \times ...\triangle\mathcal{B}_n :$$
$$exp(i, p_1, ..., p_n) = \sum_{(b_1,b_2,...,b_n) \in \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n} p_1(b_1)p_2(b_2)...p_n(b_n)m_i(b_1, b_2, ..., b_n).$$

Also, for any $i \in 1..n$ let *best-mixed-response*$_i$ be a function defined as:

$$\textit{best-mixed-response}_i : \triangle\mathcal{B}_1 \times ... \times \triangle\mathcal{B}_{i-1} \times \triangle\mathcal{B}_{i+1} \times ...\triangle\mathcal{B}_n \to \wp(\triangle\mathcal{B}_i)$$
$$\forall(p_1, ..., p_{i-1}, p_{i+1}, ..., p_n) \in \triangle\mathcal{B}_1 \times ... \times \triangle\mathcal{B}_{i-1} \times \triangle\mathcal{B}_{i+1} \times ...\triangle\mathcal{B}_n :$$
$$\textit{best-mixed-response}_i(p_1, ..., p_{i-1}, p_{i+1}, ..., p_n) = \{p_i \in \triangle\mathcal{B}_i | \forall p'_i \in \triangle\mathcal{B}_i,$$
$$exp(i, p_1, ..., p_{i-1}, p_i, p_{i+1}) \geq exp(i, p_1, ..., p_{i-1}, p'_i, p_{i+1})\}.$$

Then the mixed Nash equilibria solution concept is defined as follows:

$$\mathcal{C} : \triangle\mathcal{B}_1 \times \triangle\mathcal{B}_2 \times ... \times \triangle\mathcal{B}_n \to \{true, false\}$$
$$\forall(p_1, p_2, ..., p_n) \in \triangle\mathcal{B}_1 \times \triangle\mathcal{B}_2 \times ... \times \triangle\mathcal{B}_n :$$
$$\mathcal{C}(p_1, p_2, ..., p_n) = \forall i \in 1..n, p_i \in \textit{best-mixed-response}_i(p_1, ..., p_{i-1}, p_{i+1}, ..., p_n).$$

The pure Nash equilibria solution concept can be expressed as a special case of the mixed Nash equilibria concept. Nash solution concepts have practical utility mainly in games and economics.

The Nash Memory algorithm described in (Ficici and Pollack, 2003; Ficici, 2004) is targeted at the Nash-equilibria solution concepts.

The domains used in this dissertation have two asymmetric roles. Chapters 3–6 are concerned with compositional problems having cooperative domains with variable-sum symmetric payoff, $\mathcal{S} = \mathcal{B}_1 \times \mathcal{B}_2$ and ideal team as the solution concept. Chapter 7 considers test-based problems having competitive domains with constant-sum payoff, $\mathcal{S} = \mathcal{B}_1$, and maximum cumulative utility as the solution concept.

2.3 Coevolutionary Algorithms

A coevolutionary algorithm is a specific type of evolutionary algorithm, therefore it will feature the same key components: representation, evaluation, selection and breeding. Generally, selection and breeding do not differ between CoEAs and regular EAs[3]. It is within representation and evaluation that lies the distinction (even if fuzzy at times). This is not surprising, since it is into the representation and evaluation components that the problem description is usually mapped[4] and CoEAs are only applicable to specific types of problems (namely ones for interactive domains).

This section is organized in two parts. The first describes what CoEAs are from a problem-mapping perspective and what decisions must be made during this phase. The second presents additional choices available for CoEA design. In both parts the presentation takes a hierarchical perspective.

While the part of the CoEC framework concerning problems took shape mostly over the past two years, the algorithms-part of the framework was already fairly well established starting with Wiegand's PhD dissertation (2004), that was the first to put together an extensive hierarchy of design choices (properties) available for coevolutionary algorithms. One major change since then has been the CoEC community's realization that the payoff quality is actually a property of domains and not algorithms. To this I have added my own realization that some of the design choices (the problem-mapping ones) take precedence over others, in the sense that they define the very nature of CoEAs. Figure 2.1 is based on figure 3.1 on page 36 of (Wiegand, 2004), but modified to reflect the above two aspects, and restructured and extended to cover new algorithms developed since then to the time of my writing.

The branches displayed with bold font denote the problem-mapping choices and will be discussed in the first subsection. The remainder are additional design choices described in the second subsection. Also note that some of the leaf nodes in the hierarchy are displayed in italics. These are "true" leaf nodes and constitute values for the property in their parent node. Leaves which are not italicized are properties that can further be expanded, but were

[3]CoEAs with tuple fitness (defined further on) use selection techniques from multi-objective EAs.
[4]Occasionally problem-specific knowledge is embedded into the breeding operators.

not (for space constraints and to keep the figure readable). Their expansion is discussed in the text.

Before moving on to the review, note that with any algorithm property (parameter) there is a notion of temporality attached, denoting how a value for that property is chosen. There are three possibilities:

- static: the value is chosen prior to running the algorithm and it remains fixed throughout run-time;
- dynamic: the value varies during run-time, as a function of time;
- adaptive: the value varies during run-time, as a function of the internal state of the algorithm (which in addition to time could include, for example, population diversity, operator success, etc.).

Clearly, the simplest algorithms are the ones for which all properties have static values. However, dynamic and adaptive schemes for setting parameters are believed to offer some advantages and were investigated for traditional EAs (Eiben et al., 1999; Angeline, 1995). The issue has resurfaced for CoEAs, and some properties lend themselves to dynamic / adaptive schemes more than others. Notes of this will be made when appropriate.

2.3.1 Problem to Algorithm Mapping

How does one go about using an EA to solve problems in interactive domains? Section 1.2.1 illustrated through an example a number of different alternatives, the most promising one being CoEAs. Here I give a more detailed description of what CoEAs are and how they approach such problems.

Regular EAs map elements of the problem search space $\mathcal{S}$ into individuals. CoEAs map behaviors into individuals. Behaviors may or may not be elements of $\mathcal{S}$, since $\mathcal{S}$ is defined as an aggregation over all, some or just one of the behavior sets. Thus, in general, a CoEA is searching for elements of $\mathcal{S}$ by actually operating in other spaces. Even when $\mathcal{S} = \mathcal{B}_i$ for some role i, in order to decide if an element of $\mathcal{S}$ is a solution, the algorithm must explore the behavior spaces corresponding to the other roles.

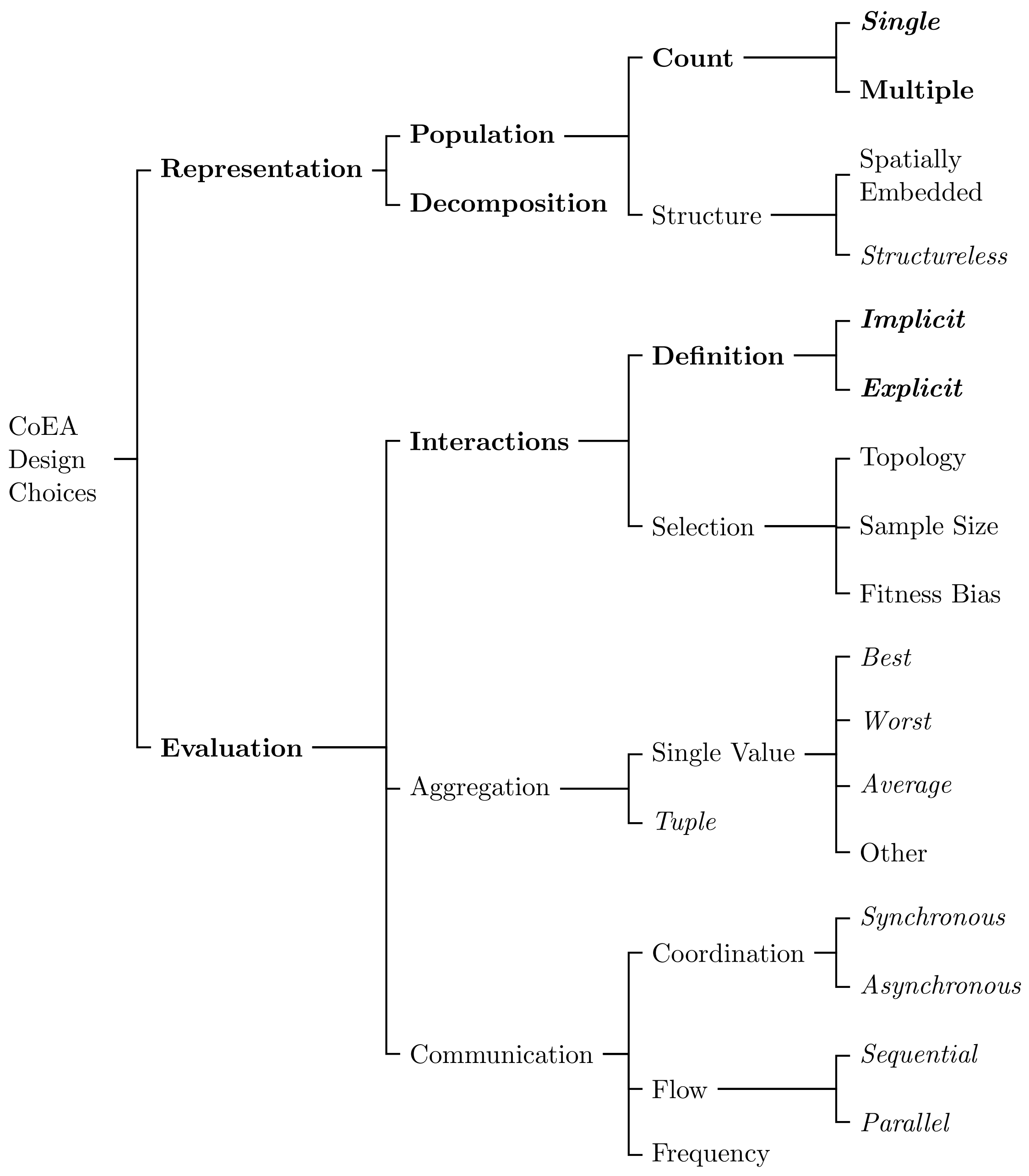


Figure 2.1: Hierarchical categorization of CoEA properties. Based on figure 3.1 on page 36 of (Wiegand, 2004), but restructured and extended. See text on previous page for font coding.

Consider first the case of domains with asymmetric roles only. The behavior sets corresponding to any two such roles will be different, therefore the algorithm will have to maintain different types of individuals for encoding the corresponding behaviors. CoEAs achieve this by keeping **multiple populations**, one for each asymmetric role[5]. Individuals in a population encode behaviors from the behavior set of the role corresponding to that population.

Remember that behaviors only acquire meaning in the context of events, i.e. behavior tuples in which each position corresponds to a role and thus contains a behavior from the behavior set of that role. Similarly, individuals can only be evaluated in the context of **interactions** with other individuals. An interaction is a tuple of individuals in which each position corresponds to a role and thus contains an individual encoding a behavior from the behavior set of that role. What this translates into for asymmetric roles is that an interaction is a tuple of individuals, one from each population, properly placed in the position corresponding to their population's role.

Usually, the cost unit for regular EAs is the evaluation of one individual[6]. The cost unit for CoEAs is the assessment of one interaction and evaluating one individual usually requires more than one interaction. The fitness an individual receives is some aggregate over multiple interactions of values returned by the domain's metrics when applied to those interactions. Various techniques for choosing interactions and aggregating metric values will be discussed later in section 2.3.2. Once fitness is computed, selection and breeding act independently in each population.

Some domains with asymmetric roles are the result of reformulating non-interactive domains as interactive ones. This reformulation occurs by partitioning the single behavior set $\mathcal{B}$ into $\mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n$. Clearly, one has many different choices for performing such a **decomposition** (e.g. whether all $\mathcal{B}_i$-s have the same size or not). Choosing a particular decomposition can be seen as an algorithm design choice, since it determines the number of populations and what their individuals encode. A few such partitioning choices were analyzed in (Wiegand et al., 2002; Wiegand, 2004). So far in the literature the decomposition

[5]In fact, at least one population per role is required, but more than one population per role may be kept, although this is uncommon in the literature.

[6]An example where this is not the case is genetic programming, where the evaluation of a tree depends on its size.

decision has been made statically.

In the case of domains where all roles are symmetric, there is a single behavior set. Thus the algorithm will need to maintain only a single type of individuals. The designer then has a choice of whether to still manage **multiple populations**, one per role (yet all populations will explore the same behavior set) or to have a **single population**.

With the former choice, interactions can still be seen as tuples from the cartesian product of populations. However, due to symmetry, all populations explore the same space and order does not matter, therefore interactions can also be seen as multisets over the behavior set with the additional constraint that each population must contribute exactly one individual to the multiset. In other words: if there are n symmetric roles sharing the same behavior set $\mathcal{B}$, and $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$ are the current populations $(\forall i = 1..n, \mathcal{P}_i \subset \mathcal{B})$[7], then interactions take the form $\ll b_1, b_2, ..., b_n \gg \in \mathcal{P}_1 \times \mathcal{P}_2 \times ... \times \mathcal{P}_n$, where $\ll\gg$ denotes a multiset. With the latter choice (i.e using a single population) interactions are multisets of size n over the current population.

Examples of domains with two symmetric roles approached with a single population can be found in (Angeline and Pollack, 1993; Reynolds, 1994; Miller, 1996; Axelrod, 1989), while ones approached with two populations can be found in (Rosin and Belew, 1995; Funes and Pollack, 2000; Stanley and Miikkulainen, 2002). Watson and Pollack (2001) and Sims (1994) investigated both approaches.

Perhaps the most controversial issue in CoEC is whether or not an algorithm choosing to use a single population for multiple symmetric roles should be called a CoEA. There are two reasons for this controversy. The first is that in biology, by definition, coevolution involves multiple species (Ridley, 1993; Thompson, 1994; Price, 1998). The second reason is that it has proven very difficult to clearly distinguish the above described single-population algorithms from some traditional single-population EAs. The EAs in question are ones using mechanisms such as fitness sharing (Goldberg and Richardson, 1987; Spears, 1994) or crowding (De Jong, 1975), where fitness of individuals becomes dependent on the contents of

[7]To be more accurate, this should read $\mathcal{P}_i \subset encoding(\mathcal{B})$.

the population, and thus one can claim that there is *some interaction* between individuals. Should we call the new single-population algorithms still simply EAs or should we call them CoEAs but also re-label some old single-population EAs as CoEAs?

This dissertation is only concerned with the analysis of two-population CoEAs, thus fully elucidating the matter is beyond the scope of my presentation. I refer the interested reader to (Luke and Wiegand, 2003) and sections 2.2.1 and 3.1.1 (subsection "Methods of Fitness Assignment") of (Wiegand, 2004). However, for the sake of clear terminology in the remainder of this background chapter, I will refer to the new algorithms as single-population CoEAs. Additionally, I offer the following criteria for distinguishing between them and the old specialized EAs: **interactions' definition**. For single-population CoEAs, there is a unit of interaction **explicitly** defined and determined by the problem specification and multiple such interactions are assessed during each evaluation round. The problem definition specifies the cardinality of each interaction and metrics of success for individuals participating in the interaction. For the old specialized EAs, the interaction is not inherent in the problem specification, but introduced into the algorithm as a feature believed to offer some advantages. There is no unit of interaction, instead the interaction usually occurs between all individuals and it is **implicitly** defined in the way fitness is assigned.

In terms of temporality, the choice between using a single- or a multi-population CoEA is a static one. Potter (1997) introduced a class of multi-population CoEAs in which the number of populations is dynamically or adaptively adjusted. These are applicable to compositional problems defined in a specific type of domains, namely those with an infinite(variable) number of roles, all of which are symmetric. As previously described, all populations explore the same behavior set and an interaction is a multiset with one element from each population. Populations are added or removed at run-time, usually according to some improvement criteria (i.e. adaptively). This can be done because of the variability of the number of roles. If $\mathcal{B}$ is the behavior set and k is the number of populations currently maintained by the algorithm, then $\mu^k(\mathcal{B})$ is currently explored. As k is varied over time, $\mu(\mathcal{B})$ is explored.

All domains analyzed in this dissertation have two asymmetric roles and are thus approached with two-population CoEAs.

2.3.2 Additional Design Choices

Representation

Most representation-related properties have been discussed in the problem-mapping section. Here I only add one more, which is population **structure**. Most evolutionary algorithms maintain populations simply as sets[8], without any additional structure to them (**structureless**). EAs where there is a certain topology associated with a population are called **spatially embedded** or spatially distributed (Sarma, 1998) (or also fine grained) and are believed to be useful for counteracting premature convergence. While population structure is not a CoEA-specific algorithm property, it is applicable to CoEAs and it is mentioned in the present hierarchy for two reasons. The first is that choosing a spatial embedding requires additional design decisions to be made concerning interactions between individuals, as mentioned in the following subsection. The second reason is that spatially-embedded CoEAs have shown early success in the work of Hillis (1990) and were subsequently used in a considerable number of papers (Cliff and Miller, 1994, 1995; Pagie and Hogeweg, 2000; Pagie and Mitchell, 2002), prompting analysis of their properties (Wiegand and Sarma, 2004; Williams and Mitchell, 2005).

All CoEAs in this dissertation are structureless.

Evaluation

Some issues concerning evaluation are pertinent to both single- and multi-population CoEAs. Regardless of whether interactions occur between individuals in the same population or in different populations, decisions need to be made as to what *interactions* should be assessed and how the outcomes of those interactions should be *aggregated* to give individuals fitness. When using multiple populations, the additional issue of *communication* between these populations arises. Each of these three matters will be discussed in turn.

[8]Unless a mechanism for preventing duplicates is implemented, populations are actually multisets.

Interactions The definition of interactions was discussed in the previous section. Here I concentrate on the **selection** of interactions (also referred to as the *interaction method* or, in cooperative domains, *collaboration method*). The simplest choice is to assess all interactions possible given the individuals present in the system at evaluation time. This has been referred to as full mixing or complete mixing. While no additional decisions would have to be made, this choice has the disadvantage of being very expensive, as the time cost of the algorithm is counted in interactions assessed. To reduce cost, one must assess only some of all possible interactions. This immediately raises the question "which ones?".

There are two main approaches to choosing a subset of interactions: individual-centric and population-centric. With the individual-centric approach, one individual at a time is considered, a set of interactions is chosen for that individual to take part in, and after the interactions are assessed, the individual's fitness is computed. These interactions may be reused (but generally are not) for computing the fitness of the other individuals that took part in them. These other individuals are called *opponents* in competitive domains and *collaborators* in cooperative domains. In this context, the phrase **sample size** denotes the number of interactions used per fitness evaluation. Note that if there is no reuse, the number of interactions an individual takes part in may be greater than the number of interactions used for its evaluation. Thus, while the sample size may be (and usually is) the same for all individuals, the number of interactions that individuals are involved in may vary.

A simple and quite common approach is to use a single interaction per fitness evaluation and have the "other" individuals in this interaction be the best from their respective populations. In cooperative domains this has been termed *single-best collaboration method* by Wiegand (2004), while in competitive domains it is referred to as *last elite opponent (LEO) evaluation* (Sims, 1994).

With the population-centric approach, a **topology** of interactions is picked such as to involve any individual in the population(s) at least once, these interactions are assessed and then fitness is computed for all individuals. Tournaments, such as single-elimination or round-robin, are the most common examples for single-population algorithms. Both for single- and for multi-population models, shuffle-and-pair (also called bipartite pairing) is a population-centric method that simplifies the reuse of assessed interactions (meant to reduce computational time). With the single-elimination tournament, different individuals will participate in a different number of interactions.

Information available from previous evaluation rounds may be used to influence how interactions are chosen for the current evaluation round. I call this **fitness bias**. Usually, individual-centric approaches use such biases, while population-centric ones do not. Clearly, to be able to perform such biasing, the algorithm must store some information about previous evaluations. This information usually consists of individuals and their fitness, but can also be more complex in nature (e.g. include other properties of individuals or relationships between individuals). The amount of information can range from remembering the last best individual to remembering all interaction assessments ever performed. With generational CoEAs, intermediary approaches include saving one or a few (usually the best) individuals from each previous generation (the Hall of Fame method introduced by Rosin and Belew (1997)) or saving all individuals in the previous generation. When the information kept spans more than just the previous generation, it is called a *memory* or *archive* (Panait et al., 2006; de Jong, 2004a,c; Ficici and Pollack, 2003; Oliehoek et al., 2006). Chapter 4 will analyze and compare some biasing mechanisms based on information from the previous generation.

Additional reviews and comparisons of various methods for selecting interactions can be found in (Angeline and Pollack, 1993; Panait and Luke, 2002; Sims, 1994).

In terms of temporality, population-centric interaction methods are usually static, as is the choice between population-centric and individual-centric. With individual-centric methods, additional choices (in particular the sample size) may be static – as is the case in chapter 4, but also dynamic (Panait and Luke, 2005; Panait et al., 2006) or adaptive (Panait and Luke, 2006; Panait et al., 2006; Panait, 2006).

Additional choices must be made when using spatially-embeded CoEAs. These tend to use individual-centric approaches and interactions are localized in space, namely neighborhood-based. The size of the neighborhood corresponds to the sample size, but the shape is an additional decision to be made. Fitness-biased selection of interactions can still be used (generally within the neighborhood).

Aggregation After an interaction is assessed and its outcome determined, metrics corresponding to the various roles can be computed for that outcome. Thus any individual that participated in the interaction can obtain a value from that interaction. In this context, the previously mentioned notion of reusing interactions denotes the fact that more than one (usually all) individuals involved in the interaction obtain a value from it. No reuse means only one individual (the one currently being evaluated) obtains a value from the interaction.

If an individual participates in a single interaction, then it must obtain a value from it, and that value becomes the individual's fitness. It may also be the case that an individual participates in more interactions, but the interaction method used is such that the individual obtains a value only from one of those interactions. Its fitness will then also be equal to that value. But when an individual participates in multiple interactions and obtains values from all (or more than one), then a choice must be made on how to aggregate these values.

One approach is to input values from multiple interactions into some computations and output a **single value** per individual (to be used as fitness). The computations can simply use all values obtained by the individual and determine the **best**, the **worst** or the **average** of those values. A comparison of these three methods can be found in (Wiegand et al., 2001). **Other**, more complex computations can take into account values from other individuals as well, as is the case for competitive fitness sharing (Rosin and Belew, 1995, 1997) or the biasing technique introduced by (Panait et al., 2003, 2004a). The advantage of the single-value approach is that traditional parent-selection methods can then be used by the algorithm based on single-valued fitness. The disadvantage is that the different ways of computing a single value have different (and strong) biases, and it is not always straightforward to tell which bias is more helpful (or less harmful) for the solution concept at hand. In particular, when dealing with the ideal team solution concept, the biases of the averaging method proved harmful (Wiegand, 2004) while choosing the best is helpful (Panait, 2006).

The alternative is to have fitness be a **tuple** of values, usually the values obtained by the individual from multiple interactions. Selecting parents based on such fitnesses requires more specialized methods, in particular ones akin to multi-objective EAs. The tuples of different individuals of the same role must be somehow comparable, which imposes constraints on the interactions that generated the values in the tuples. The advantage of this approach is that its bias may be more appropriate for solution concepts such as Pareto optimality or simultaneous maximization of all outcomes. Algorithms using tuple fitness are most common in competitive domains (de Jong, 2004a,c; Ficici and Pollack, 2001). An example using tuple fitness in cooperative domains can be found in (Bucci and Pollack, 2005).

The CoEAs I analyze in the following chapters all use best-single-value aggregation.

Communication When using a CoEA with multiple populations, in order for individuals in one population to interact with individuals from other populations, the populations must have access to one another's contents. This is an issue of communication, and thus entails choices typical of any distributed system, such as coordination, flow and frequency.

In terms of **coordination**, the communication can be synchronous or asynchronous. In the **asynchronous** case, the populations evolve at their own pace and communicate with one another through shared memory. They decide independently when to write new information about their state to the shared memory and when to check the memory for new information about the other populations (which may or may not be available). Asynchronous CoEAs are uncommon. In the **synchronous** case, there is a centralized clock that dictates when the populations exchange information.

In terms of **flow**, the asynchronous model is always **parallel**, in the sense that at any point in time there may be more than one population running[9]. The synchronous model can be either parallel or sequential (also called serial). In the **parallel** case, all populations run simultaneously for a certain period of time (dictated by the central clock), after which they all pause and exchange (communicate) information and then they all continue. In the **sequential** case, at any point in time there is a single population running and populations take turns in a round-robin fashion. In Wiegand (2004)'s hierarchy of CoEA properties, flow is called *update timing*. I feel the new terminology can reach a larger audience, by speaking in terms of communication. An analysis and comparison of synchronous parallel and sequential two-population CoEAs will be presented in chapter 4.

[9]When a parallel CoEA is run on a single processor, this translates into the fact that there is no guarantee about the order in which the populations run.

Frequency refers to the number of evaluation-selection-breeding cycles that a population goes through between two communication events. The frequency may be uniform across all populations or it may differ from one population to another. Chapter 4 analyzes the effects of the communication frequency in a synchronous sequential two-population CoEA.

2.4 Coevolutionary Dynamics

As described so far, coevolutionary algorithms seem a natural, highly adaptive search method for certain kinds of problems. Unfortunately, CoEAs have often disappointed engineers with poor performance and / or counterintuitive behavior. As the first example in the following chapter will show, even modifying the algorithms often leads to counterintuitive responses on certain problems. In order to use CoEAs successfully, one needs to understand why these phenomena happen, and the best way to gain such understanding is to analyze the time-dependent properties of coevolutionary systems, commonly referred to as their *dynamics*.

This section has three parts. The first discusses the reason *why* CoEAs exhibit complex behaviors to begin with, namely fitness relativity. The second discuses *how* to monitor the behaviors and the third summarizes *what* behaviors are currently known to occur.

2.4.1 Relativity in CoEC

As presented in the previous two sections, there is a wide variety of CoEC approaches. What all of them have in common is the fact that fitness is relative. It is in this feature that lie both the promises and the caveats of CoEC. Section 1.2.1 focused on the promises: providing a dynamic gradient, helping maintain diversity and incrementally building complexity. In this section, the focus is on the caveats.

The metaphor most commonly used to describe relativity in coevolution is the *Red Queen*. The Red Queen is a character in Lewis Carroll's novel "Through the Looking Glass" that perpetually runs without getting very far because the landscape moves with her. The Red Queen became a metaphor first in biology and subsequently in CoEC. There are multiple ideas that can be extracted from the scene in the novel, and therefore different people use the Red Queen metaphor itself to express different things.

According to Paredis (1997), "biologists use the term Red Queen hypothesis for this phenomenon in which constant change is needed to survive". Ridley (1993) writes that in biology the Red Queen is used to denote the "concept that all progress is relative".

In CoEC, some researchers use the phrase "Red Queen dynamics", but even so they mean different things by it. From (Pagie and Hogeweg, 2000): "two species can show a continued evolutionary change which can be oscillatory in nature or which is best described as a runaway process. The ... outcome ... is often referred to as 'Red Queen dynamics' or an 'arms race'[10]". From (de Jong and Pollack, 2004): "... continuously change without making overall progress, possibly resulting in cyclic behavior. This phenomenon is known as Red Queen dynamics ... or mediocre stable states".

Other researchers talk about a "Red Queen effect", but again there is no agreement as to what this means. Wiegand (2004) considers it to be a "diagnostic problem that occurs when populations seem to be changing, but the internal subjective measure shows no progress is occurring. This phenomena may describe stagnation or an arms race". Watson and Pollack (2001) give a similar definition: "This is the Red Queen effect ... though the performance of individuals improves, the performance of their opponents improves at the same time and they find themselves no better off (subjectively)". Cliff and Miller (1995) have a much broader view of the term: "... the Red Queen effect: the fitness landscape of one population is affected by the current strategies of any opponent populations; and the movements of one population over a fitness landscape can significantly alter the fitness landscapes of the other populations".

Regardless of how one uses the Red Queen metaphor, the important issue to be acknowledged is that in CoEC fitness is relative. This generates two problems, discussed in the following two subsections. The first problem is that comparisons between fitness values at different evolutionary times are meaningless. Fitness can no longer be used to monitor progress towards the goal. Different metrics are needed and they are discussed in the following subsection. The second problem is that relativity can generate many intricate and often undesirable dynamics. These will be reviewed in 2.4.3.

[10]See subsection 2.4.3 for explanations of this term.

2.4.2 Instrumenting Coevolutionary Dynamics

As discussed in section 2.2, it is very important to define a goal (solution concept) for the problem to be solved with a CoEA. What is equally important is to have a way to measure whether at run-time the algorithm is making progress towards that goal. For CoEAs, this is particularly problematic due to the relativity issue discussed above.

The behavior of a coevolutionary system over time (its *dynamics*) can be observed from two perspectives: the genotypic / phenotypic makeups of individuals and their quality. Of course, to measure progress towards the goal, one needs to monitor quality. However, to understand how the algorithm functions, monitoring genotypic / phenotypic change can be very useful.

Both for quality and for genotypic makeup, one has a choice of whether to measure all individuals in the population(s) or only select ones (e.g. best). Measuring all individuals can provide more information, but it may make it difficult to extract the important patterns within and also pose a challenge for visualization. Additionally, one must beware that compressing information (e.g. by means of averaging) may actually eliminate important trends in the data.

For quality measures, the trends one usually looks for include: increase, decrease, stagnation, noise, repeated values. For genotypic change, trends include: cycling, convergence, divergence, chaos, etc.

Quality Metrics

I discuss quality measures first. In this respect, CoEC was plagued by confusion for a while, due to failure to properly acknowledge the importance of fitness relativity. Wiegand (2004) cleared matters by providing a clustering of metrics for coevolution based on two criteria: contextual dependence and influence on the algorithm. I reproduce this clustering here, as it is essential to this review.

"**Definition 1.** ***Objective measure*** *– A measurement of an individual is* ***objective*** *if the measure considers that individual independently from any other individuals, aside from scaling or normalization effects.*

Definition 2. ***Subjective measure*** *– A measurement of an individual is* ***subjective*** *if* *the measure is not objective.*

Definition 3. ***Internal measure*** *– A measurement of an individual is* ***internal*** *if the measure influences the course of evolution in some way.*

Definition 4. ***External measure*** *– A measurement of an individual is* ***external*** *if the measure cannot influence the course of evolution in any way.*" (Wiegand, 2004)

So, based on contextual dependence, metrics can be objective or subjective, and based on influence on the algorithm metrics can be internal or external. When using both criteria, four combinations can be formed. As used in EC (not biology), fitness is always an internal metric. For traditional EAs (not using special techniques such as crowding or fitness sharing), fitness is objective. For CoEAs, fitness is always subjective, as it depends on other evolving individuals.

Performance of an algorithm towards the problem solving goal should always be measured with an objective metric. For CoEAs, such a metric has to be external, since fitness, the internal quality metric, is subjective. In some cases, designers of CoEC systems have such a metric in mind to begin with (e.g. (Hillis, 1990; Watson and Pollack, 2001)), but for one reason or another (see 1.2.1) they do not use it as an internal objective metric. In other cases, when CoEAs are applied without a precise goal (e.g. modeling) or with a non-testable goal (often), an objective metric either does not exist or cannot be computed. Therefore, some external subjective metrics of performance are used to tell whether the algorithm is making *any kind* of progress. While such metrics tell us something about the behavior of the algorithm, and can be useful in this respect, they cannot tell us how the algorithm is doing with respect to the actual goal.

In the following, I review external quality metrics that have been used in the literature.

In the context of a game between string generators and string predictors, without a specified solution concept, Ficici and Pollack (1998) use an external objective metric to investigate the dynamics of a CoEA. The metric is domain-specific and uses notions from information theory, such as entropy and order.

Subjective external metrics tend to be more general and usually rely on the history of evolutionary runs. The first such metric was introduced by Cliff and Miller (1995) in the form of CIAO plots. The acronym stands for "current individual ancestor opponent".

For a two-population CoEA, a CIAO plot is a matrix in which rows represent generations of one population and columns represent generations of the other population. Every cell represents an interaction between the best individual in each population at the corresponding generations. Thus, individuals from later generations of one population interact with individuals from early generations of the other population (ancestors). The cells are color-coded on a gray scale, and can be constructed to reflect success from the perspective of either population.

Floreano and Nolfi's "master tournament" metric (1997) is basically a compression of the information in CIAO plots. Averages are taken along lines for one population and across columns for the other population.

Both these methods are computationally expensive, as they require the evaluation of n^2 interactions, where n is the number of generations. The master tournament metric makes it easier to identify "broadly-successful" individuals, but the averaging it performs may obscure some circularities which can be observed using the CIAO plots. An in-depth critique of CIAO plots is available in (Cartlidge and Bullock, 2004b).

While these metrics were introduced in the context of domains with two asymmetric roles, they are easily applicable to domains with two symmetric roles, whether approached by single-population or two-population CoEAs. Additionally, the techniques do not require imposing any constraints on the success metrics of the domain.

In the context of a domain with two symmetric roles and a two-population CoEA, Stanley and Miikkulainen (2002) introduced a less costly technique called "dominance tournament". At each generation, the best individual in each population is determined and the two of them are paired; out of their interaction, the more successful one is designated the generation champion. The dominance property is then defined as follows. The champion from the first generation is automatically considered dominant. In every subsequent generation, the champion is paired only with previous dominant champions and it is itself labeled dominant only if it is more successful than all of them. The method is easily applicable to domains with two symmetric roles and a single-population CoEA. The paper also suggests a (less straight forward) extension to some domains with two asymmetric roles.

CIAO, master and dominance tournament are all techniques that track only the best of generation individual. This was contrasted by Bader-Natal and Pollack (2004) that introduced a "population-differential" technique monitoring all individuals in each generation. Plots similar to the CIAO plots are produced, but now each cell is an aggregation over the results of all pair-wise interactions between individuals in the two populations at the generations corresponding to that cell. This method is clearly more expensive, therefore a number of memory policies are introduced for reducing time complexity.

All the subjective metrics described so far require performing additional evaluations, rather than using the ones already performed by the algorithm. Funes and Pollack (2000) introduced a technique that took the latter approach. Their metric, called "relative strength" is still subjective, as the value it returns for a particular individual depends on the current history of all interactions, and this history grows with time. The metric is based on paired comparison statistics and is applicable to domains with two symmetric roles.

Genotypic / Phenotypic Metrics

Cliff and Miller (1995) were also the first to use techniques for tracking genotypic changes in order to analyze the run-time behavior (dynamics) of CoEAs. They introduce "elite bitmaps", "ancestral Hamming plots" and "consensus distance plots" as tools complementary to CIAO plots. They all work on binary representations. Elite bitmaps simply display the genotype of best-of-generation individuals next to each other in temporal sequence. Some additional processing can reveal interesting patterns. Ancestral Hamming plots display the Hamming distance between elite individuals from different generations. Consensus distance plots monitor the genotypic make-up of the whole population through the distribution of Hamming distances from the genotype of each individual to the population's "consensus sequence". All three techniques are applicable to EAs in general, not just CoEAs.

Some techniques for tracking genotypic change have also been devised for models of the algorithms rather than the algorithms themselves. For a domain with two symmetric roles sharing a behavior set with two elements (the Hawk-Dove game), Ficici et al. (2000, 2005) track the state of a two-genotype infinite population as a percentage (e.g. the percentage of hawks in the population). "Cobweb" plots (Alligood et al., 1996) are produced to reflect how the population state changes under the influence of various selection mechanisms operating over frequency-dependent fitness. Ficici and Pollack (2000) further use the technique in a CoEA with a finite population, while Ficici (2006) extends the method to models with two genotypes and *two* infinite populations.

Similarly, Panait et al. (2004b) visualize basins of attraction for the population state for models with two infinite populations operating in domains with two symmetric roles sharing a behavior set of size two or three.

Wiegand (2004) uses two-population CoEAs to approach problems where the domain has two asymmetric roles and symmetric payoff, and the solution concept is ideal team. He also models the CoEA with infinite populations and discretizes the behavior sets to 8 elements each, then plots two-dimensional "take-over curves" analogue to those of Goldberg and Deb (1990) from traditional EC.

2.4.3 Known Dynamics

As previously mentioned, the relativity of fitness in CoEAs generated many phenomena that were not encountered in traditional EAs. Moreover, these phenomena were quite complex, thus difficult to understand and even to describe, causing once again terminology problems. However, equipped with the measuring techniques described above, one can do a better job at understanding and characterizing the dynamics of coevolutionary systems.

The one phrase that is probably as controversial as the Red Queen is ***arms race***. Outside of CoEC, the general use of the term denotes "any competition where there is no absolute goal, only the relative goal of staying ahead of the other competitors" (WKP). This is also the meaning it has for biologists, as they do not have any stake in objective measurement.

In CoEC, from an external perspective, there is (or there should be) an absolute goal, even if it may not be testable. And yet, especially in two-population, competitive payoff CoEC, the phrase arms race has been used to denote increase in the external objective metric in one population caused by and causing in return increase in this metric for the other population.

Unless otherwise noted, all other dynamics described in the following are considered pathological when using CoEC for optimization. They may however become useful in other types of applications of CoEC.

Cycling is commonly agreed upon to mean visiting the same areas of the space multiple times in a repeated sequence. This will generate some repeating patterns in the external quality metric. Conversely, when such patterns in the metric are noticed, they may be caused by cycling, but also by other dynamics, such as chaos. Thus, the only reliable way to trace cycling is by observing changes in individuals at the genotypic / phenotypic level. A number of different works have shown the occurrence of cycling, and initially it was presumed to be caused by intransitivities present in the problem's definition. Subsequent work (de Jong, 2004b) has shown that cycling can appear even in the absence of intransitivity.

Mediocre stable states (Angeline and Pollack, 1993) is a term whose description usually includes cycling. Wiegand (2004) explicitly specifies that: "...mediocre stable states ... here limiting behaviors (either fixed-point or cyclic) ... at particularly suboptimal points in the space, from some external perspective". Similarly, in (Ficici and Pollack, 1998), a situation where "the two populations ... fall into a circular pattern of convention chasing" is considered an example of a mediocre stable state, states more generally defined as those "where the agents in the evolving population(s) ... discover a way to collude to give the impression of competition without actually forcing each other to improve in any objective sense". The definition of Watson and Pollack (2001) implicitly allows for cycling: "a condition where the coevolutionary system is not producing improved performance in the objective metric despite continued adaptive steps in the subjective metric". What all these definitions have in common is that they refer to a lack of improvement in the external objective metric over time, when such improvement could actually be achieved in the search space. The term "mediocre" seems appropriate to describe this. On the other hand, the phrasing "stable state" seems less appropriate for all the situations have been included under it. Some of the definitions above (Ficici and Pollack, 1998; Watson and Pollack, 2001) actually *require* that some internal change accompany the external mediocrity. In recent years, the suggestion (agreed upon by many members of the community) is that this term be dismissed and replaced with other more specific terms.

Another dynamic is the ***loss / lack of gradient***, described as a case "in which one population comes to severely dominate the others, thus creating an impossible situation in which the other participants do not have enough information from which to learn" (Wiegand, 2004). As an example, for test-based CoEC, the tests in one population may be so tough that no learner in the other population can solve. In (Wiegand and Sarma, 2004), the same problem is presented for cooperative / compositional CoEC as a situation when "the diversity of a subset of the populations suddenly decreases, leaving the others searching only a static projection and not the full problem". More generally, gradient in CoEC has been defined (CoEC-Wiki) as follows: "Since the evaluation of individuals in coevolution is dependent on other evolving individuals (*evaluation individuals*), appropriate evaluation individuals must be present in order to determine the relative merit of different individuals. The information provided by evaluation individuals is called the gradient, as its role is analogous to that of the gradient in standard optimization problems".

Abstracting from these definitions, lack of gradient is a situation in which the distribution of fitness values in (at least) one population is (almost) flat. Loss of gradient is the event instating lack of gradient. Lack of genotypic / phenotypic diversity always generates a lack of fitness diversity, but the latter can occur even while genotypic / phenotypic diversity is still present, simply because fitness is dependent on the contents of the other population(s). Lack of gradient may be temporary (gradient may be regained due to changes in either population) or persistent. Lack of gradient will cause genetic drift (Watson and Pollack, 2001), which sometimes can actually lead to regaining gradient. Loss of gradient has also been referred to as *disengagement* and claimed to "generally lead to over-specialization" (de Jong and Pollack, 2004).

Overspecialization is a dynamic in itself and need not necessarily be the result of loss of gradient; it is the phenomenon that individuals improve on some of the underlying objectives but fail to do so on others (CoEC-Wiki). A classic example is that of players discovering an opponent's weaknesses and exploiting them but failing to learn the task in a general way (Watson and Pollack, 2001).

While most of the above dynamics have been identified and described in the context of competitive CoEC, one dynamic found to be problematic for cooperative CoEC is ***relative overgeneralization*** (Wiegand, 2004). It refers to the phenomenon that components that perform well in combination with a large number of other components are favored over components that are part of a global optimum (but do not yield successful partnerships in general). The equivalent of this behavior in competitive settings is actually often considered useful.

2.5 Coevolutionary Analysis

With so many things that can go wrong and only a few considered right, it is no wonder that CoEC as an optimization tool has been described as "sometimes frustratingly slow and baffling" (Cliff and Miller, 1996). Even being aware of all these pitfalls is a step forward and it is due to the research efforts invested in CoEC.

This section reviews work analyzing CoEC setups. It does so from the perspective of the framework described in the previous sections. Moreover, the focus is on pointing out issues that were not addressed and providing motivation for the particular type of analysis presented in this dissertation.

At the beginning of the field a lot of the CoEC work was "experimental" and domain-focused. Namely, the researcher was interested in solving a particular problem (usually optimization) and was trying out coevolutionary algorithms as a potential method. If the problem was previously approached using regular EC, the standard practice was to conduct a comparison between the two methods. Otherwise, the threshold to separate good results from bad ones was set in a domain-dependent way (occasionally arbitrary). In both cases, CoEAs' performance ranged from good to very bad. As the field began to mature, the focus shifted from the application domain to analyzing CoEAs as a tool. Two different approaches were taken: analyzing the actual algorithms or analyzing models of the algorithms.

2.5.1 Algorithm Analysis

The vast majority of analyses of algorithms were empirical in nature and can be further classified as either "black-box" analysis or dynamics analysis. Theoretical analysis of actual algorithms is much less common.

Empirical Black-box Analysis

With this approach, the CoEC system was viewed as a black box whose inputs were the algorithm and the problem, and the output was observed performance (e.g. quality of solution produced given a certain amount of time). The algorithm and / or the problem would then be varied and the output of the system re-observed, with the goal of determining the rules governing the dependency between inputs and outputs.

Two different approaches can be distinguished within the black-box analysis category. One approach focuses on *properties* of the algorithms, of the problems or of both. When algorithm properties were involved, this approach has been referred to as component analysis (Wiegand, 2004). The other approach varies the algorithm and / or the problem and compares the resulting performance without a clear isolation of the properties responsible for the differences in performance.

I discuss this latter approach first. Most works introducing new algorithms are of this type. This includes some of the early research comparing CoEAs with traditional EAs, such as (Hillis, 1990; Angeline and Pollack, 1993) for competitive test-based setups and (Potter and De Jong, 1994) for cooperative compositional setups. Later on, the approach was used to compare CoEAs amongst themselves, usually "enhanced" algorithms with basic ones. Examples include (Ficici and Pollack, 2001, 2003; Ficici, 2004; de Jong and Pollack, 2004; de Jong, 2004a,c, 2005) for competitive test-based setups and (Bucci and Pollack, 2005; Panait et al., 2006; Panait and Luke, 2006; Panait, 2006) for cooperative compositional setups. Some of the algorithms introduced by these works (de Jong, 2004a, 2005; de Jong and Pollack, 2004) are backed-up by theoretical results, as I will discuss in a coming subsection.

With the property-focused black-box approach, studies addressed either problem properties, algorithm properties or the interaction between them. I discuss these cases in order.

Some of the CoEA-versus-EA comparisons were extended to determine the classes of problems for which one type of algorithm would provide performance advantages over the other. Such work was performed for cooperative compositional setups by Potter (1997) and Wiegand and Potter (2006). The scrutinized problem properties were *cross-component interaction* (also called *separability* (Wiegand, 2004) or *inter-agent epistasis* (Bull, 1998)), *dimensionality*[11], *noise* and *relative sizes of basins of attraction of local optima.*

For competitive test-based setups, empirical black-box analysis was used to investigate problem properties such as *role asymmetry* (Olsson, 2001), *intransitivity* (de Jong, 2004b) and *dimensionality*[12] (de Jong and Bucci, 2006). These works introduced specialized algorithms intended to target the respective problem property.

Studies focusing only on algorithm properties investigated either the mechanism for *selecting interactions* (Panait et al., 2006; Panait and Luke, 2005; Parker and Blumenthal, 2003; Blumenthal and Parker, 2004) or the mechanism for *aggregating* the result of those interactions (Panait et al., 2003; Wiegand et al., 2001; Cartlidge and Bullock, 2002). All but the last cited work were concerned with cooperative compositional setups.

Finally and perhaps most importantly, some black-box empirical studies analyzed the effects on performance of the interdependency between algorithm properties and problem properties. The most studied algorithm property was the mechanism for *selecting interactions.* The majority of the work targeted cooperative compositional setups (Potter, 1997; Wiegand et al., 2001, 2002; Wiegand, 2004; Bull, 1997, 2001). The launched hypothesis was that the performance effects of the interaction method are tightly related to the (previously mentioned) problem property called *separability* and the type of *cross-population epistasis* that it translates into. This dependency however proved not to be as straightforward as thought, and I will elaborate more on this in chapter 4. For test-based competitive setups, two different methods for *selecting interactions* were analyzed by Panait and Luke (2002), suggesting that the amount of *noise* in the problem affects their influence on performance.

For cooperative compositional setups, Panait et al. (2004a) extended their previous work on the mechanism for *aggregating* results from multiple interactions by studying how its performance effects are affected by the problem's *local optima and their basins of attraction.* Wiegand and Sarma (2004) studied the potential benefits of *spatially-distributed schemes for selecting interactions and / or selecting parents* on problems with *role asymmetry.*

Bull (1998) studied the effects of *mutation* and *crossover* in the context of "mixed-payoff" domains (Kauffman and Johnson (1991)'s NKC landscapes) and found them to be sensitive to *inter-agent epistasis.*

[11]Here dimensionality refers to the number of roles.

[12]Here dimensionality refers to the number of underlying objectives implicitly defined by the test role(s).

Empirical Dynamics Analysis

While black-box analysis can provide some heuristics for improving performance, it cannot explain the causes for the observed dependencies between the inputs and the outputs of a CoEC setup. To understand these causes, one needs to observe the system while running and track some of its time-varying properties. In EC lingo, this has been called dynamics analysis. However, it did not go all the way to using the dynamics to explain the connection between algorithm and problem properties on one side and performance on the other side. Individual studies connected only a subset of the pieces; they analyzed either:

- dynamics in isolation (Cliff and Miller, 1995; Stanley and Miikkulainen, 2002; Bader-Natal and Pollack, 2004; Floreano and Nolfi, 1997; Axelrod, 1989; Cartlidge and Bullock, 2004b);
- the connection between problem properties and dynamics (Bull, 2005a);
- the connection between problem properties, algorithm properties and dynamics (Bull, 2005b; Kauffman and Johnson, 1991);
- the connection between dynamics and performance (Pagie and Mitchell, 2002; Juillé and Pollack, 1998; Ficici and Pollack, 1998; Miller, 1996; Funes and Pollack, 2000; Paredis, 1997);
- the connection between problem properties, dynamics and performance (Watson and Pollack, 2001);
- the connection between algorithm properties, dynamics and performance (Williams and Mitchell, 2005; Pagie and Hogeweg, 2000; Rosin and Belew, 1995, 1997; Cartlidge and Bullock, 2003, 2004a; Bucci and Pollack, 2003);

The research approach of the proposed dissertation is to tie all the pieces of the chain together, namely how the combination of algorithm properties and problem properties influences dynamics and how dynamics influence performance.

Theoretical Analysis

Theoretical analysis of actual algorithms is still in its early stages, as the mathematics involved can be very complex.

The most common results are those guaranteeing that if a solution exists, given enough time, the algorithm will find it. For coevolution, this is not a trivial result, given the possibility of dynamics such as cycling and the fact that random search, the baseline for comparison in traditional search problems, is no longer clearly defined for coevolutionary problems.[13]

Some of the studies mentioned in the empirical analysis subsection (de Jong, 2004a, 2005; de Jong and Pollack, 2004) include such theoretical guarantees for the new algorithms they introduce. These proofs hold under various (potentially idealistic) conditions such as unbounded memory or a finite search space. While the proofs are not problem-dependent, they do not provide bounds on the amount of time needed to find a solution, which is why comparisons between different algorithms are still performed empirically. The mathematical techniques used are mainly ones from algebra and set theory. All cited works dealt with various solution concepts for competitive test-based problems.

A stronger type of guarantees comes from the works of Schmitt (2003a,b). For a solution concept that is a variation of maximum cumulative utility and could be applied to both test-based and compositional problems, the proposed algorithm is guaranteed not only to find a solution, but to have its populations converge to containing only genotypes corresponding to the solution(s). Markov-chain analysis is used for the proofs. Bounds on convergence speed are not given.

The only studies giving such bounds come from Jansen and Wiegand (2003a,b) who performed asymptotic run time analysis of a 1+1 CoEA in the context of function optimization reformulated as a cooperative compositional problem. While insightful, the results are problem dependent and the problems used were carefully crafted pseudo-boolean functions. The math involved is also reliant on the nature of the problems and the simplicity of the CoEA.

Finally, the only theoretical study of the dynamics of actual algorithms comes from

[13]For coevolutionary problems, the space of potential solutions and the space of events (interactions) are usually different. The cost of the search is measured in interactions assessed, but an undetermined number of interactions may be needed to determine if a potential solution is a true solution.

Funes and Pujals (2005) who used probability theory to investigate the relationships between problem properties, algorithm properties and dynamics. Trajectories through the behavior sets were characterized in terms of their resemblance to random walks.

2.5.2 Model Analysis

To make the math more tractable, some researchers took the approach of studying models of the algorithms. These models consider various abstractions or simplifications, such as infinite population sizes, very small, discrete search spaces (e.g. 2x2 games) or lack of variational operators. The first phase of the analysis consists of deriving formulas that describe the change of the model state in one step. The behavior of the model over time can then be analyzed in one of two ways: 1) iterate the formulas using a computer and observe trajectories and attractors; or 2) mathematically derive properties of the time limits. Examples of such works include (Chang et al., 2004; Bergstrom and Lachmann, 2003; Ficici, 2004, 2006; Wiegand, 2004; Panait, 2006).

The question that always arises with model analyses is how well the model predicts the behavior of the actual algorithm, since the algorithm violates one or more of the assumptions of the model. Therefore, some of the model studies are accompanied by empirical results from running the actual algorithm. However, Liekens et al. (2004); Liekens (2005) shows, for example, that there are plenty of cases when small population sizes translate into very different behavior from infinite population models.

2.6 Open Issues

Clearly, a considerable amount of research has been invested in CoEC, and thanks to this research we do know more now than what was known at the beginning of the field. However, I will argue that this knowledge is still incomplete and insufficient from a practical standpoint.

When a CoEA failed, the question that arose was "*Why did it fail?*". What was generally meant by failure was poor values in the external objective metric. The first step in trying to answer this question was to analyze the various internal features of the algorithm at runtime (i.e. the dynamics), like internal fitness, genetic diversity, etc. This type of research identified and named the (unfortunately) large number of pathologies of CoEAs described in section 2.4.3. While valuable, this research answered the question "*Why did it fail?*" only at a superficial level. For instance, saying "The CoEA failed in this case because it was cycling" immediately raises the question "Why did it cycle?". I believe that the question that was really answered was "*How can a CoEA fail?*", or, more precisely, "*What run-time behaviors can a CoEC setup exhibit and which of them represent failure?*".

The deeper question that was not answered was "*What properties of the CoEC setup caused the observed run-time behavior?*". If we are to understand CoEAs in a way that will facilitate successful application, we need to start providing answers to this question and this is what I claim my dissertation does.

This is achieved by means of a holistic analysis that investigates how the interplay of problem properties and algorithm properties influences dynamics and how dynamics determine performance.

Chapter 3

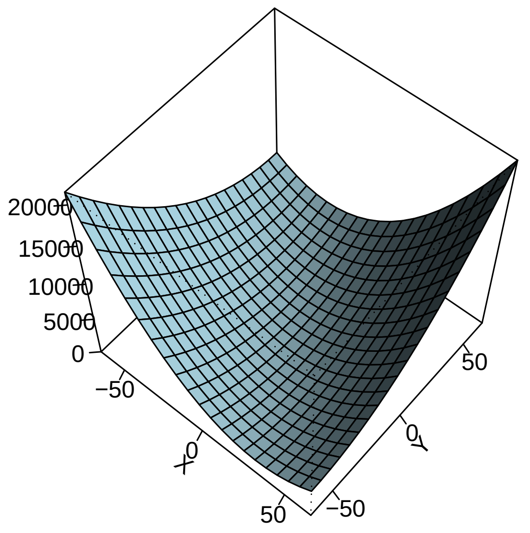
New Tools for CoEC Analysis

The main hypothesis of this dissertation is that dynamics hold the key to understanding coevolutionary algorithms and therefore their analysis should provide insights into the dependency **problem properties** + **algorithm properties** → **performance**. This chapter presents an example that supports this hypothesis and also serves as the stage for introducing the main analysis tools used in the remainder of the dissertation.

The first section shows some intriguing performance effects obtained when attempting to tune some basic parameters of a simple CoEA on "simple" problems. The following three sections focus on understanding these effects. Section two introduces a new way of tracking the dynamics of the algorithm, based on best-individual trajectories. This points to an influential problem property (named *best responses*) discussed in section three. Section four combines all this information in order to explain the observed performance effects. Section five introduces a family of problems that isolate the best-response property. The results of re-running the experiments on these new problems align with the explanation provided by the analysis. In the light of these insights, new, refined hypotheses are formulated and they will be tested (and confirmed) in the following chapters.

3.1 An Intriguing Example

Consider the task of multi-parameter function optimization, reformulated as a cooperative compositional coevolutionary problem and approached with a CoEA, as described in section 2.2.3. This section shows how, even for simple functions and a simple CoEA, intriguing performance effects are observed when attempting to tune the algorithm for better performance.



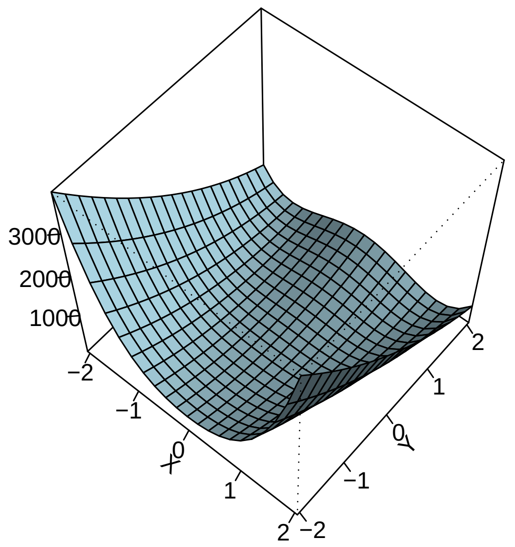


Figure 3.1: Left: $offAxisQuadratic$. Right: $rosenbrock$.

3.1.1 Two Functions

The two functions used for the example are common test functions from the EC function optimization literature, $rosenbrock$ and $offAxisQuadratic$. These were previously analyzed both in standard EC and in CoEC settings (Wiegand et al., 2001). They are defined as follows:

$$
\begin{aligned}
&offAxisQuadratic(x, y) = x^2 + (x + y)^2,\\
&\qquad x, y \in [-65.536, 65.536];\\
&rosenbrock(x, y) = 100(x^2 - y)^2 + (1 - x)^2,\\
&\qquad x, y \in [-2.048, 2.048].
\end{aligned}
$$

Figure 3.1 shows the three-dimensional surfaces described by the functions[1]. Our task is to find the pairs (x, y) in which the functions reach their minimum value (0 in both cases). For $offAxisQuadratic$ this is the point $(0, 0)$ and for $rosenbrock$ it is $(1, 1)$.

From a traditional EC perspective, these landscapes are similar with respect to properties such as continuity, modality, ruggedness, etc. But, as this chapter will reveal, there are important differences when using coevolutionary algorithms to optimize them.

[1]All plots in the dissertation were generated using (R Development Core Team, 2006)

3.1.2 A Simple CoEA

In order to be able to approach such problems with a CoEA, one needs to reformulate them as compositional coevolutionary problems. The "natural" way of doing this is to have roles correspond to parameters, role behaviors to parameter values, and potential solutions to tuples of values, one for each parameter. Also, traditionally, all roles share the same metric of behavior success, namely the function to optimize. The domain thus defined features a cooperative, variable sum, symmetric payoff. The solution concept is ideal team, as defined in section 2.2.4. The CoEA then maps each each parameter (role) into a separate population. For the above landscapes this results in two populations, one evolving values for the x function parameter and the other evolving values for the y parameter.

The design decisions for the experiments were based on two heuristics: keep things simple and use reasonable settings. This led to the use of the "vanilla" CoEA described below.

Each population uses a non-overlapping-generational EA with a real-valued representation, binary tournament selection, and a Gaussian mutation operator with fixed sigma. Sigma was fixed for simplicity, but its value depends on the size of each function's domain: 0.08 for $rosenbrock$ and 2.6 for $offAxisQuadratic$ (namely about 1/50 the size of the variables' range). To obtain effects similar to a $1/L$ bit-flip mutation for the binary representation, the probability that the mutation alters a gene (and therefore individual) was set to 90%. The two populations had equal sizes, which were varied as described in the next section.

The simplest method of evaluating an individual in one population is to couple it with the current best member of the other population and the value of the function at that point is assigned as fitness. As mentioned in the previous chapter, in cooperative domains this has been termed *single-best collaboration method* by Wiegand (2004), while in competitive domains it is referred to as *last elite opponent (LEO) evaluation* (Sims, 1994).

The two populations take turns in evolving, which means that during each generation only one population of the two is active. Using the communication lingo introduced in section 2.3.2, this translates into synchronous communication and sequential flow.

The pseudo-code of the algorithm for one X generation is given below for clarity (the one for Y can easily be inferred):

- evaluate the X population using the current y_{best};
- determine x_{best};
- select parents according to determined fitness values;
- breed;

At the begining of each run, both populations are initialized uniformly random across the domain. The algorithm starts by evaluating the members of the initial X generation in conjunction with a random y individual. For the first Y generation (second generation of the run) the x_{best} used is the actual best x individual from the first (X) generation.

For any experimental setup 100 independent runs were conducted.

3.1.3 Interesting Performance Effects

As is fairly standard with EA applications, we now go through a short tuning exercise to improve the performance of our CoEA. In this illustration we experiment with different population sizes and with the inclusion of elitism, while keeping a fixed budget in terms of the total number of evaluations.

We experimented with a total of 4000 evaluations for the whole system and the following setups: population size 5 x 800 generations, population size 10 x 400 generations, population size 20 x 200 generations, population size 50 x 80 generations, population size 100 x 40 generations, population size 200 x 20 generations. For all experiments, both populations had the same size and during each generation only one population was active (therefore performing evaluations). Population size 200 was the largest used, as increasing it more would leave the CoEA with too few generations for any evolution to take place. Elitism was used in the context of population size 10.

Figures 3.2 through 3.5 show best-so-far (left) and best-of-run (right) fitness statistics, summarized over 100 runs. Every point on a best-so-far curve corresponds to a certain number of evaluations and displays the median[2] over 100 runs of the best fitness found after that number of generations. 95% confidence intervals[3] for the medians are displayed

[2]As it was shown in (Popovici and De Jong, 2003), fitness distributions are often not normal, thus using the median tends to be more appropriate than using the mean.

[3](R Development Core Team, 2006) states that the confidence intervals are computed as +/- 1.58

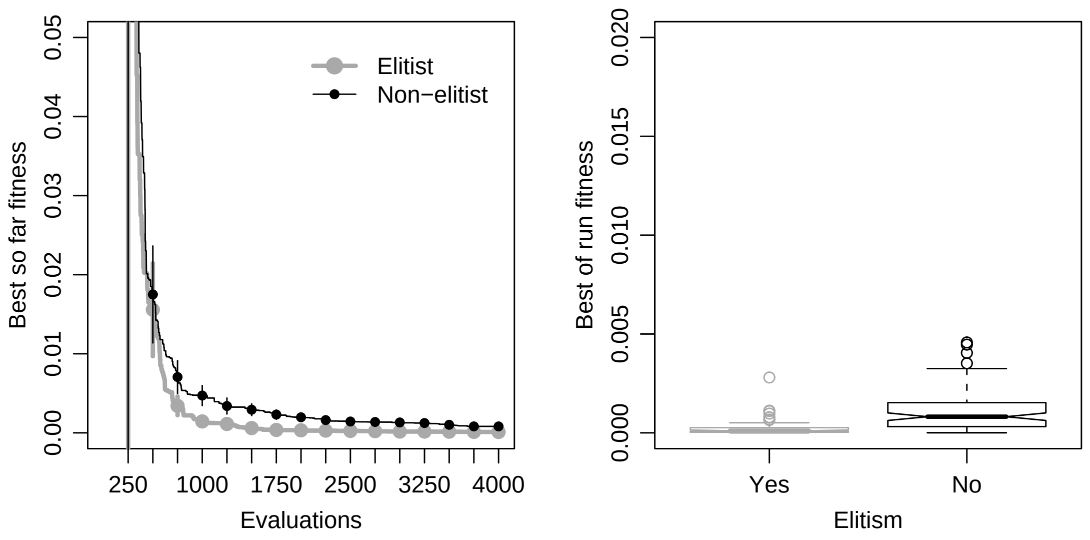


Figure 3.2: $offAxisQuadratic$ – Effects of **elitism**.

only every 25 evaluations, for visibility.

Best-of-run statistics are presented in the form of boxplots, which permit concise, comparative visualization of the median (center line), the 95% confidence interval for the median (notch around the median line), the inter-quartile range (box), the outliers (circles) and the spread of the remaining data (dotted lines and whiskers).

The best-of-run boxplots compare performance in terms of the quality of the best solution found given a certain time budget (number of evaluations). The best-so-far plots provide less detailed information (only median and its confidence interval) for a particular time budget, but do so for the history of the whole run. Additionally, they allow extracting performance comparisons in terms of amount of time needed to reach some quality threshold.

Note that the y ranges for the plots are individually adjusted for best visibility. Specifically, the ranges for best-of-run boxplots are narrower than those for best-so-far plots, in order to provide zooming in for the end of the run.

IQR/$\sqrt{n}$, where IQR is the inter-quartile range and n is the number of samples. This formula gives roughly 95% confidence intervals, according to Chambers et al. (1983) and McGill et al. (1978) and is said to be rather insensitive to the underlying distributions of the sample.

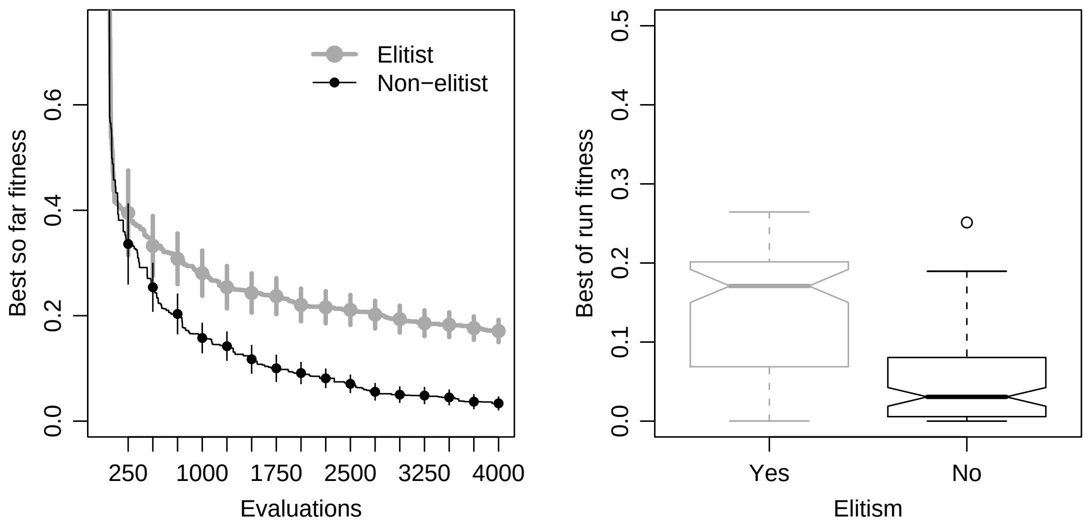


Figure 3.3: *rosenbrock* – Effects of **elitism**.

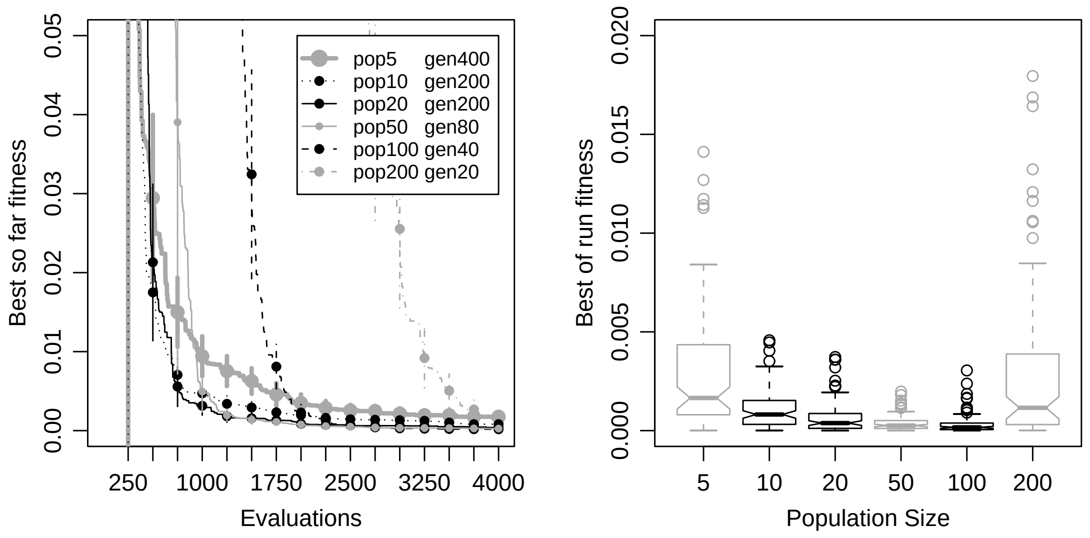


Figure 3.4: *offAxisQuadratic* – Effects of **population size**.

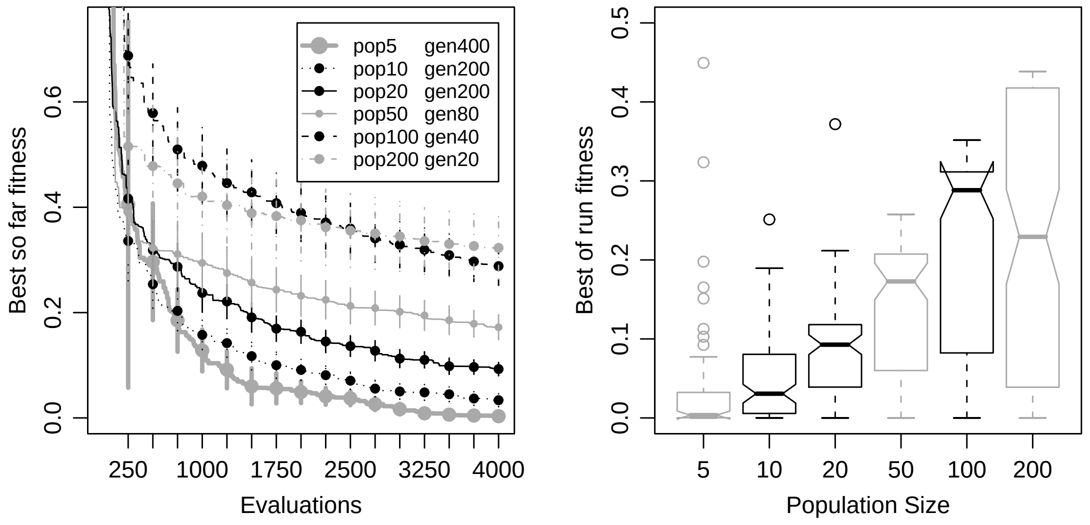


Figure 3.5: *rosenbrock* – Effects of **population size**.

The results obtained are rather surprising. Inspecting the best-of-run plots first, one can see that on the *offAxisQuadratic* landscape, introducing elitism improves performance and increasing the population size has the effect of first increasing performance and then decreasing it. This is exactly what we would expect based on our experience with standard EAs. However, on the *rosenbrock* landscape, introducing elitism resulted in significant decreases in performance, as did increasing population size!

Turning to the best-so-far plots for additional information, we see that for elitism the "winning" algorithm is decided very early in the run and, given the slope of the curves, it appears unlikely that the order of the two would change in the near future, were the time budget increased. For population size, this is mostly the case for *rosenbrock*. For *offAxisQuadratic*, the smaller population sizes improve at a fast rate at the beginning and then level off, while the larger population sizes improve at a slower rate but do so for longer. The point on the population size axis at which the performance trend switches from increasing to decreasing depends on the size of the time budget. With smaller budgets, this point will correspond to smaller population sizes, while with larger budgets it will correspond to larger population sizes.

Clearly, there is something very different about how the CoEA works on these two landscapes. We analyze this difference in the remainder of this chapter, by looking into the dynamics of the algorithm.

3.2 Best-individual Trajectories

Dynamical systems theory (Alligood et al., 1996) has proved to be a good source of inspiration for constructing methods of analysis for coevolutionary algorithms. Such methods have been reviewed in section 2.4.2. Some of the works used the term *coevolutionary dynamics* to refer to population-level dynamics, while others used it for individual-level dynamics. Here, I take the latter approach and talk about dynamics of *individuals* rather than population(s). Specifically, I analyze the *time trajectories of best-of-generation individuals across the search space*. There are two reasons for this. First of all, we are trying to analyze the performance of CoEAs for optimization, where the main concern is with the best individuals the algorithm produces. Second, we wanted to understand the behavior of a basic CoEA before moving to more complex ones, and the simplest one out there (that we picked for our initial experiments) uses a single-best collaboration strategy for evaluation.

The analysis of such trajectories exposed a problem property (named *best responses*) that has a strong influence on the behavior of CoEAs, as it will be shown time and time again in this dissertation. This property will be described in the next section. This section focuses on best-individual trajectories.

The fact that our basic CoEA alternates populations suggests a *line-search*-like way of operation. It therefore makes sense to plot the best individual of one generation (and therefore one population) by coupling it with its collaborator during fitness assessment (here, the best individual of the other population at the previous generation). We thus obtain one point of the search space per generation and we connect these points chronologically. It is obvious that the lines connecting them will only be vertical (when connecting an X generation with the following Y generation) and horizontal (when connecting a Y generation with the following X generation).

An example of the result of this procedure for a single run for both $offAxisQuadratic$ and *rosenbrock* can be seen in Figure 3.6. The trajectory is displayed as grey lines con-

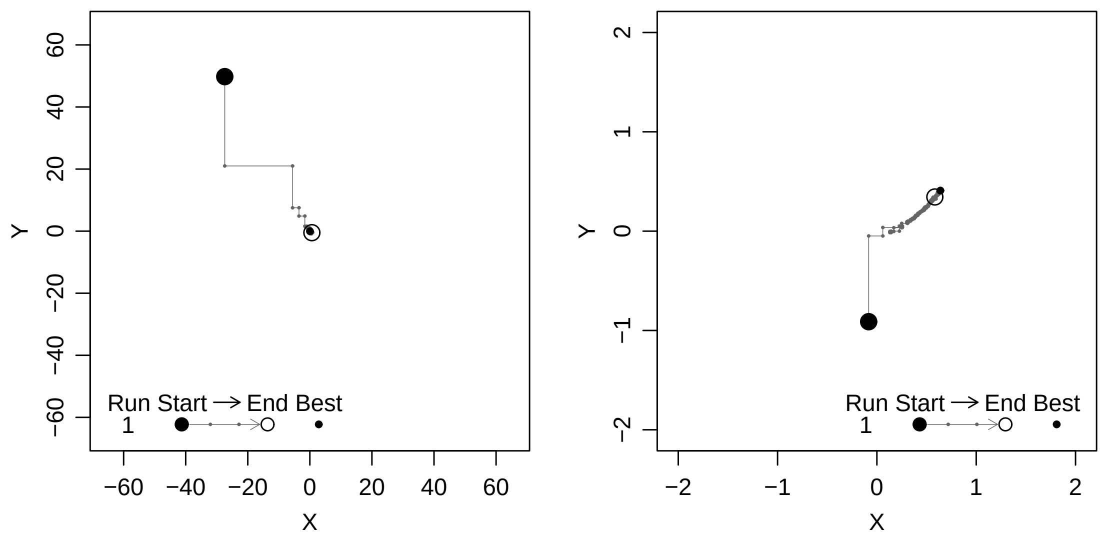


Figure 3.6: Best of generation trajectory for sample runs. Population size 20, no elitism. Left: *offAxisQuadratic*. Right: *rosenbrock*.

necting small dots (each dot representing one generation). The starting point is marked by a filled geometrical figure (in this case a circle) and the end point is marked by the same figure but empty. The best of the run (which is not necessarily the end of the run) is marked by the same figure filled and smaller.

Since for each setup we ran 100 runs, we end up with 100 such pictures. Although, as we will see later, there is value in looking at individual runs, for now we would like to get a combined view of all of them. For this purpose, we superimpose all runs of a setup on a single plot. For better visibility, we no longer draw the lines connecting the generations; instead, we use different plotting symbols for X generations and Y generations. We use the resulting images to compare the two functions. Figures 3.7 and 3.8 show cumulative best-of-generation plots for 6 of the 7 algorithm variations tried, for $offAxisQuadratic$ and $rosenbrock$ respectively.

These pictures vividly show that there are some areas of the search space that strongly attract the best-of-generation individuals. Moreover, these areas have very regular shapes. In particular, for these two domains, they seem to be lines/curves. We also notice that the X best-of-generation individuals "draw" different patterns from the Y best-of-generation

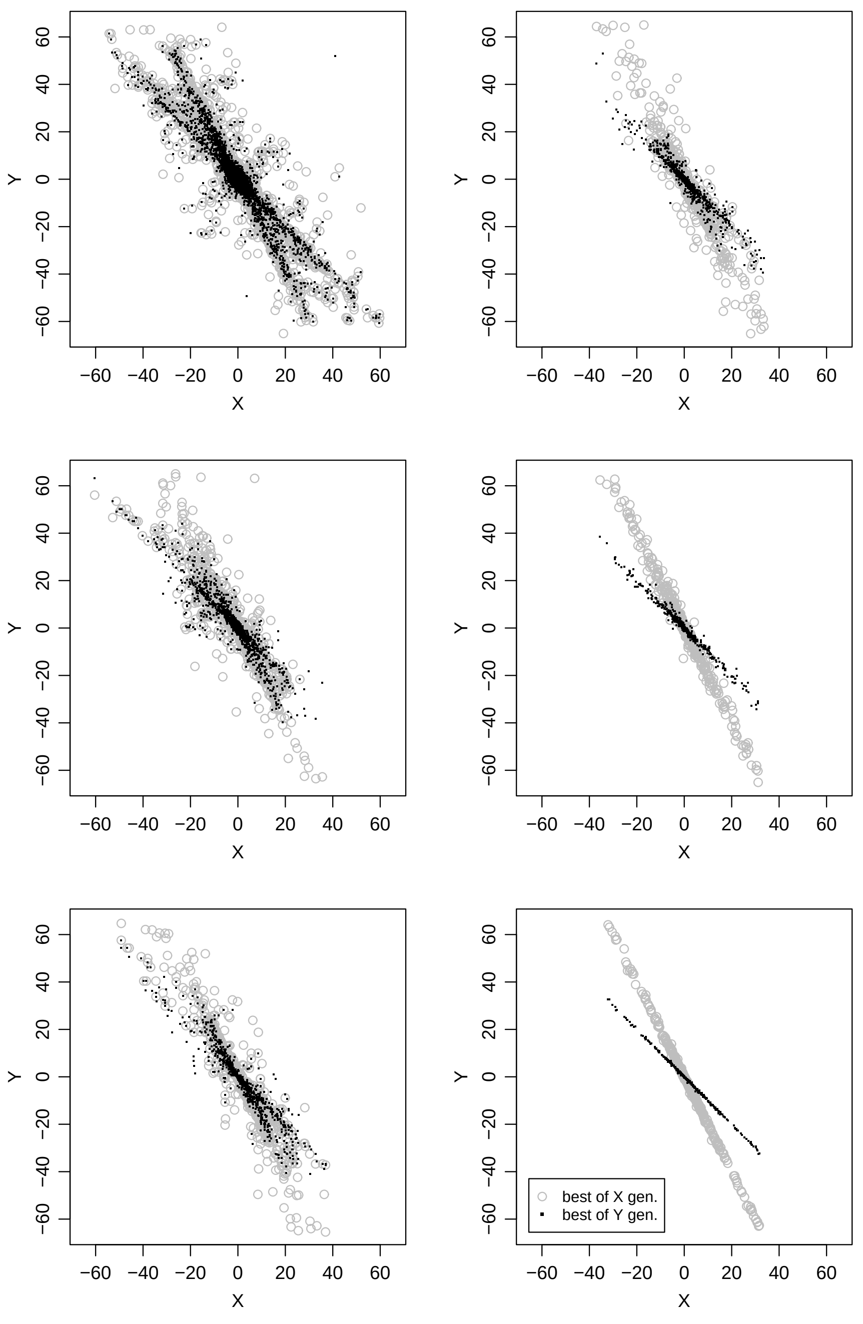


Figure 3.7: Cumulative best-of-generation points for $offAxisQuadratic$. Left column, top to bottom: population sizes 5 and 10 without elitism and population size 10 with elitism. Right column, top to bottom: population sizes 20, 50 and 200 without elitism.

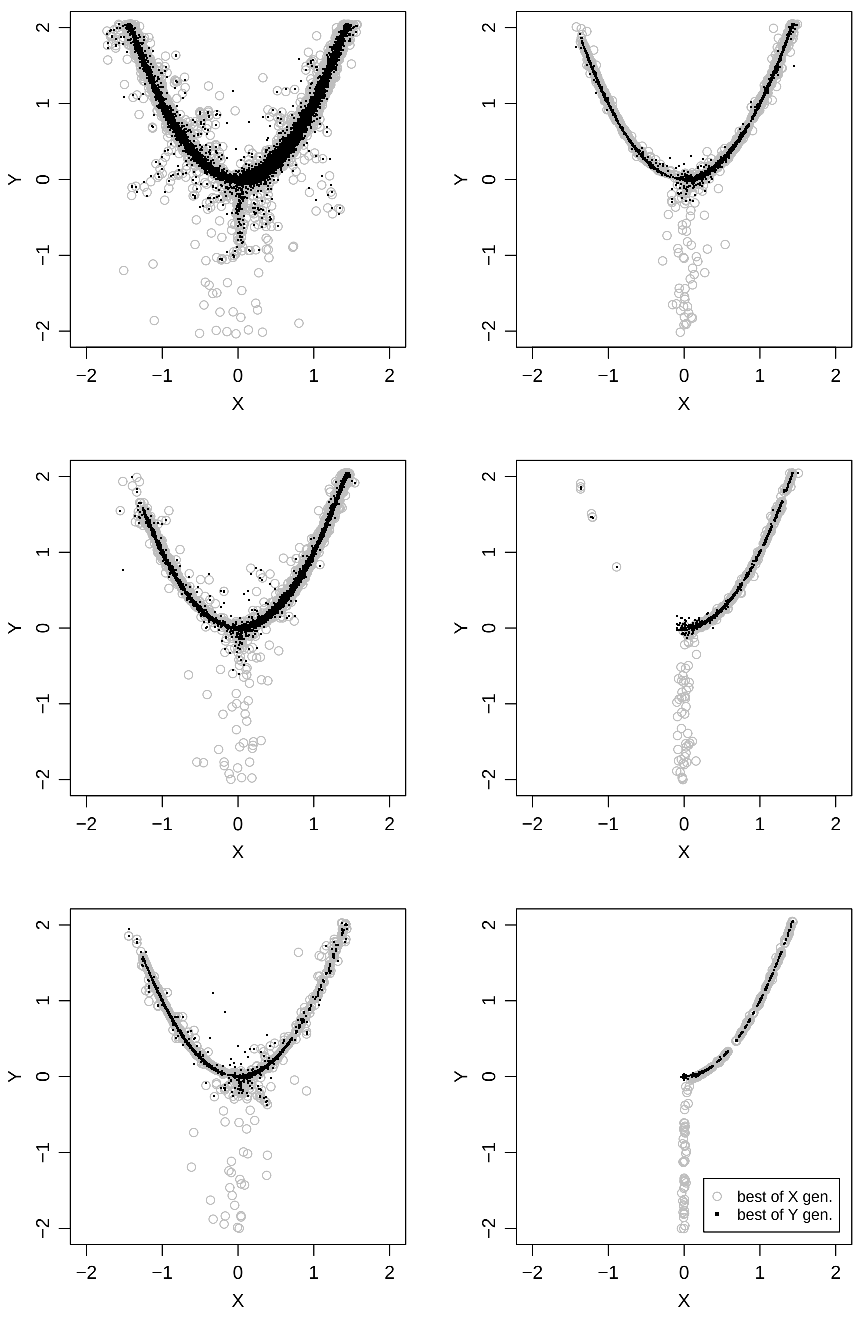


Figure 3.8: Cumulative best-of-generation points for *rosenbrock*. Left column, top to bottom: population sizes 5 and 10 without elitism and population size 10 with elitism. Right column, top to bottom: population sizes 20, 50 and 200 without elitism.

individuals. In the following sections we focus on understanding what these patterns are and why best-of-generation individuals are generating them.

3.3 Best-response Curves

Let us take another look at the way the algorithm works. It alternates populations and in each generation the active population is evaluated in combination with a single individual (the best) from the opposite population and then evolved. This is like a one-dimensional search in a slice through the domain, where the active population is searching for an individual that gives the best (in this case minimal) function value in that slice. The slice is determined by the best individual from the opposite population. In other words, the active population is trying to give the *best response* possible to the best reported by the frozen population.

Here is a more formal description. Suppose we are working with a function $f : D_X \times D_Y \to R; D_X, D_Y \subset R$. Let the current active population be the X population and the current best individual in the Y population be y_0. Then the X population is searching for an individual $x^*_{y_0}$ such that $f(x^*_{y_0}, y_0) = min_{x \in D_X} f(x, y_0)$. For every $y \in D_Y$ an x^*_y individual exists, regardless of whether the algorithm finds it or not. In general, there may be more than one, but in this chapter we deal only with functions for which for every $y \in D_Y$ there is a unique x^*_y. We can therefore define a function $bestResponseX : D_Y \to D_X, bestResponseX(y) = x^*_y$. Non-unique best responses will be discussed in later chapters. A $bestResponseY : D_X \to D_Y$ function can be similarly defined (under the same assumption of uniqueness of best responses). These functions are a property of the domain f and do not depend on any algorithm.

To obtain the formula for $bestResponseX(y)$, we need to solve the equation $\frac{\partial f(x,y)}{\partial y} = 0$ for variable y. Similarly, in order to compute $bestResponseY(x)$, we need to solve $\frac{\partial f(x,y)}{\partial x} = 0$ for variable x.

For $offAxisQuadratic$ we get:

$$\frac{\partial(x^2+(x+y)^2)}{\partial y} = 2(x+y) = 0 \Rightarrow y = -x \text{ and}$$
$$\frac{\partial(x^2+(x+y)^2)}{\partial x} = 2x + 2(x+y) = 2(2x+y) = 0 \Rightarrow x = -y/2.$$

Thus, for $offAxisQuadratic$, the formulas for the best responses are:

$$bestResponseX(y) = -y/2 \text{ and}$$
$$bestResponseY(x) = -x.$$

So they are the lines of equations $x+y=0$ and $2x+y=0$ that intersect at point $(0,0)$, which is where the function reaches its minimum.

For $rosenbrock$ the situation is somewhat more complicated. We have:

$$bestResponseY(x) = \begin{cases} 2.048 & \text{if } |x| \in [\sqrt{2.048}, 2.048]; \\ x^2 & \text{if } |x| \in [0, \sqrt{2.048}). \end{cases}$$

Obtaining $bestResponseX(y)$ requires solving a cubic equation, and we do that graphically by interpolation rather than by mathematical formula. Once again, the two best-response curves intersect in a single point, $(1,1)$, which is also where the function reaches its minimum.

All best-response curves are shown on the top row of Figure 3.9. For $rosenbrock$, note that the two best-response curves seem to overlap for some part. In fact, they only do so in $(1,1)$, but for $x \in [1, \sqrt{2.048}]$ the distance between them (i..e. between $bestResponseY(x)$ and $bestResponseX^{-1}(x)$) is less than 10^{-3}. The curves become progressively closer for $x \in [0,1]$.

The two bottom rows show an intuitive geometrical view of how these curves relate to slices through the two-dimensional surfaces described by the functions. In the left column we illustrate the construction of the $bestResponseY(x)$ curve for $offAxisQuadratic$. The plot in the middle row shows five curves (plotted in different line styles) which were obtained by slicing through the two-dimensional surface of $offAxisQuadratic$ with different values for X. For each such slice-curve, we determine the Y-value for which the curve reaches its minimum (best), plot a symbol at this point and draw a vertical line to better show the Y-value. What we have done is identify $bestResponseY(x)$ for each of the five X values considered. On the bottom plot we display the points $(x_i, bestResponseY(x_i)), i \in \{1,2,3,4,5\}$ (using the same symbols as above) and how they fit on the $bestResponseY(x)$ curve. The right column shows the similar process for $bestResponseX(y)$ for $rosenbrock$.

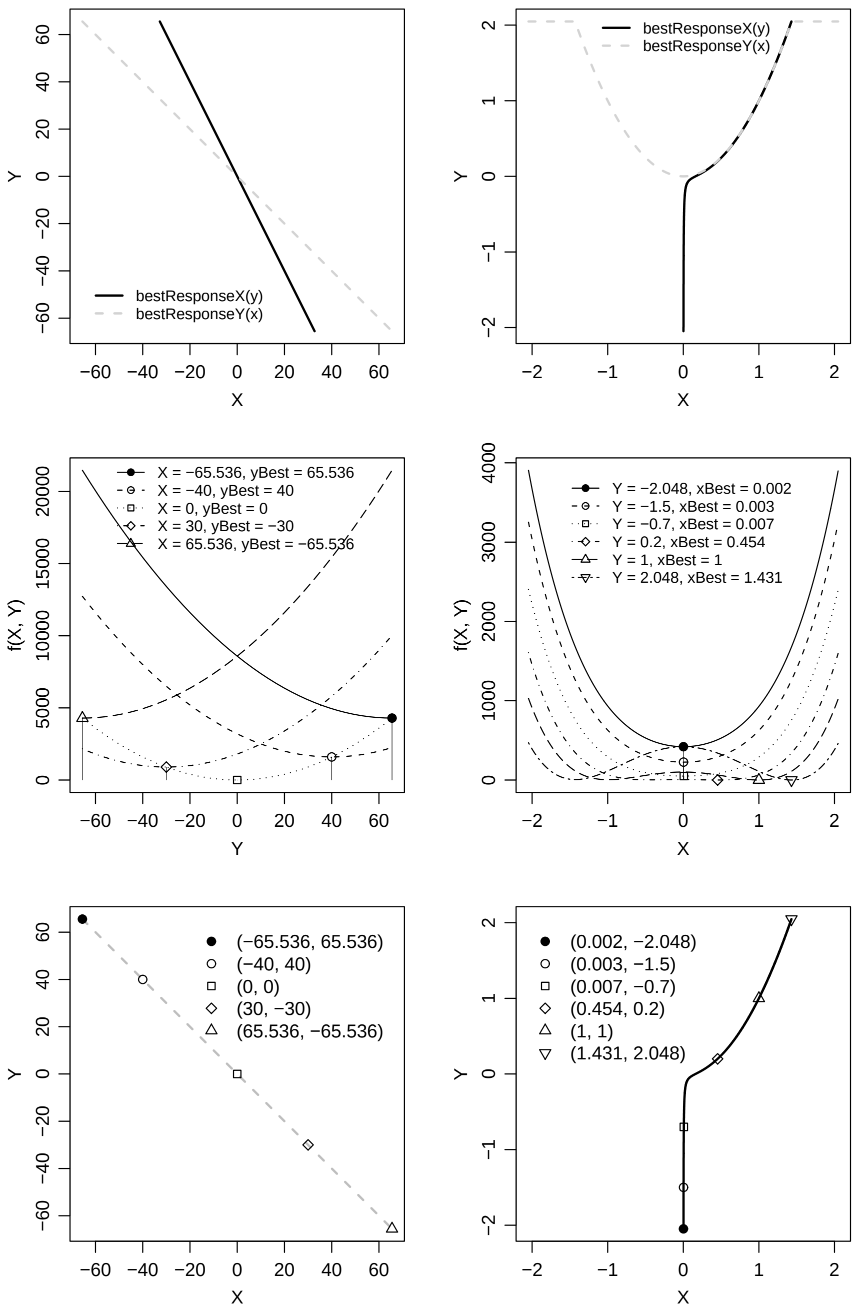


Figure 3.9: Best-response curves and slices. Left: $offAxisQuadratic$; top to bottom: best-response curves, slices with X, $bestResponseY(x)$ with points determined from slices. Right: $rosenbrock$; top to bottom: best-response curves, slices with Y, $bestResponseX(y)$ with points determined from slices.

The resemblance of the plots in Figures 3.7 and 3.8 to the best-response curves is striking. The X best-of-generation individuals were drawing pieces of the $bestResponseX(y)$ curve and the Y best-of-generation individuals were drawing pieces of the $bestResponseY(x)$ curve[4]. We begin to unravel the mystery of what this simple CoEA is doing.

3.4 Explanatory Analysis

We return to our initial goal, namely to understand why population size and elitism have different effects on the two functions. We also return to investigating individual runs. This time however, we combine on the same plot the best-of-generation trajectories as initially described in section 3.2 with the best-response curves in order to see the relationship between the two and how this relationship is affected by the various changes in parameters.

Figures 3.10 through 3.13 show sample runs for 6 of the 7 setups we have experimented with for both functions. Each plot shows 1, 2 or 3 sample runs from the 100 that were carried out for that particular setup. Typical samples were picked, while trying to still show the breadth of behaviors possible and, at the same time, keeping the images readable. Each run has a different geometrical figure to denote the beginning, end and best of the run. The left hand side plots in all figures show the whole search space, while the right hand side plots show a zoom in.

Let us first investigate the effects of increasing the population size in the absence of elitism. For both functions we see that with population sizes 5 and 10, the trajectories, while clearly influenced by the best-response curves, are quite jittery in following them. This is particularly visible in the zoomed-in plots. Increasing the population size causes the best-of-generation trajectories to follow the best-response curves closer and closer. This makes sense, as with a small population size, during one generation the evolutionary process is unlikely to come up with the exact best response. Increasing the population size increases the precision with which the algorithm approximates the best response.

While this underlying phenomena is the same, its effects are different on the two functions. For $offAxisQuadratic$, due to the relative positions of the best-response curves, a

[4]Which pieces of the best-response curves are visited depends partially on which population is active during the first generation. Section 5.3 further discusses this issue.

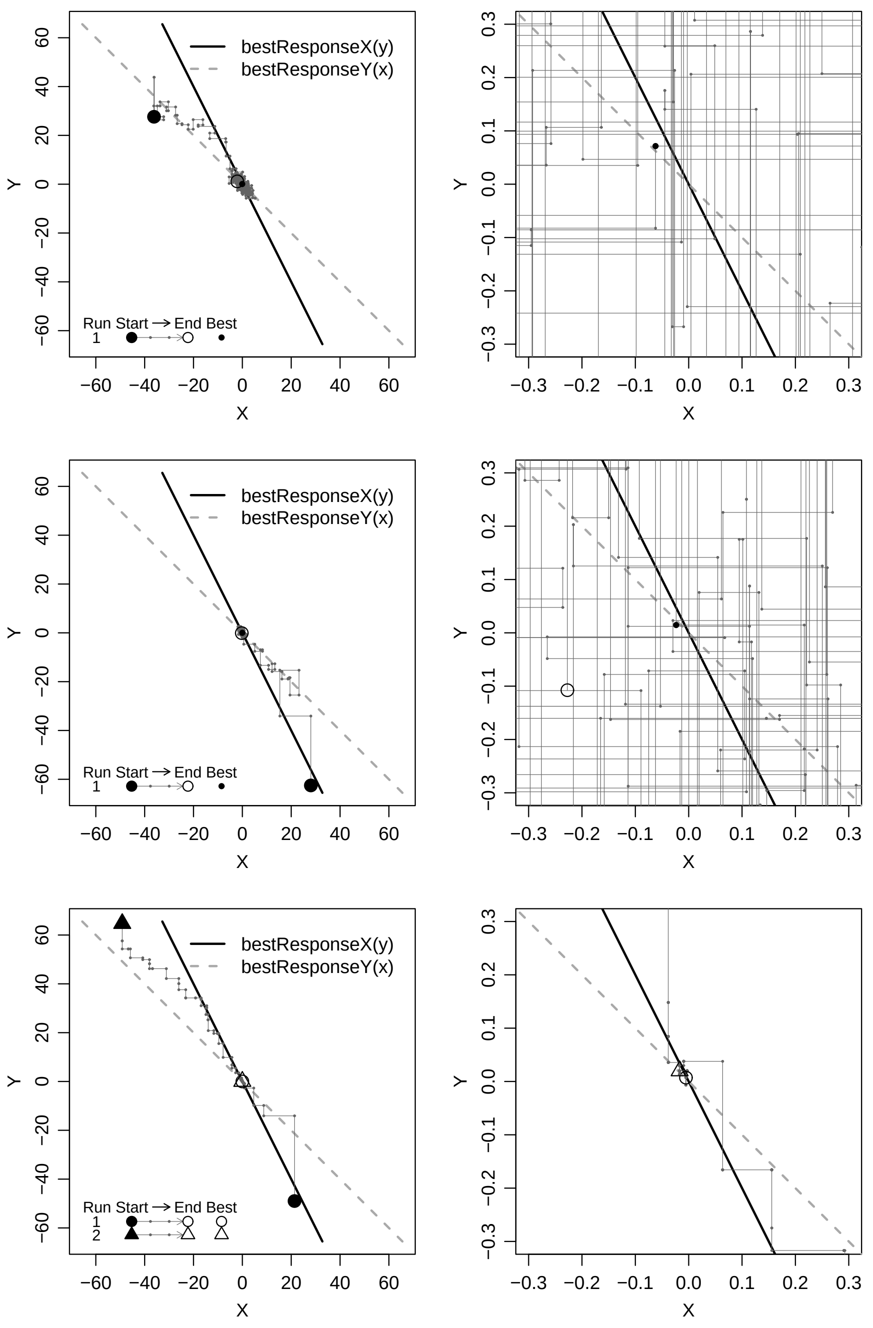


Figure 3.10: Best-of-generation trajectories for *offAxisQuadratic*. Left column shows full space, right column shows a zoom in. Top to bottom: population sizes 5 and 10 without elitism and population size 10 with elitism.

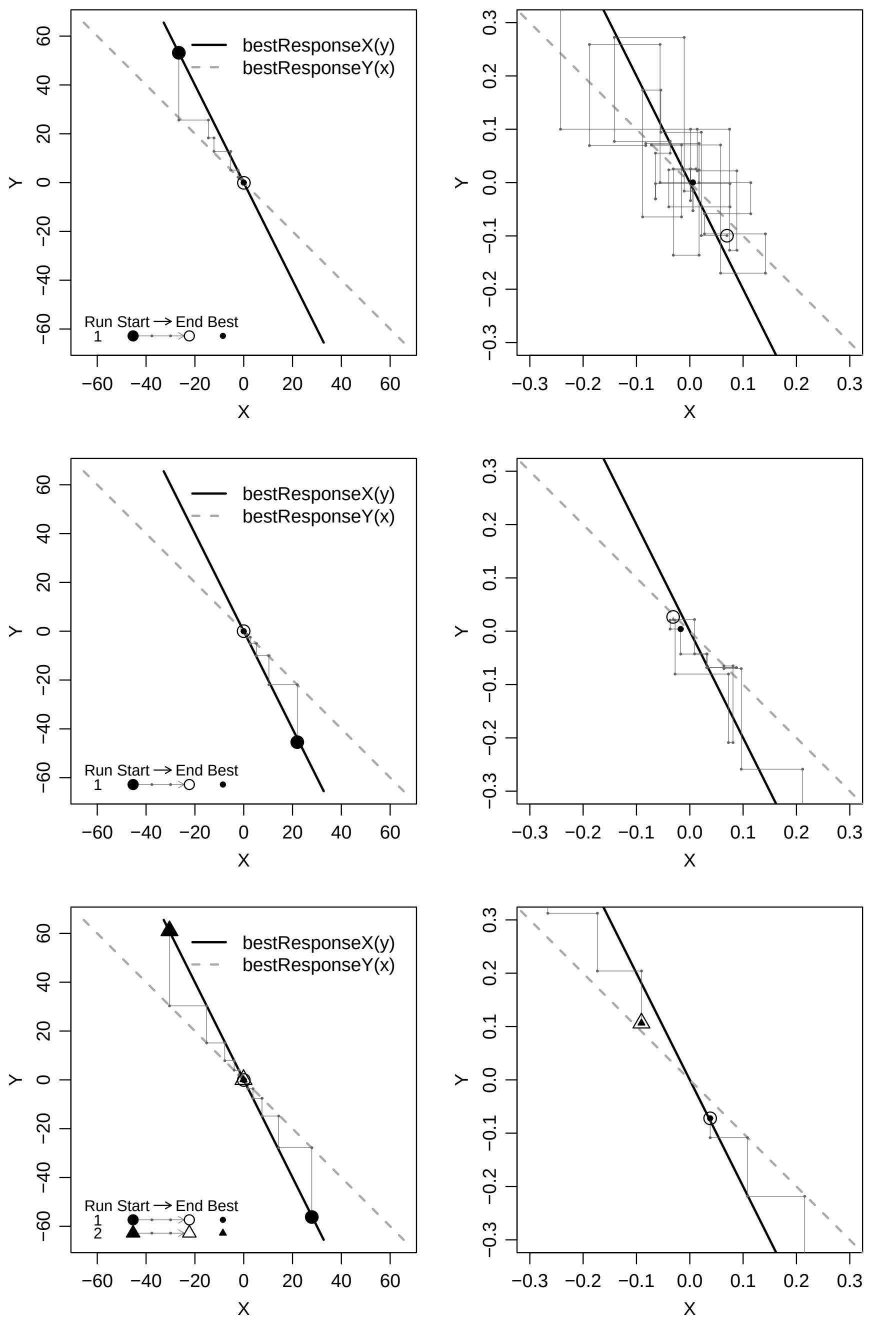


Figure 3.11: Best-of-generation trajectories for *offAxisQuadratic*. Left column shows full space, right column shows a zoom in. Top to bottom: population sizes 50, 100 and 200 without elitism.

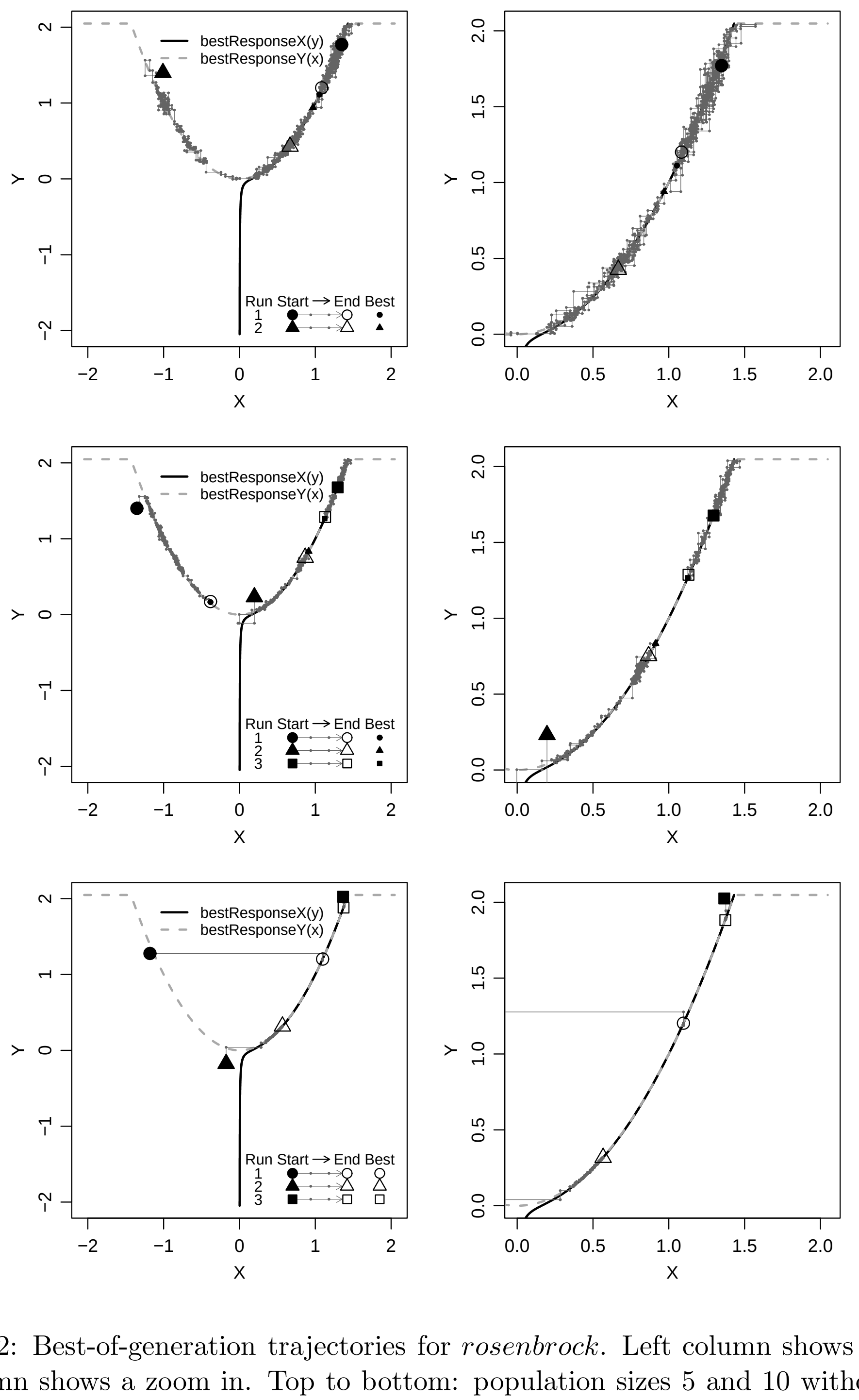


Figure 3.12: Best-of-generation trajectories for *rosenbrock*. Left column shows full space, right column shows a zoom in. Top to bottom: population sizes 5 and 10 without elitism and population size 10 with elitism.

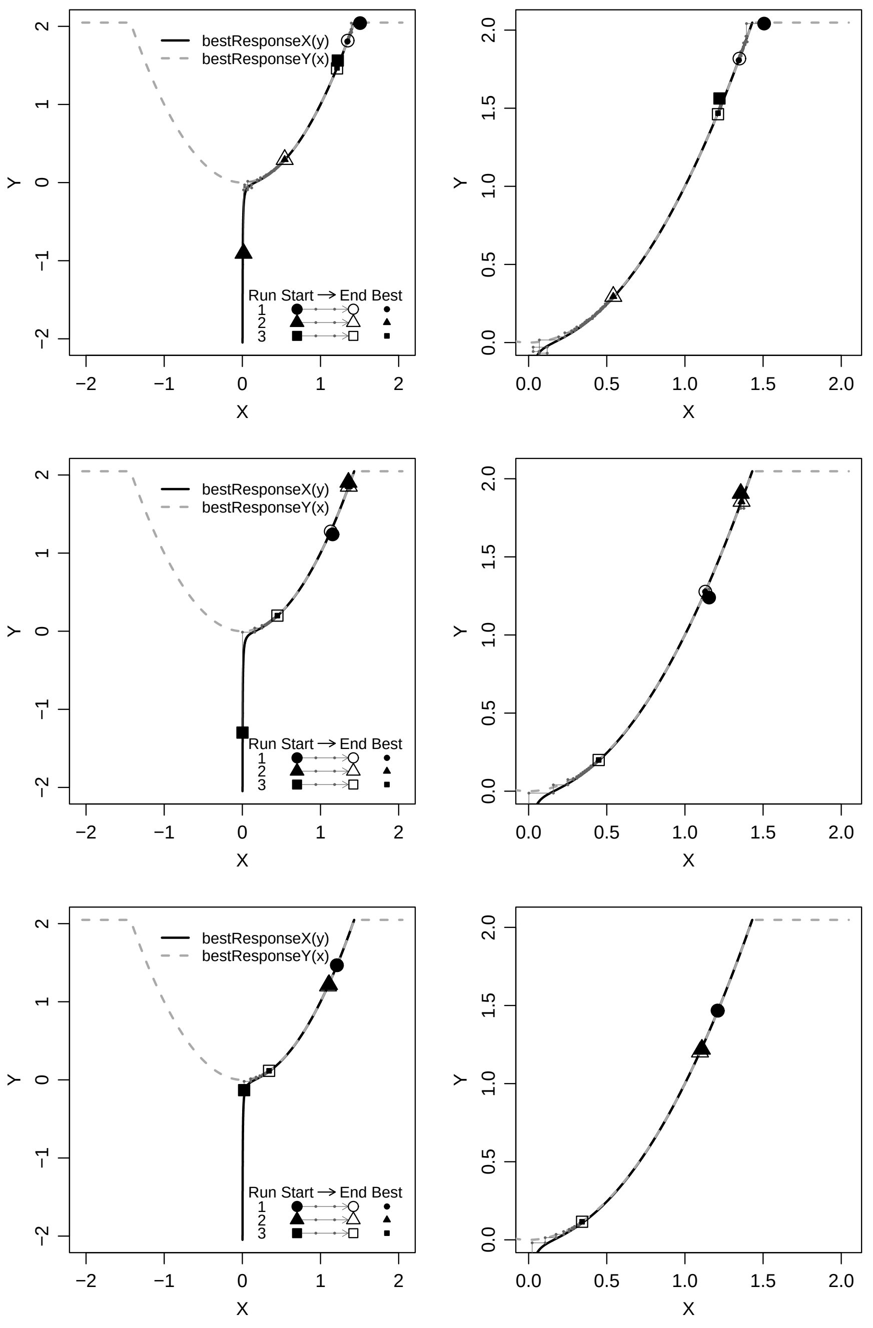


Figure 3.13: Best-of-generation trajectories for *rosenbrock*. Left column shows full space, right column shows a zoom in. Top to bottom: population sizes 50, 100 and 200 without elitism.

deterministic system exactly following them would advance like on a ladder towards the intersection point. In game theoretic terms, this intersection point is a *Nash equilibrium* for the deterministic system. With large population sizes, our algorithm closely approximates this behavior, while with small population sizes, the algorithm gets "distracted" from it. The intersection point is also where the function reaches its minimum; for both best-response lines, the function value decreases from their ends towards the middle. We now understand the initial gradual improvement in performance shown by the boxplots on the right side of Figure 3.4. But why does performance start to decrease again as we further increase the population size? Remember we are using a fixed number of evaluations, so larger population size means fewer generations. There is therefore a tradeoff between the accuracy of following the best-response curves and the number of steps taken along them. If the number of steps becomes too small (e.g., population size 200 x 20 generations), the algorithm doesn't have enough time to reach the optimum, the target towards which it is so precisely moving. This can be seen in the zoomed-in plot on the bottom row of Figure 3.11.

For *rosenbrock*, the best-response curves are such that the trajectory of a deterministic system exactly following them would after the first or second step enter the region of almost-overlap and then be forced to take extremely small steps. Once again, the intersection point of the two best-response curves is a Nash equilibrium for the deterministic system and it is also where the function reaches its minimum. With small population size, the algorithm's best-of-generation trajectory has low accuracy in following the best-response curves, thus taking large steps rather than small ones. Additionally, the large number of steps that are available for a small population size make the trajectory be highly "explorative" and thus inevitably visit points close to the optimum. With increasing population size, the accuracy of approximating the best-response increases and this means smaller step size. Moreover, the number of steps decreases (due to the fixed budget) and these two effects combined make the trajectory of best-of-generation individuals crawl and/or very quickly get stuck in a point close to where it (randomly) started. The algorithm will get close to the optimum only if by chance it started close to it. In this case high accuracy doesn't balance off a small number of steps, but instead it works to the disadvantage of performance as well. These effects can be seen in figure 3.13.

We now turn to elitism and analyze the bottom row of Figures 3.10 and 3.12. Since elitism does not decrease the number of generations, we would expect to see the same number of points as in the corresponding plots of population size 10 without elitism (middle rows of the same pictures). Instead, the elitism plots show very few points (especially for *rosenbrock*, but the same can be seen in the zoomed plots for $offAxisQuadratic$). In fact, there is the same number of points, but many are overlapping. This is because elitism causes the best-of-generation trajectory to stay put until the algorithm discovers an individual better than the current best. For these functions, increasing fitness is generally done by moving closer to the best-response curve of the current population (subcomponent). The algorithm thus stays focused on following the best-response curves. As mentioned in the discussion about population size effects, following the best-responses is good for $offAxisQuadratic$ and bad for *rosenbrock*, thus explaining the boxplots on the right side of Figures 3.2 and 3.3.

3.5 A Test Suite

At this point we have some intuition about the effects that population size and elitism have on performance in the presence of certain problem features. We further investigate this by constructing a family of functions for which we can control the degree of proximity of the best-response curves by varying one parameter, $\alpha \in [0, 1]$.

The family of functions is defined as follows:

$$BR_n^\alpha(x, y) = \begin{cases} 2y + \frac{\alpha-3}{2\alpha}(x-n) & \text{if } \alpha y < x + (\alpha-1)n; \\ 2x + \frac{\alpha-3}{2\alpha}(y-n) & \text{if } y > \alpha x + (1-\alpha)n; \\ n + \frac{x+y}{2} & \text{otherwise.} \end{cases}$$
$$n \in N; \alpha \in [0,1]; x, y \in [0, n],$$

and the two-dimensional surfaces described by the functions for $n = 10$ and $\alpha \in \{0, 0.25, 0.50, 0.75, 0.90, 1\}$ are shown in Figure 3.14. $n = 10$ was used for all the experiments with BR_n^α reported in the dissertation.

For these functions the task is maximization and regardless of α there is a unique maximum $BR_n^\alpha(n, n) = 2n$. At $\alpha = 0$ the surface is just a plane. For $\alpha \in (0, 1)$ two ridges

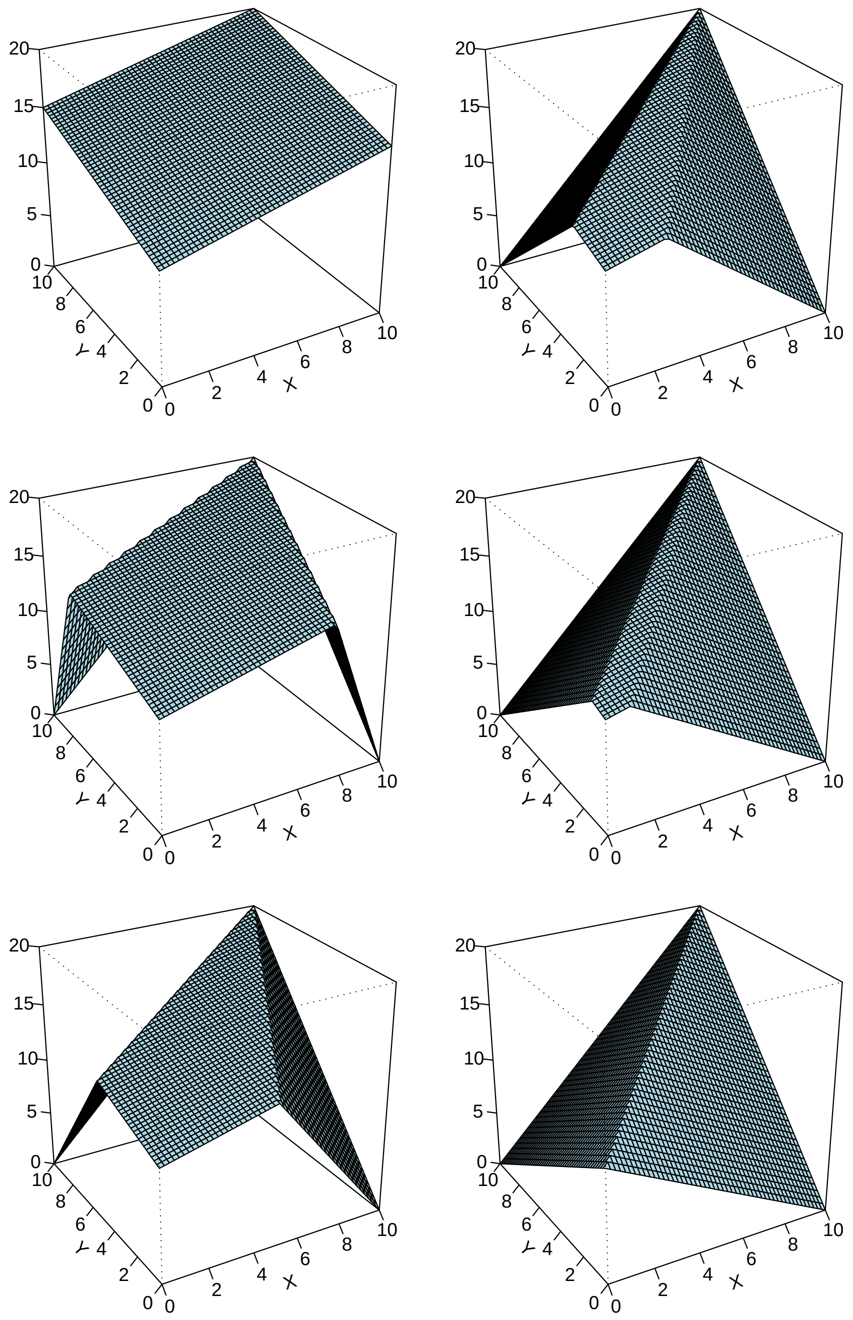


Figure 3.14: $\boldsymbol{BR_n^{\alpha}}$. Left, top to bottom: $\alpha = 0, 0.25, 0.50$. Right, top to bottom: $\alpha = 0.75, 0.90, 1$.

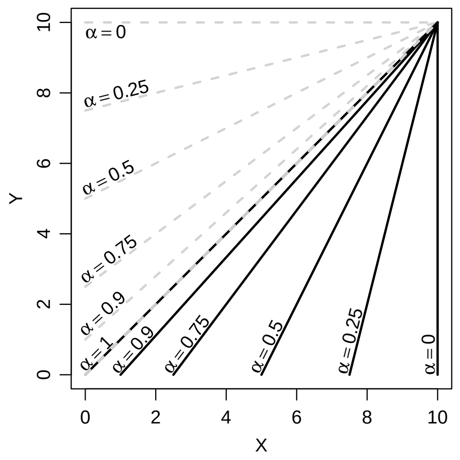


Figure 3.15: Best-response curves for BR_{10}^{α}. *bestResponseX* shown in black continuous lines, *bestResponseY* shown in gray dashed lines.

appear and they get closer and closer as α increases. At $\alpha = 1$ the two ridges merge into one.

The formulas for the best-response curves, updated for maximization, are:

$$bestResponseX(y) = \alpha y + (1-\alpha)n$$
$$bestResponseY(x) = \alpha x + (1-\alpha)n.$$

These formulas describe two lines that intersect in (n, n) (where the optimum is!). Figure 3.15 plots them for the same values of α as above. At $\alpha = 0$ the two best-response lines are perpendicular; as α increases, the angle between them decreases, till finally they overlap when $\alpha = 1$. The value of the function along any such line decreases from $2n$ in (n, n) down and towards the left (i.e. towards smaller x and y values).

Our hypotheses are as follows:

- at low α, increasing population size and introducing elitism will have beneficial effects on performance; very few steps are necessary to get very close to the optimum, so high accuracy in following the best-response curves is good;
- as α increases, we will begin to see "curving" effects for the population size performance, as the benefits of accuracy in following the best-response curves are counterbalanced by a number of generations too small to have time to reach proximity of

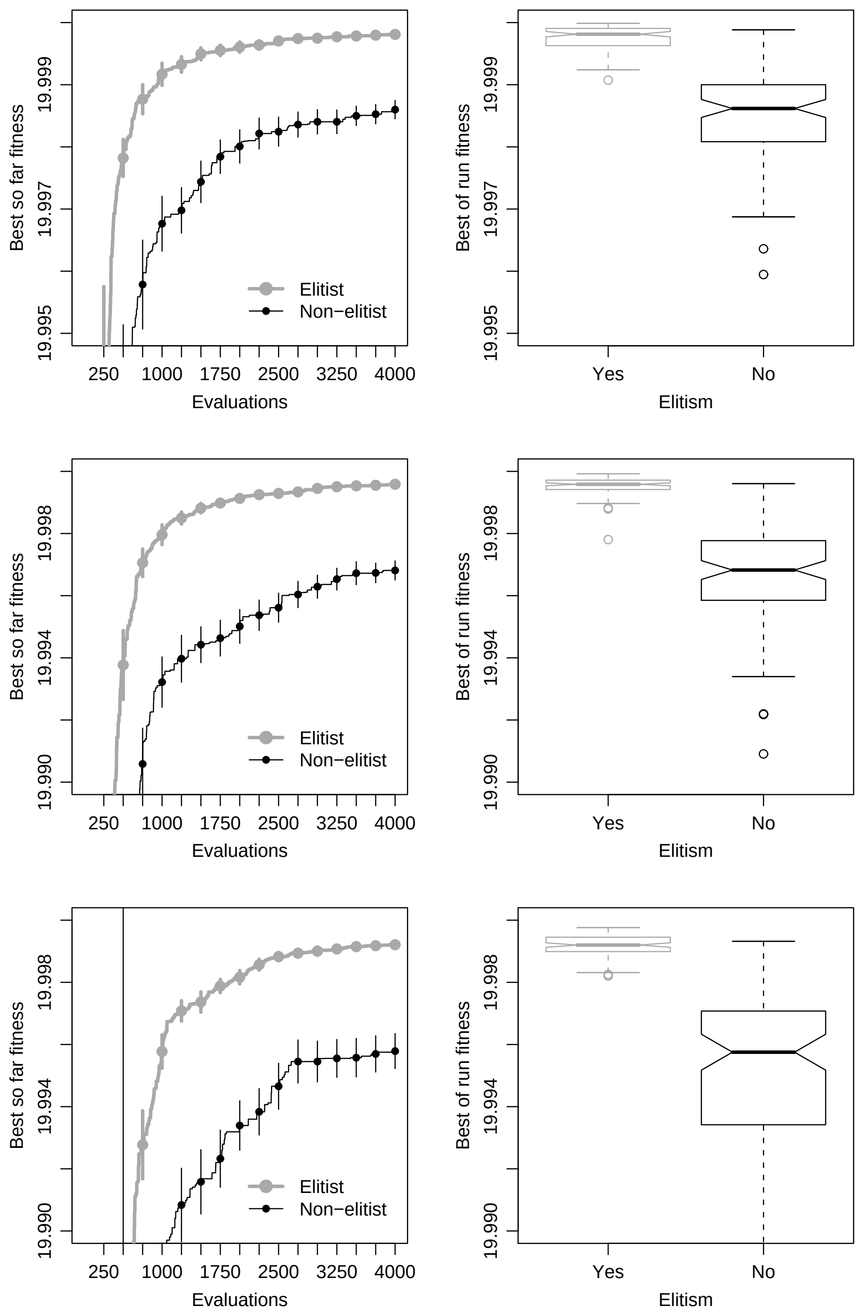


Figure 3.16: **Elitism** effects on performance (left: best-so-far, right: best-of-run) for BR_{10}^{α}. Top row: $\alpha = 0$; middle row: $\alpha = 0.25$; bottom row: $\alpha = 0.5$.

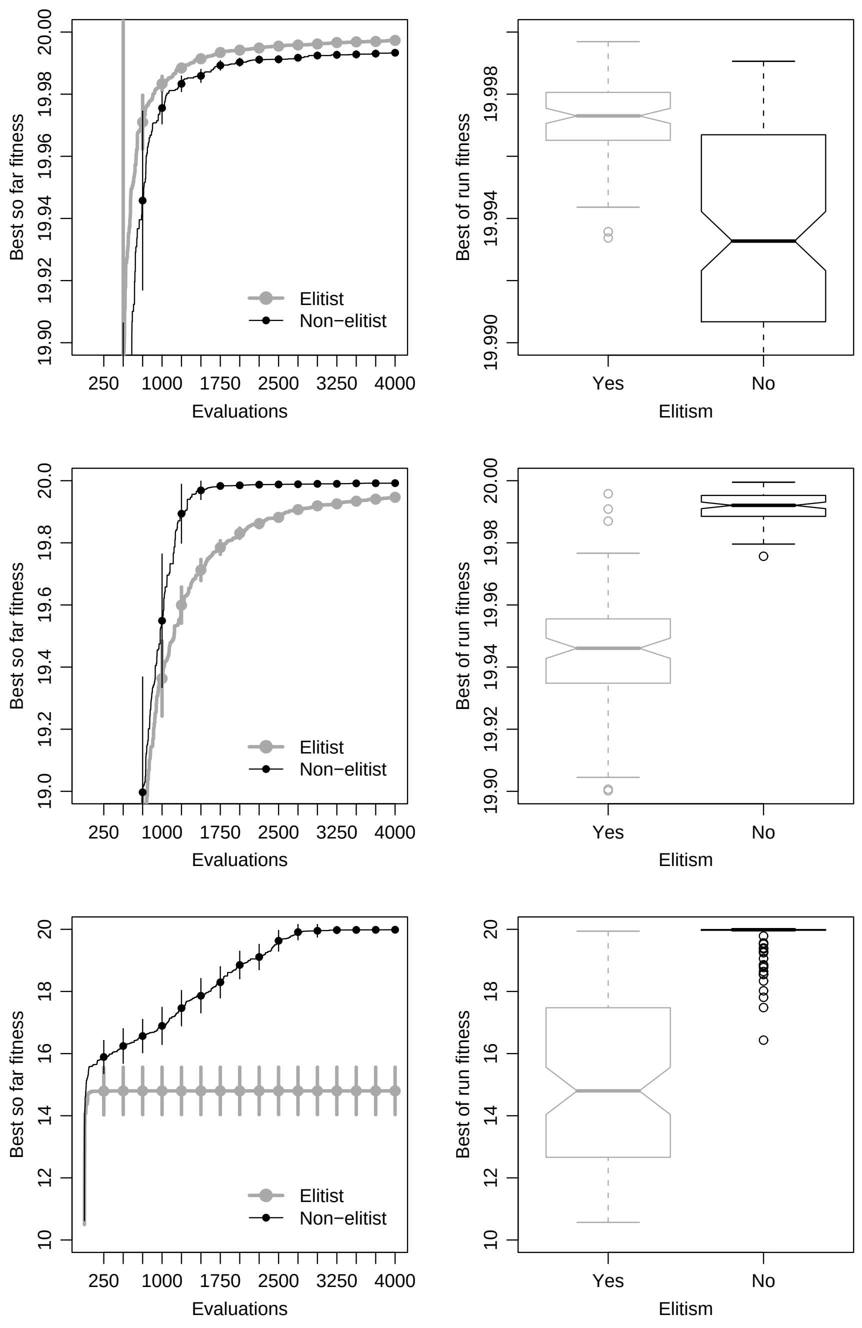


Figure 3.17: **Elitism** effects on performance (left: best-so-far, right: best-of-run) for BR^{α}_{10}. Top row: $\alpha = 0.75$; middle row: $\alpha = 0.95$; bottom row: $\alpha = 1$.

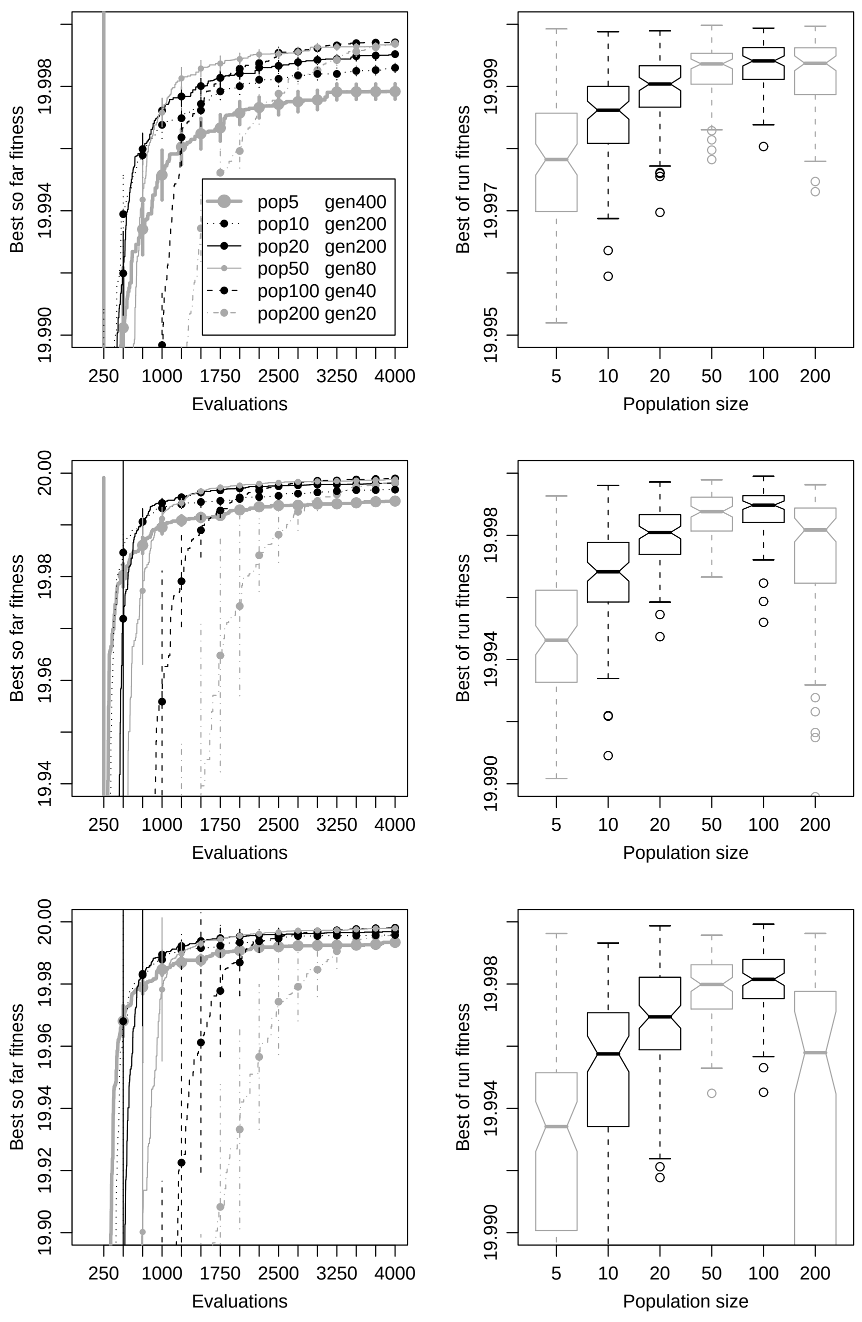


Figure 3.18: **Population size** effects on performance (left: best-so-far, right: best-of-run) for BR^{α}_{10}. Top row: $\alpha = 0$; middle row: $\alpha = 0.25$; bottom row: $\alpha = 0.5$.

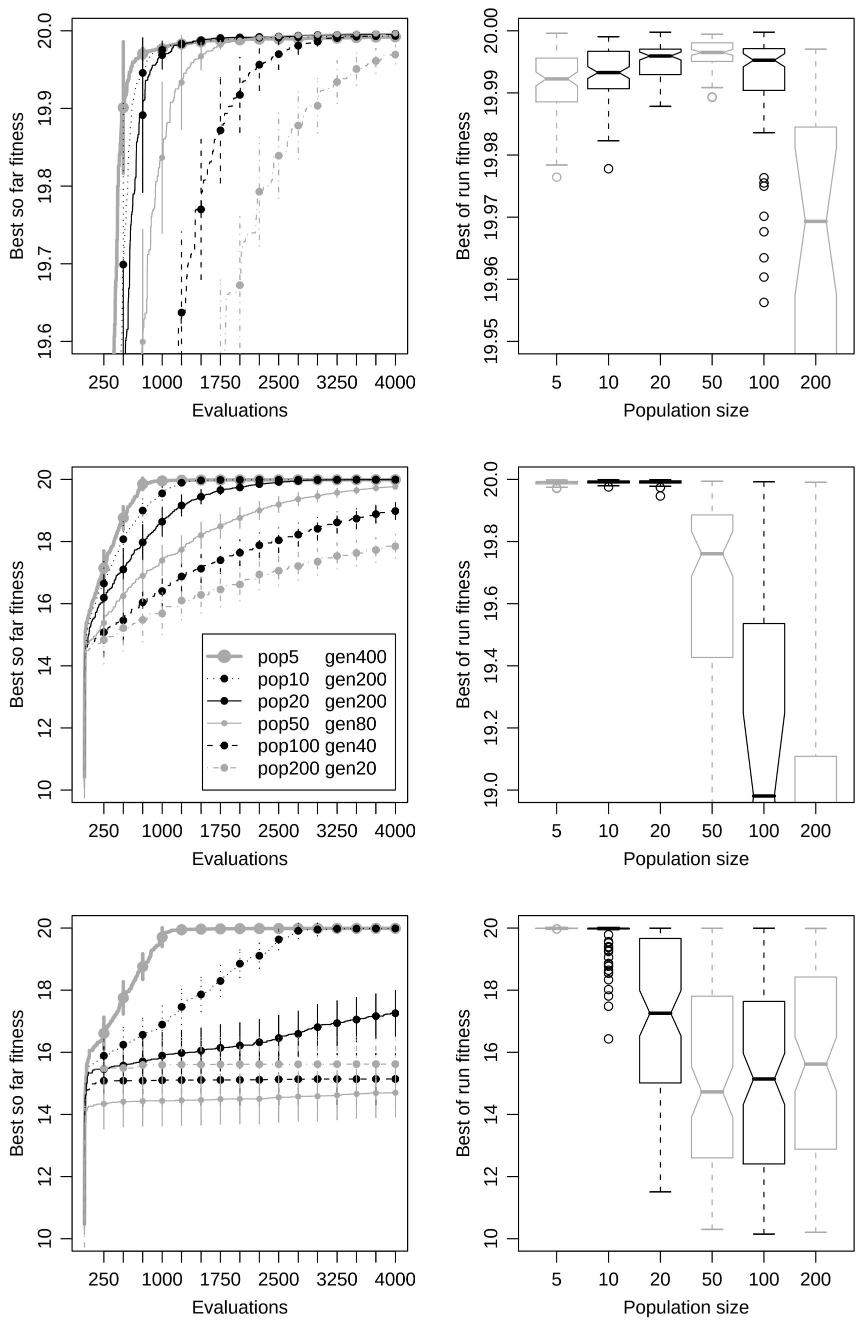


Figure 3.19: **Population size** effects on performance (left: best-so-far, right: best-of-run) for BR^{α}_{10}. Top row: $\alpha = 0.75$; middle row: $\alpha = 0.95$; bottom row: $\alpha = 1$.

the optimum; the law of diminishing returns sets in; elitism should continue to be beneficial for a while, since it does not decrease the number of generations;

- as α reaches 1, increasing population size and introducing elitism will decrease performance; closely following the best-responses is bad when they are extremely close or overlapping, as the steps taken by the trajectories are forced to be very small.

For the six values of α mentioned above we ran the same population size and elitism experiments as before. The only change was adjusting the value for the standard deviation of the Gaussian mutation to the size of these domains, namely setting it to 0.2. As can be seen in the right-hand side best-of-run boxplots of Figures 3.16 through 3.19, our expectations were confirmed. We also note on the best-so-far plots from the same figures the same time-dependent effects as before: for elitism, the better algorithm is decided early on in the run and appears unlikely to change in the future, while for population size the curving point in performance depends on the time budget.

3.6 Summary

This chapter introduced a new set of tools for analyzing the dynamics of coevolutionary algorithms, based on trajectories of best-of-generation individuals. The use of these tools provided interesting insights into the ways CoEAs function and what drives their performance, thus showing evidence supporting the hypothesis that a dynamics-focused approach is key to understanding these algorithms. In particular, these tools helped expose a problem property, namely the best responses, whose relevance for CoEC setups has been previously overlooked.

The analyses presented are instances of connected analysis, as pictured in Figure 1.1. Namely, dynamics (best-individual trajectories) were used to explain how the interplay of a problem property (best-response proximity) and an algorithm property (population size / elitism) affects performance (quality of best solution found given a fixed amount of time). The insights gained argue that the best responses are an important property of coevolutionary problems and their relationship can be a key factor in determining which algorithm settings are better for performance. Therefore, the general hypothesis was refined into the following, more concrete *hypotheses*:

- Best responses would influence the performance effects of other algorithm parameters as well; and
- Best-individual trajectories would help understand why.

Throughout the following chapters, evidence is brought in support of these hypotheses, by performing the same type of connected analysis. The approach taken is to start simple and incrementally add complexity.

Thus, chapter 5 further investigates the influence of best-response proximity (using the BR_n^α test suite) on additional (CoEA-specific) parameters.

However, clearly, the BR_n^α family of functions does not exhaust all possible interactions between best responses for cooperative payoffs. For example, one would expect that performance will not vary with population size and elitism if along a region of overlap the function value is constant. There may also be cases where there are several disjoint points (or regions) of intersection for best-response curves. We expect these to be related to the presence of local optima and performance is likely to also be affected by the shapes and sizes of their basins of attraction. Also, the best responses may not be unique. Therefore, additional test suites exemplifying such problem properties are introduced and analyzed in chapter 5.

For most practical applications, formulas for the best responses will not be available. However, as it was shown in sections 3.2 and 3.3, analyzing the trajectories of best-of-generation individuals will help infer their shapes and/or the relationships between them. This issue will be addressed in chapter 6 which features a domain that is not available in closed form.

Chapter 4

Analysis of Simple Synthetic Compositional Cooperative Problems

The previous chapter introduced a problem property (best responses), some visualization tools (best-individual trajectories) and a test suite (BR_n^α) useful for analyzing the behavior of coevolutionary algorithms. It used them to shed light on some surprising performance effects of traditional EA parameters such as population size and elitism. As a result, two new, refined hypotheses were formulated: 1) best responses would influence the performance effects of other algorithm parameters as well; and 2) best-individual trajectories would help understand why. This chapter brings evidence in support of these hypotheses, by using the same techniques to analyze three *CoEA-specific* parameters: the selection of interactions for fitness assessment and the inter-population communication frequency and flow. In all three cases, the analysis concerns the application of CoEAs to compositional problems defined over cooperative domains (with symmetric payoff) and ideal collaboration as a solution concept (as defined in chapter 2).

Each of the parameters is discussed in its own section, however all three sections follow the same format, reflecting the methodology used. First, related work (if any) is reviewed. Second, experiments are performed on problems from the BR_n^α family, varying the targeted CoEC parameter and observing the performance effects. Third, an analysis of algorithm dynamics is conducted, as a means to explain performance. Finally, the knowledge gained from this analysis is used to make (and then confirm) predictions for additional functions from the literature.

As a reminder, the BR_n^α functions are given in closed form, thus have analytically computable best responses and known optima. Moreover, the best responses are continuous. These restrictions will be relaxed for the test suites introduced and studied in chapters 5 and 6.

4.1 Collaboration Methods: Sample Size & Fitness Bias

The fact that in order to be evaluated individuals must participate in interactions with other evolving individuals is the main feature of CoEAs. As discussed in section 2.3.2, choosing out of all possible interactions which ones to assess can be done in a variety of ways. In the context of cooperative CoEC, this aspect of the algorithm has been termed the *collaboration method*. This section examines a number of such methods, specifically some individual-centric ones. When evaluating an individual in one population, one or more "collaborator" individuals are selected from the other population. The individual to be evaluated interacts in turn with each collaborator and from each such interaction it receives a value. To assess fitness, these values are then aggregated somehow.

In line with previous research, (Wiegand et al., 2001; Panait, 2006), the aggregation method used here consists in taking the optimum of all values. This choice is fixed for all experiments in the dissertation. In this section, what is varied is the sample size, i.e. the number of collaborators used for one fitness evaluation, and the fitness bias, i.e. whether fitness is used to influence how these collaborators are chosen.

4.1.1 Related Work

Collaboration methods were investigated by Wiegand et al. (2001, 2002) and their performance effects were shown to be dependent on the way the problem was decomposed and the pieces assigned to different populations. However, the nature of the relationship between problem decomposition and problem difficulty in the context of different collaboration methods was not a trivial one to investigate.

Wiegand (2004) describes a class of problems (functions) which he calls "linearly separable"[1] that can be very easy for CoEAs if decomposed in agreement with its separability. For such cases, using a single, *fixed* collaborator[2] for all individuals in the current popula-

[1]Informally, this means there exists a clustering of the function's parameters and a set of corresponding functions, one per cluster (thus operating on disjoint groups of parameters), such that the original function can be written as a sum of the functions in this set. For a formalism, please consult Definition 13 in section 4.1.1 of (Wiegand, 2004).

[2]This individual can vary from one generation to another, but it should be fixed during the generation.

tion is enough for tackling the problem. What is not clear is when would one need more complex schemes for evaluation. Wiegand et al. (2001) shows that the simple presence of "non-linear" interdependencies between the pieces of the problem represented in different populations (called "cross-population epistasis") is not enough to necessitate more complex evaluation schemes.

Wiegand et al. (2002) and Wiegand (2004) take the analysis further to see what type of such cross-population epistasis can really cause problems for the single-best collaboration method. In the context of a study on pseudo-boolean functions, the hypothesis offered is that to blame is the *contradictory* cross-population epistasis, a situation in which "the linear effects of lower order components existing in different populations are opposite in magnitude and direction of the higher order blocks" and thus "the algorithm is tricked into believing that the lower order components accurately predict the total function value, when in fact they are misleading". That analysis followed an epistasis-focused approach. Problems with and without contradictory cross-population epistasis were constructed and the results of running a CoEA with varied collaboration mechanisms were recorded. The observed performance agreed with the hypothesis. It seemed like this was the end of the story. In this section we shall see that is not the case.

4.1.2 Experiments

We start by showing that contradictory cross-population epistasis is not necessarily a good indicator of difficulty. We do this by using two functions from the BR_n^α family (namely, BR_{10}^{1} and $BR_{10}^{0.75}$) that both exhibit this kind of epistasis for the natural decomposition (one piece per function parameter), however for one of them the single-best collaboration method is all there is needed to solve the problem, whereas for the other this method does poorly and can be significantly improved upon.

To see the contradictory interaction of the X and Y components for BR_{10}^{1}, consider any point along the diagonal ridge. Changing the x value while keeping the y constant will result in a decrease in function value, regardless of whether x is increased or decreased. Similarly, changing the y value while keeping the x constant will always result in a decrease in function value. However, if both x and y are changed together, in some cases the function value will increase (e.g. moving *up* along the diagonal) and in other cases it will decrease (e.g. moving *down* along the diagonal). Here's a concrete example:

$$\left.\begin{array}{l} BR^{1}_{10}(5,4) < BR^{1}_{10}(4,4) \\ BR^{1}_{10}(4,5) < BR^{1}_{10}(4,4) \end{array}\right\} BR^{1}_{10}(5,5) > BR^{1}_{10}(4,4) \quad \text{and}$$

$$\left.\begin{array}{l} BR^{1}_{10}(3,4) < BR^{1}_{10}(4,4) \\ BR^{1}_{10}(4,3) < BR^{1}_{10}(4,4) \end{array}\right\} BR^{1}_{10}(3,3) < BR^{1}_{10}(4,4) \ .$$

There are also cases when both the change in x and the change in y independently have the effect of increasing the function value, and such changes combined can either increase it or decrease it. Take for example the point $(4,3)$. We have both of the following situations:

$$\left.\begin{array}{l} BR^{1}_{10}(3,3) > BR^{1}_{10}(4,3) \\ BR^{1}_{10}(4,4.5) > BR^{1}_{10}(4,3) \end{array}\right\} BR^{1}_{10}(3,4.5) < BR^{1}_{10}(4,3) \quad \text{and}$$

$$\left.\begin{array}{l} BR^{1}_{10}(3.5,3) > BR^{1}_{10}(4,3) \\ BR^{1}_{10}(4,3.5) > BR^{1}_{10}(4,3) \end{array}\right\} BR^{1}_{10}(3.5,3.5) > BR^{1}_{10}(4,3)$$

An infinite number of points with such behaviors exist. The same is true for $BR^{0.75}_{10}$. As an example,

$$\left.\begin{array}{l} BR^{0.75}_{10}(5.5,6) > BR^{0.75}_{10}(7,6) \\ BR^{0.75}_{10}(7,6.5) > BR^{0.75}_{10}(7,6) \end{array}\right\} BR^{0.75}_{10}(5.5,6.5) < BR^{0.75}_{10}(7,6) \quad \text{and}$$

$$\left.\begin{array}{l} BR^{0.75}_{10}(6.5,6) > BR^{0.75}_{10}(7,6) \\ BR^{0.75}_{10}(7,6.5) > BR^{0.75}_{10}(7,6) \end{array}\right\} BR^{0.75}_{10}(6.5,6.5) > BR^{0.75}_{10}(7,6) \ . \text{ Additionally,}$$

$$\left.\begin{array}{l} BR^{0.75}_{10}(7,6) < BR^{0.75}_{10}(5.5,6) \\ BR^{0.75}_{10}(5.5,6.5) < BR^{0.75}_{10}(5.5,6) \end{array}\right\} BR^{0.75}_{10}(7,6.5) > BR^{0.75}_{10}(5.5,6) \quad \text{and}$$

$$\left.\begin{array}{l} BR^{0.75}_{10}(7,4) < BR^{0.75}_{10}(6,4) \\ BR^{0.75}_{10}(6,3) < BR^{0.75}_{10}(6,4) \end{array}\right\} BR^{0.75}_{10}(7,3) < BR^{0.75}_{10}(6,4) \ .$$

An infinite number of points with such behaviors exist for this function as well.

The experiments reported in this section used as a basis one of seven variations of the simple CoEA described in the previous chapter (population size 10 with elitism) and varied the collaboration method. Five such methods were used: single-best (1B), one best plus one random (1B1R), one random (1R), two random (2R) and five random (5R) collaborators. Thus, both the sample size and fitness bias were varied.

The random collaborators were selected per evaluation (i.e. per individual) rather than per generation. When more than one collaborator is used per individual, the fitness is the

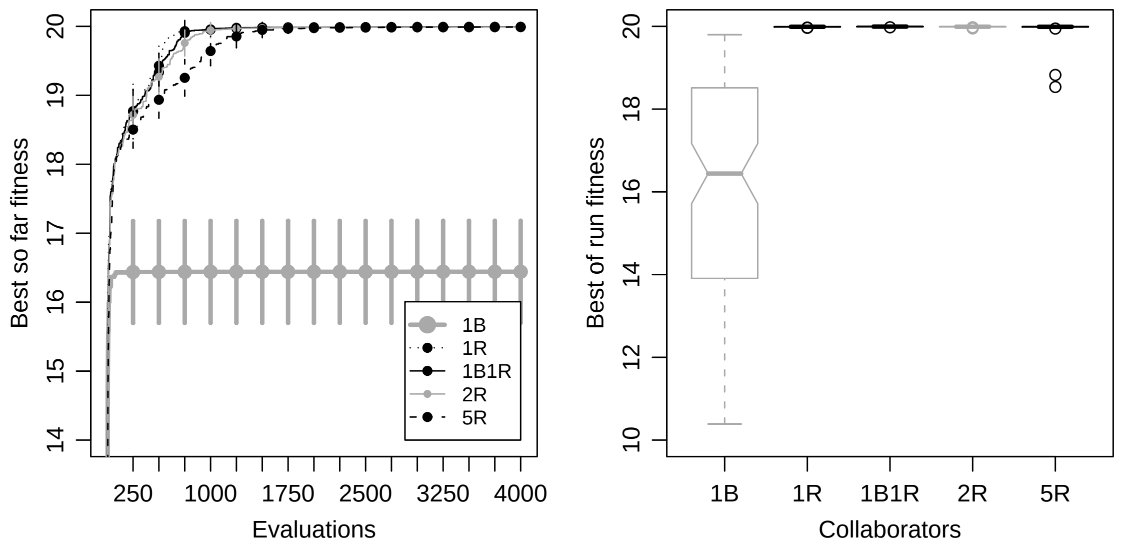


Figure 4.1: Best-so-far (left) and best-of-run (right) fitness statistics for the different **collaboration methods** on the $\boldsymbol{BR^{1}_{10}}$ function. Maximization problem – bigger is better.

best of the obtained values. Selection of random collaborators is done with replacement (even when choosing collaborators for the same individual). All variants had a fixed budget of 4000 function evaluations, therefore using more collaborators per individual meant fewer generations. Since the population size was set to 10 and the communication flow was sequential, this translated into 400 generations for methods 1B and 1R, 200 generations for 1B1R and 2R and 80 generations for 5R.

The results are shown in Figures 4.1 and 4.2. As in the previous chapter, these display both best-so-far fitness curves and best-of-run fitness boxplots, summarized over 100 runs. As these were maximization problems, bigger values are better.

Clearly, the single-best collaboration method has quite opposite performance on the two functions. For BR^{1}_{10}, it is largely outperformed by all other methods. For $BR^{0.75}_{10}$, all methods are in the same ballpark, yet some strongly statistically significant differences[3] can still be observed. Methods using the best as a collaborator (1B and 1B1R) outperform random-only based methods (1R, 2R and 5R) with p-values in the order of 10^{-15}. The differences in medians are in the order of 5×10^{-3}. The single-best collaborator method is

[3]The Wilcoxon / Mann-Whitney test was used.

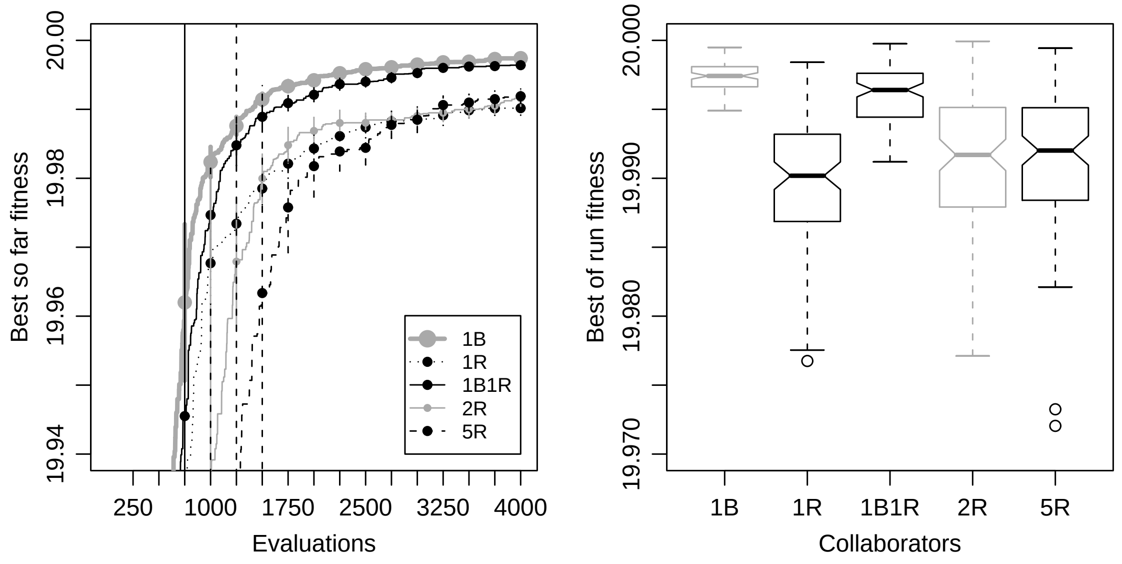


Figure 4.2: Best-so-far (left) and best-of-run (right) fitness statistics for the different **collaboration methods** on the $\boldsymbol{BR_{10}^{0.75}}$ function. Maximization problem – bigger is better.

better than the one best + one random at a p-value of 5×10^{-6} with a median difference of 10^{-3}. The order among methods other than the single best is the same for BR_{10}^{1} and $BR_{10}^{0.75}$, namely the one best + one random method outperforms the random-only based ones. The p-values are weaker for BR_{10}^{1} (between 0.002497 and 5.868×10^{-7}).

For two problems with the same type of contradictory cross-population epistasis, the single-best collaboration method is in one case the best method and in the other case the worst method by large. The recurring hypothesis in this chapter and the whole dissertation is that the differences in performance are due to the different layout of the best-response curves. The following section shows the dynamics analysis performed, which confirms this hypothesis, thus explaining the results.

4.1.3 Dynamics Analysis

Figures 4.3 through 4.8 show typical runs[4] for the two functions and the five different collaboration methods. The plots display best-of-generation trajectories, as described in

[4]As before, plots for all 100 runs were generated and visually investigated. Figure 4.3 displays three runs, all other plots display a single run.

section 3.2, overlaid on the best-response curves. One difference is that each point on a trajectory represents the best individual of a generation (and corresponding population) coupled with the collaborator from the other population to which it owes its fitness. This may or *may not* be the best individual of that other population.

As a consequence, once a random collaborator is used, the trajectory is no longer forced to move only horizontally and vertically (as is the case with the single-best method), but it can take any direction. Additionally, when using only random collaborators, elitism no longer guarantees ever increasing fitness.[5] This means that the end of the run is not necessarily the best of the run as well, as can be seen on the corresponding zoomed-in plots.

Since the total number of evaluations per run was kept constant, more collaborators per individual meant fewer generations, and this is reflected in the varying number of small dots in the plots.

As we have already seen in the previous chapter, for single-best collaboration the dynamics of best individuals are strongly influenced by the best-response curves. We can now see from Figures 4.4, 4.5, 4.7 and 4.8 that this is true for the other evaluation schemes as well.

Overlapping best responses – BR^1_{10}.

As we know from the previous chapter, the BR^1_{10} function has fully overlapped best responses (the main diagonal of the space) and this causes problems to the combination of elitism and single-best collaboration. The best-of-generation trajectory of such a CoEA is constrained to move only horizontally and vertically *and* only to better points. The effect is that fairly quickly after start-up, the trajectory converges to some diagonal point close to the starting location. The starting point of a run puts a bound on the highest fitness that run can achieve. Unless the algorithm starts close to the global optimum, it will not get close to it. The best of a run is thus highly dependent on the start of the run. This explains the high variance in performance over multiple runs and the generally poor results of the single-best collaboration strategy, as shown by the corresponding boxplot in Figure

[5]Even for the one best + one random method, elitism may not guarantee monotonic best-of-generation fitness, depending on some implementation details. The implementation used for the experiments in this thesis *does* have such a guarantee.

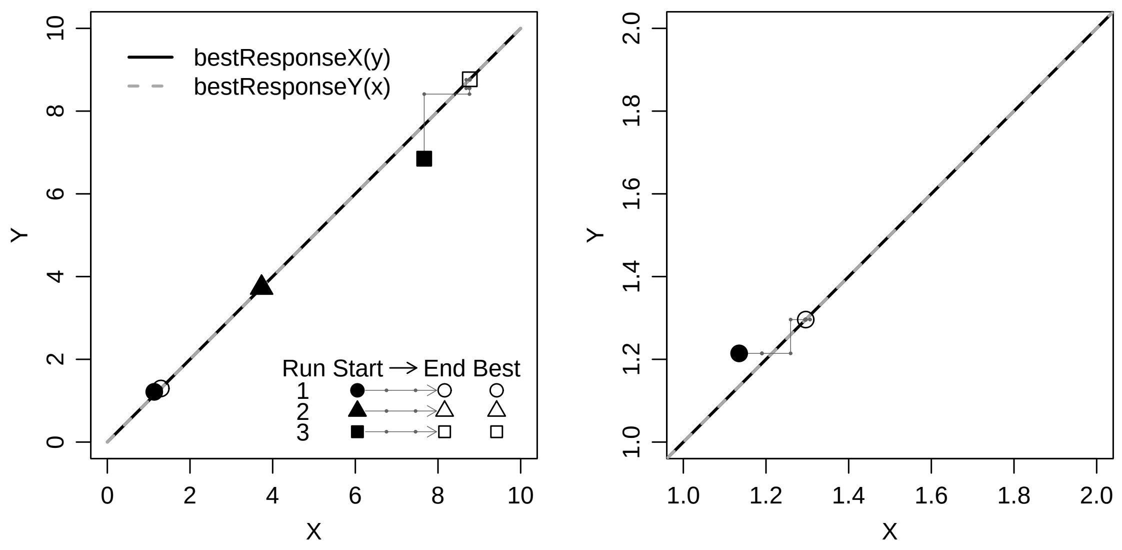


Figure 4.3: $\boldsymbol{BR^{1}_{10}}$. Best-of-generation trajectories for the **single-best collaboration** method. Three runs. Left: whole search-space. Right: zoom in.

4.1.

Using a random collaborator has the effect of preventing the trajectory from getting "stuck" on overlap points, by allowing jumps in any direction. The trajectory can (and does) move diagonally towards areas of high fitness (upper right corner), regardless of where it started. This can be seen on the left hand side plots of Figures 4.4 and 4.5. This explains while all methods using at least one random collaborator heavily outperform the single-best collaboration method.

Investigating the zoomed-in plots helps explain why combining one best and one random gives the best results on BR^{1}_{10}. We can see that the random-only methods have highly "explorative" trajectories. Jumps are taken both towards higher and towards lower fitness. By contrast, 1B1R combines the focus on best responses given by the best collaborator with the exploration given by the random collaborator to avoid both getting stuck on the diagonal and wandering around inefficiently.

Increasing the number of random collaborators from one to five has the effect of making the trajectories less explorative and more exploitative. Due to the fixed budget, there are fewer steps, but each step has higher chances of finding a better point. These two effects

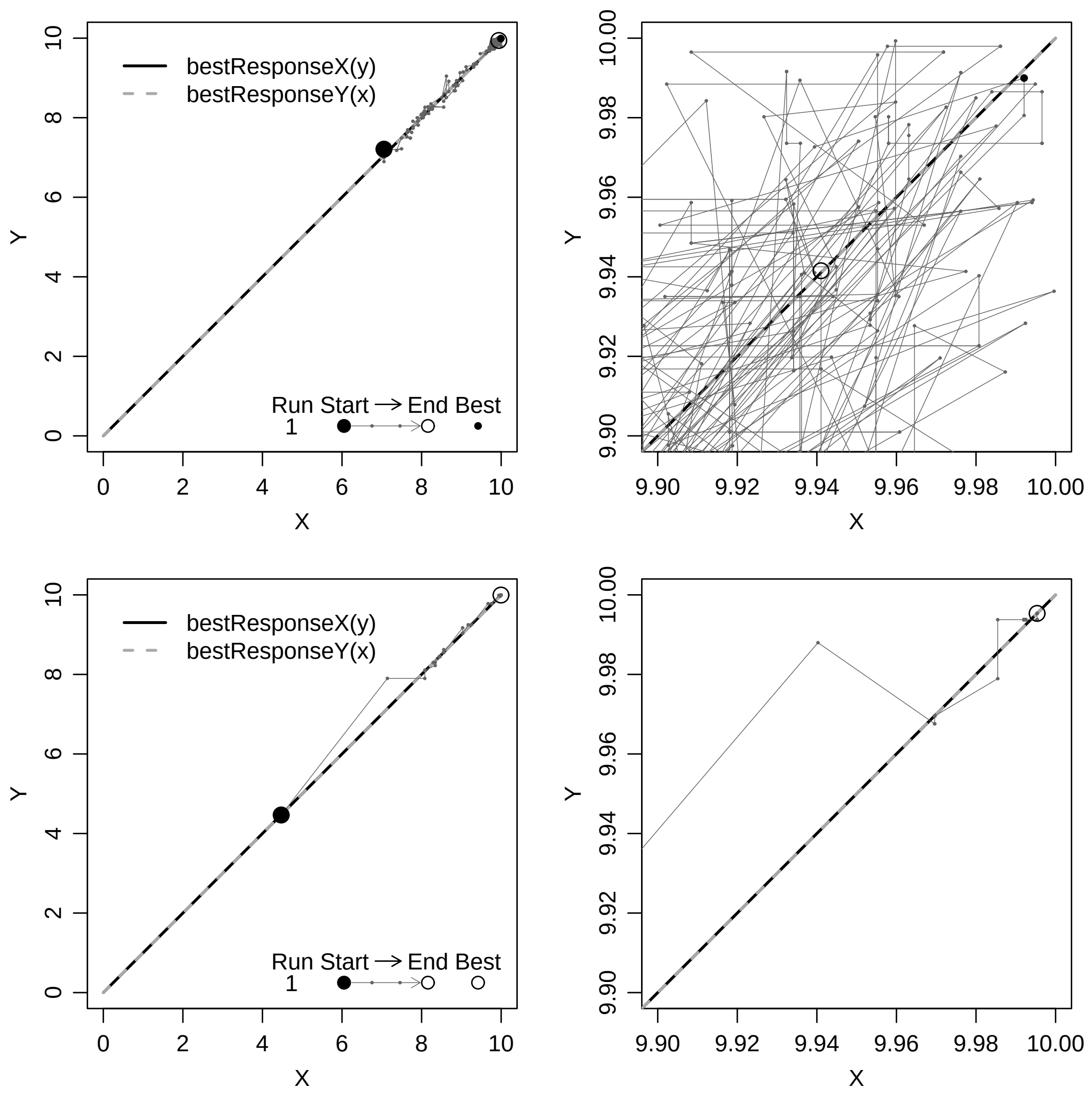


Figure 4.4: ***BR***$^{\boldsymbol{1}}_{\boldsymbol{10}}$. Best-of-generation trajectories for the **single-random** (top) and the **one best + one random** (bottom) **collaboration methods**. One run each. Left: whole search-space. Right: zoom in.

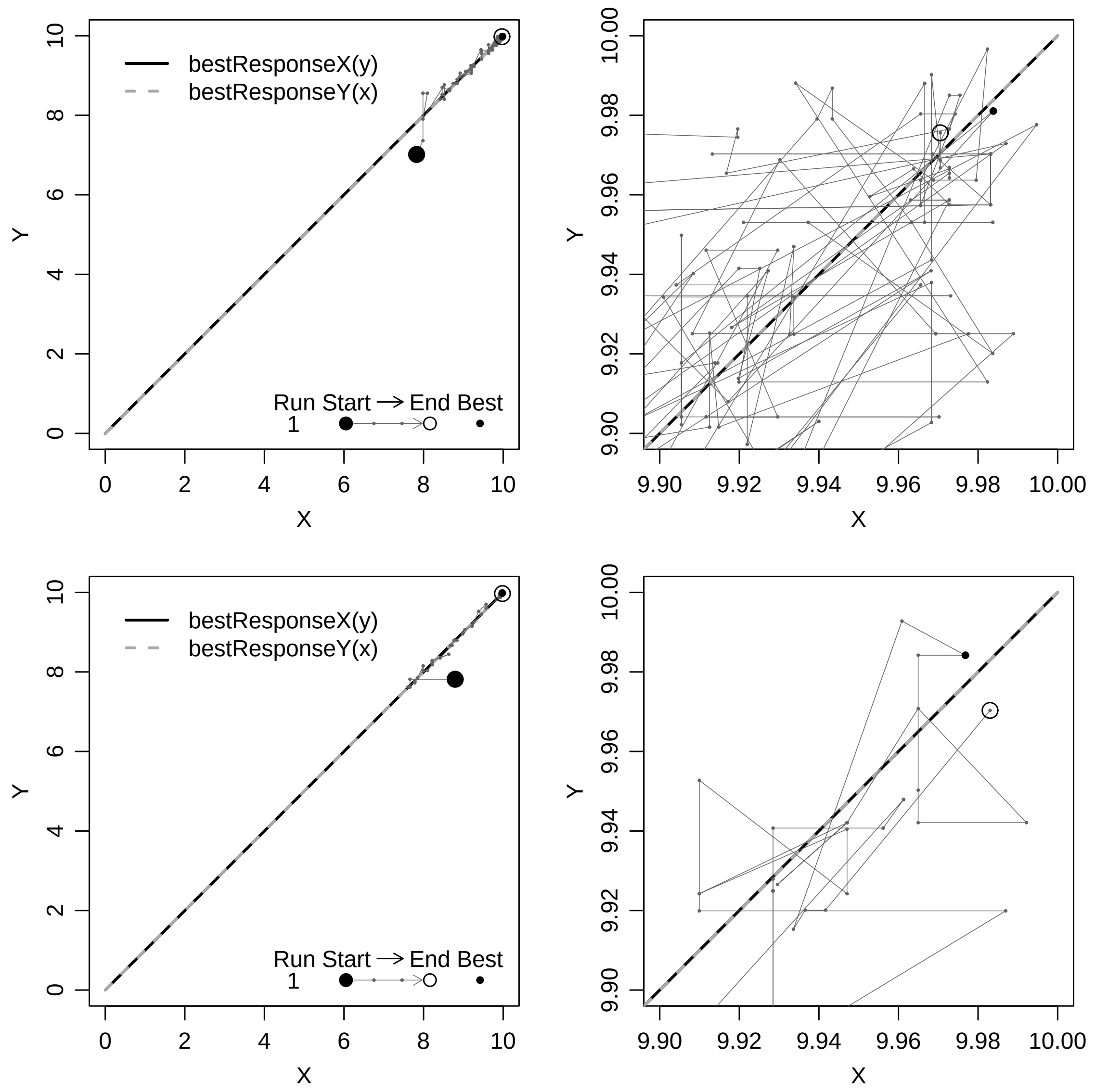


Figure 4.5: $\boldsymbol{BR^{1}_{10}}$. Best-of-generation trajectories for the **two random** (top) and the **five random** (bottom) **collaboration methods**. One run each. Left: whole search-space. Right: zoom in.

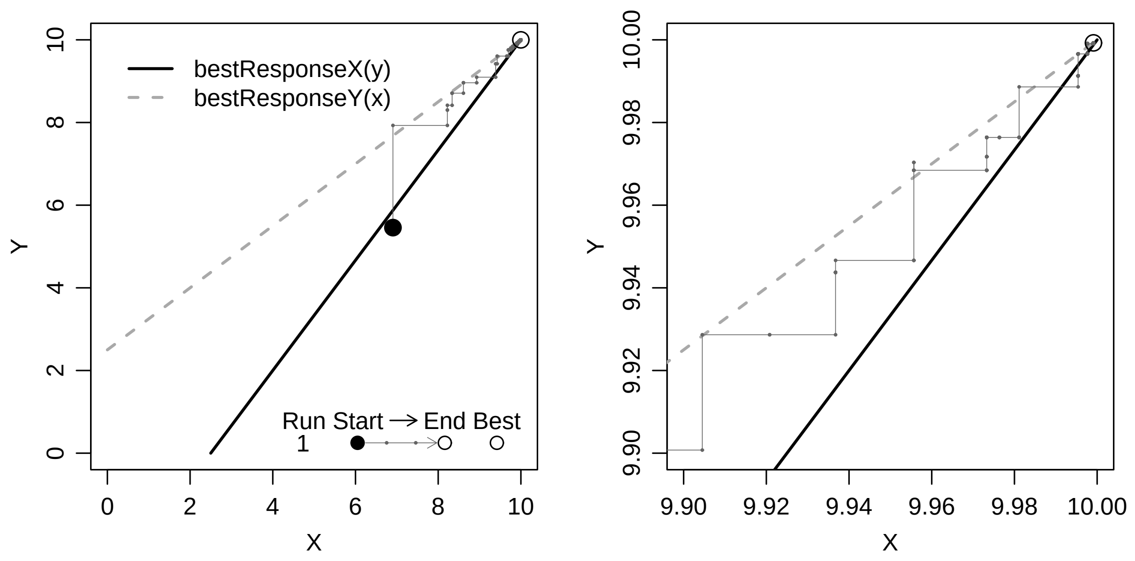


Figure 4.6: $\boldsymbol{BR_{10}^{0.75}}$. Best-of-generation trajectories for the **single-best collaboration** method. Three runs. Left: whole search-space. Right: zoom in.

balance out and the best-of-run fitness distributions are not statistically distinguishable.

Non-overlapping best responses – $\boldsymbol{BR_{10}^{0.75}}$.

Performing the same type of analysis for the $BR_{10}^{0.75}$ function explains why in this case the single-best collaboration method has the best performance rather than the worst. For this function, the *bestResponseX* and *bestResponseY* curves differ and they intersect only in the point that is the global optimum of the function. They are shown in Figures 4.6 through 4.8 with thick lines, dashed grey and continuous black, respectively.

In the case of the single-best collaboration method the trajectory alternates vertical steps towards the *bestResponseY* curve with horizontal steps towards the *bestResponseX* curve, as we see in Figure 4.6. This causes it to climb like on a ladder towards the global optimum.

Introducing randomness in the collaboration mechanism causes the algorithm to move away from this focus of following the best responses. In the case of the single-best plus single-random method (bottom row of Figure 4.7), the trajectory may take bigger steps toward high function values, but it does so at the cost of cutting in half the number of

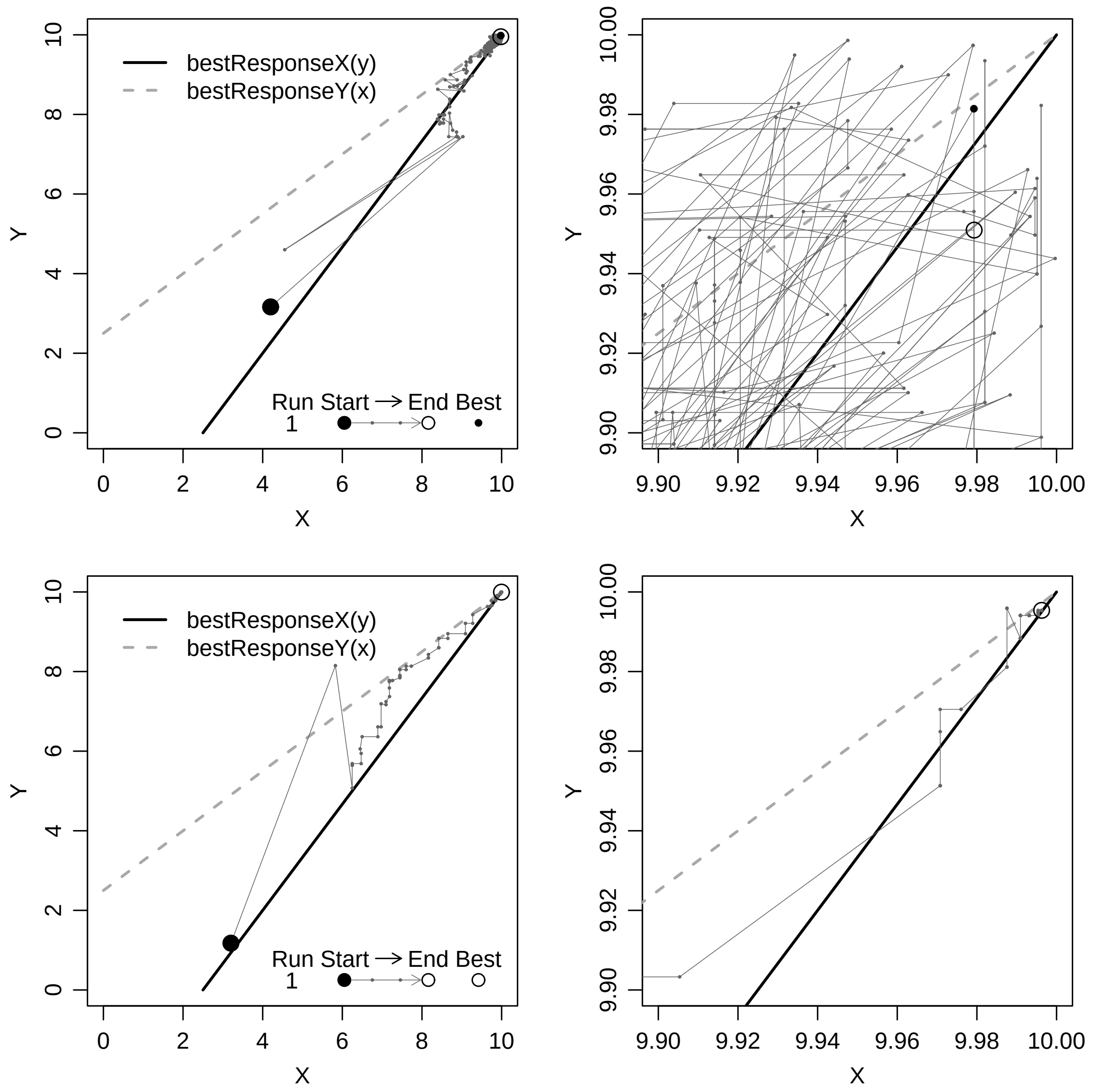


Figure 4.7: $\boldsymbol{BR_{10}^{0.75}}$. Best-of-generation trajectories for the **single-random** (top) and the **one best + one random** (bottom) **collaboration methods**. One run each. Left: whole search-space. Right: zoom in.

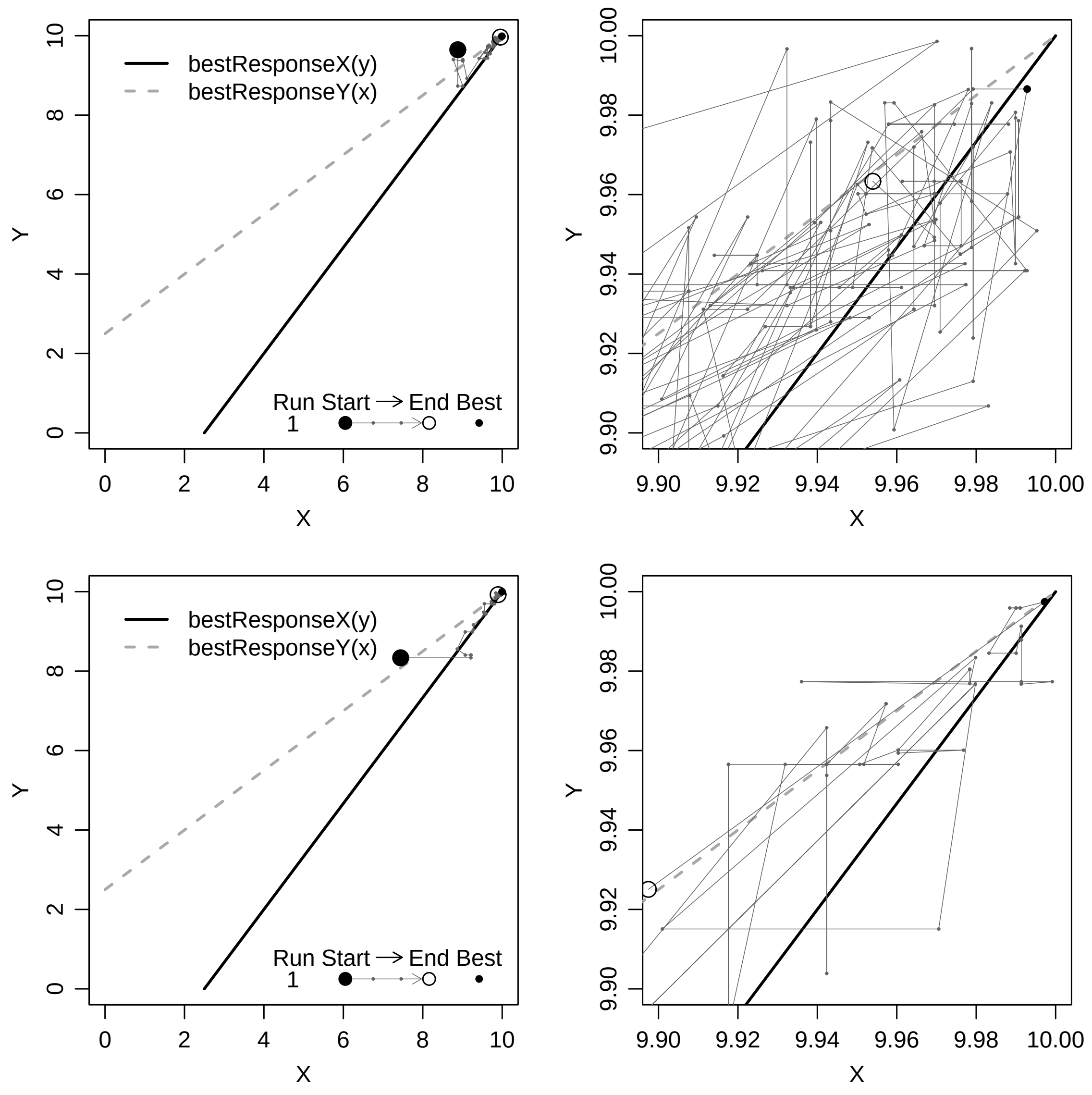


Figure 4.8: $\boldsymbol{BR_{10}^{0.75}}$. Best-of-generation trajectories for the **two random** (top) and the **five random** (bottom) **collaboration methods**. One run each. Left: whole search-space. Right: zoom in.

steps, thus it does not get quite as close to the optimum as using the single best only.

Using a single random collaborator (top row of Figure 4.7) generates again a lot of exploration at the cost of almost no exploitation. For this function however, due to the nature of its best-response curves, exploitation alone is a guaranteed way of reaching the optimum, while exploration alone will get there only by luck. Increasing the number of random collaborators (Figure 4.8) reduces exploration, but fails to increase exploitation to the level of using the single-best.

The analysis of the dynamics thus explained why we observed the results in Figures 4.1 and 4.2 and it provided additional understanding of how the collaboration methods under investigation work. Specifically, the best-response curves, which are a property of the problem, have a strong impact on the dynamics of the algorithm. Different collaboration methods interact with this property in different ways, affecting optimization performance.

In particular, the overlapping of the best-response curves in more than one point is bound to cause problems for the single-best collaboration method, as it makes the algorithm get stuck quickly in one of these points. Which of the points will attract the trajectory largely depends on the initial configuration, rather than the function value at those points. Thus, for functions that have different values across these overlapping areas, the single-best collaboration method will exhibit poor performance at finding the optimum and methods based on random collaborators should be used.

4.1.4 Predictive Power

We now use the heuristics developed through dynamics analysis in the previous section to predict the effects of collaboration methods on the *rosenrock* and $offAxisQuadratic$ functions. Remember that for these functions the task is minimization. The best-response curves for these functions were shown in Figure 3.9. For both functions, the best responses intersect in a single point, corresponding to the global optimum. For $offAxisQuadratic$ the best responses are lines with an angle similar to that of $BR_{10}^{0.75}$. *rosenbrock*'s best responses are extremely close.

For *rosenbrock*, the CoEA using the single-best collaboration method will soon after

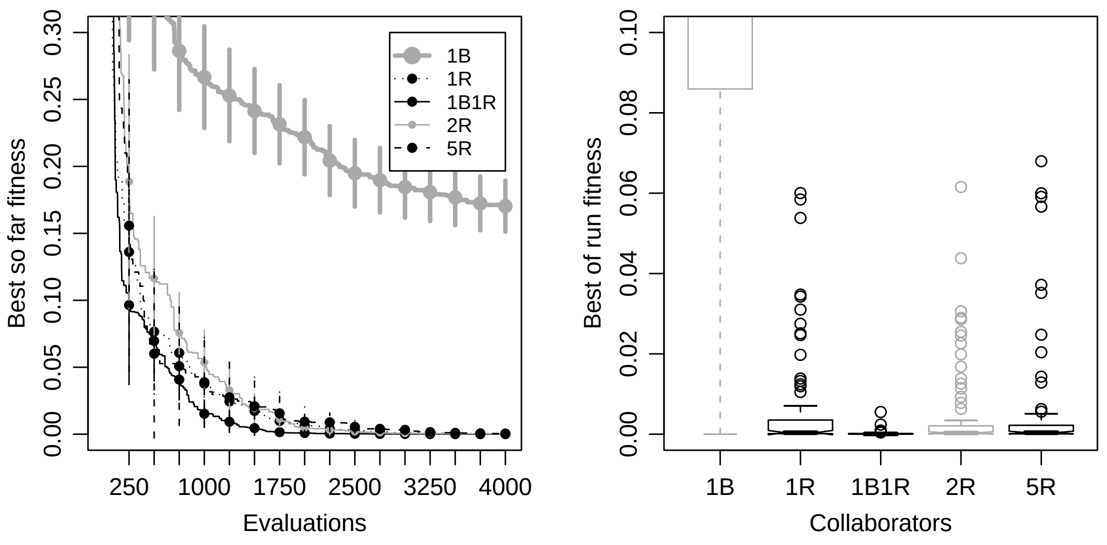


Figure 4.9: Best-so-far (left) and best-of-run (right) fitness statistics for the different **collaboration methods** on the ***rosenbrock*** function. Minimization problem – smaller is better.

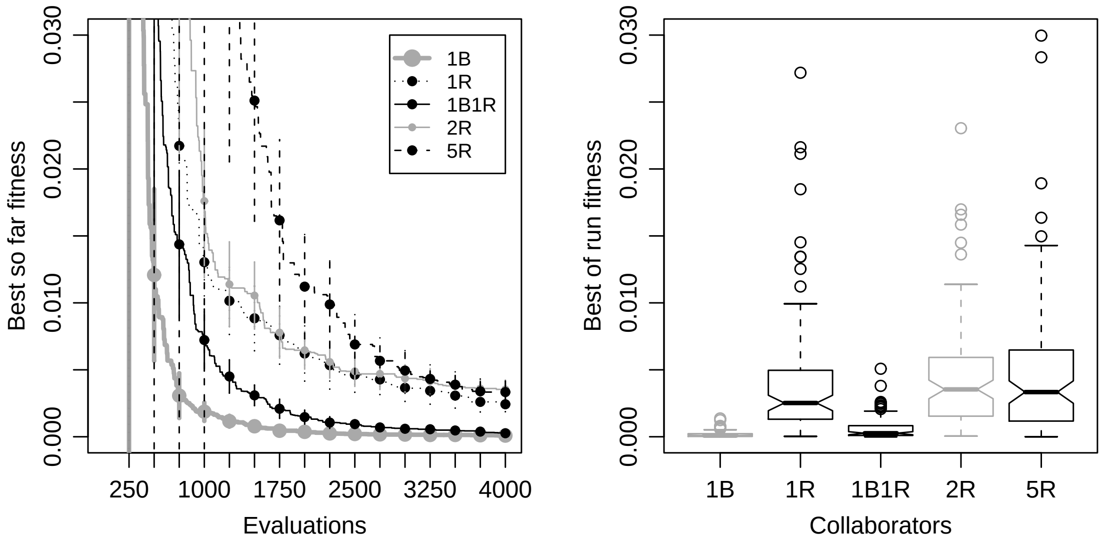


Figure 4.10: Best-so-far (left) and best-of-run (right) fitness statistics for the different **collaboration methods** on the ***offAxisQuadratic*** function. Minimization problem – smaller is better.

start-up get stuck somewhere on the area of near-overlap, the position depending on the start-up position. It will get close to the optimum only by chance (if it starts close to it), therefore we predict poor performance for this case. Introducing a random collaborator is likely to increase exploration and visit several points along the nearly-overlapping region, thus increasing the chances of getting closer to the optimum. We therefore expect these methods to give better results than the single-best, in a manner similar to the BR_{10}^{1} function.

Indeed, by running our CoEA with the same 5 collaboration methods as before, we get the results plotted in Figure 4.9, which confirm our expectations. Note that since these were minimization problems, smaller values are better.

For the $offAxisQuadratic$ function, the trajectory of best individuals for the CoEA using the single-best collaboration method will alternately move horizontally towards the *bestResponseX* curve and vertically towards the *bestResponseY* curve. This will direct the algorithm reliably towards the point of intersection of the best-response curves, thus towards the global optimum. Introducing random collaborators can only disrupt this focus. Exploitation alone will solve the problem, exploration is not needed and in fact can be harmful. We predict the results will be similar to the ones obtained for the $BR_{10}^{0.75}$ function, and Figure 4.10 confirms this.

4.2 Communication Frequency

In order for individuals in one population to interact with individuals from other populations, the populations must have access to one another's contents. This is a matter of communication between populations, and the question that arises is how often this communication should take place. The standard choice is to have the EAs in the different populations be generational and have the populations communicate after each generation. However, one could also allow the populations to evolve independently for longer or shorter, in other words, adjust the *communication frequency.* This section investigates the performance effects of this CoEA-specific parameter and finds them once again dependent on the problems' best responses.

4.2.1 Related Work

Most work in both compositional and test-based setups has used the standard choice of communicating after each generation. Various other choices for this parameter were used mainly in domain-focused applications of CoEAs, e.g. (Bongard and Lipson, 2005) for a test-based problem and (Parker and Blumenthal, 2003; Blumenthal and Parker, 2004) for a compositional problem. In particular, Parker and Blumenthal (2003); Blumenthal and Parker (2004) use a CoEA with a non-uniform evaluation scheme, evaluating individuals one way during communication with other populations and another way when evolving independently. In comparing different parameter choices, in order to keep a fixed budget, changing the communication frequency also changed the number of collaborators used for evaluation at communication. Two sensitivity studies[6] of the communication frequency were performed for robot domains (box-pushing and pursuit-evasion).

No cross-domain systematic study of the effects of communication frequency was previously performed.

[6]Only five runs were performed and averaged for each setting.

4.2.2 Experiments

The Problems

The BR_n^α family was used for this analysis, instantiated with $n = 10$ and $\alpha \in \{0, 0.2, 0.25, 0.4, 0.5, 0.6, 0.75, 0.8, 0.9, 0.94, 0.95, 0.96, 0.98, 0.99, 1\}$. For brevity and clarity, the data presented below is only from 9 of these settings for α, namely 0, 0.2, 0.4, 0.6, 0.75, 0.9, 0.96, 0.98 and 1. However, the results for the omitted values fit well in the trend described by the values showed.

The Algorithm

We call a *communication point* a point in evolutionary time at which the populations communicate. In this section we only look at cases where the time between any two consecutive communication points is the same. We call this period of time an epoch. In the experiments reported here we varied the epoch size as a means to control the frequency of communication (bigger epoch size equals smaller frequency and vice versa). Because we have used a generational EA, we measured the epoch size in generations. For example, for the base-CoopCoEA the epoch size is one.

Using a sequential flow now translates into only one population being active during each epoch, while the other population is frozen. At the end of the epoch (at the communication point), the population that was evolving communicates its best individual to the population that was frozen and then they switch roles (unidirectional communication). The previously frozen will be active during the following epoch and at the end of it, it will report its best individual to the other population. Below is the pseudo-code for an X epoch n generations long:

```
- evaluate the $X$ population using the current $y_{best}$;
- repeat $n$ times:
    - select parents according to determined fitness values;
    - breed;
    - evaluate the new population using the same $y_{best}$;
- determine $x_{best}$ and report it to the other population.
```

For fair comparisons between communication schemes, we kept the number of evaluations constant across experiments. Note that with the above code, for an epoch that is n generations long, the active population will perform $(n+1)*m$ evaluations, where m is the population size. For k epochs, sequential flow will require $(n+1)*m*k$ evaluations, as there is only one population active per epoch.

We have performed experiments with 8 settings for the epoch size, namely 1, 2, 3, 5, 8, 11, 17 and 26 generations. With a fixed budget of 2160 function evaluations, this meant running for 54, 36, 27, 18, 12, 9, 6 and respectively 4 epochs (e.g. for epoch size 3 we have $(3+1)*20*27 = 2160$). For each setting we performed 100 independent runs, 50 of which started with the X population active and 50 with the Y population active. We then repeated all of this for every value of α mentioned above.

The Performance

Figure 4.11 summarizes performance of the 8 epoch size settings for the 9 mentioned α values. It shows boxplots of best-of-run fitness collected over the 100 independent runs. Although BR_n^α has the same range for any α, we have zoomed in on a different fitness interval for each plot in order to be able to see the differences between epoch sizes.

As we increase α from 0 to 1, we observe a gradual change in the performance effects of the epoch size. We start (for $\alpha = 0$) with an upward-only slope for performance as we increase epoch size from 1 to 26. At $\alpha = 0.2$, the performance "curve" is first going upward, then level and then downward. As we further increase α, the upward part of the performance curve is gradually getting shorter and shorter until it vanishes (at about $\alpha = 0.6$), while the downward part is getting more and more pronounced. From there to $\alpha = 1$ we see a more rapid change, with the downward part flattening out through performance decrease for low epoch sizes. A close look at the y axis ranges of the plots will show an overall decrease in performance with increasing α. In other words, BR_{10}^α becomes a harder problem for our CoEA as α increases.

While each particular plot in Figure 4.11 nicely portrays the trend of changing performance with increasing epoch size, it is hard to tell from such a plot whether any two

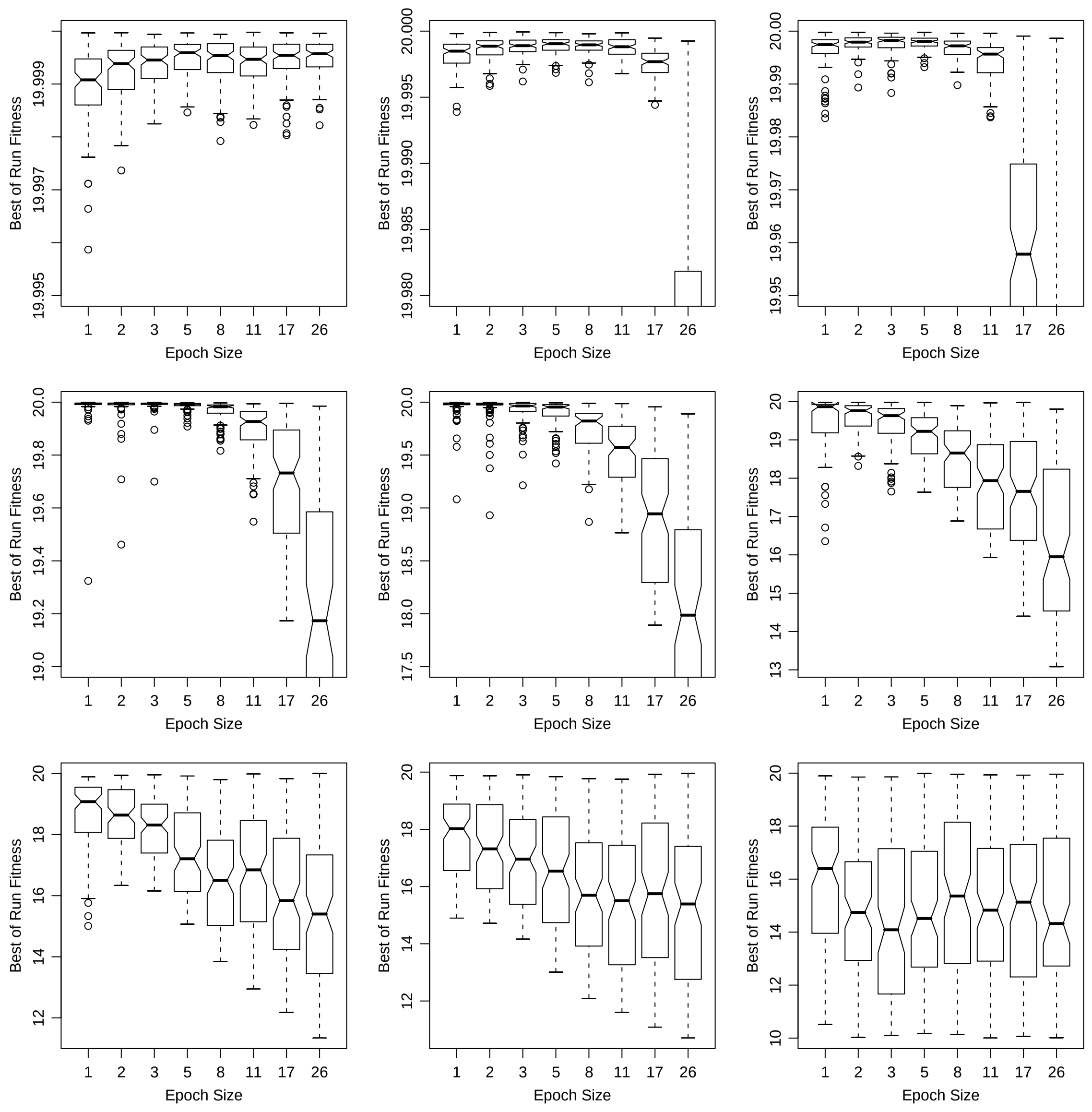


Figure 4.11: Best of run statistics. Maximization problems: bigger is better.

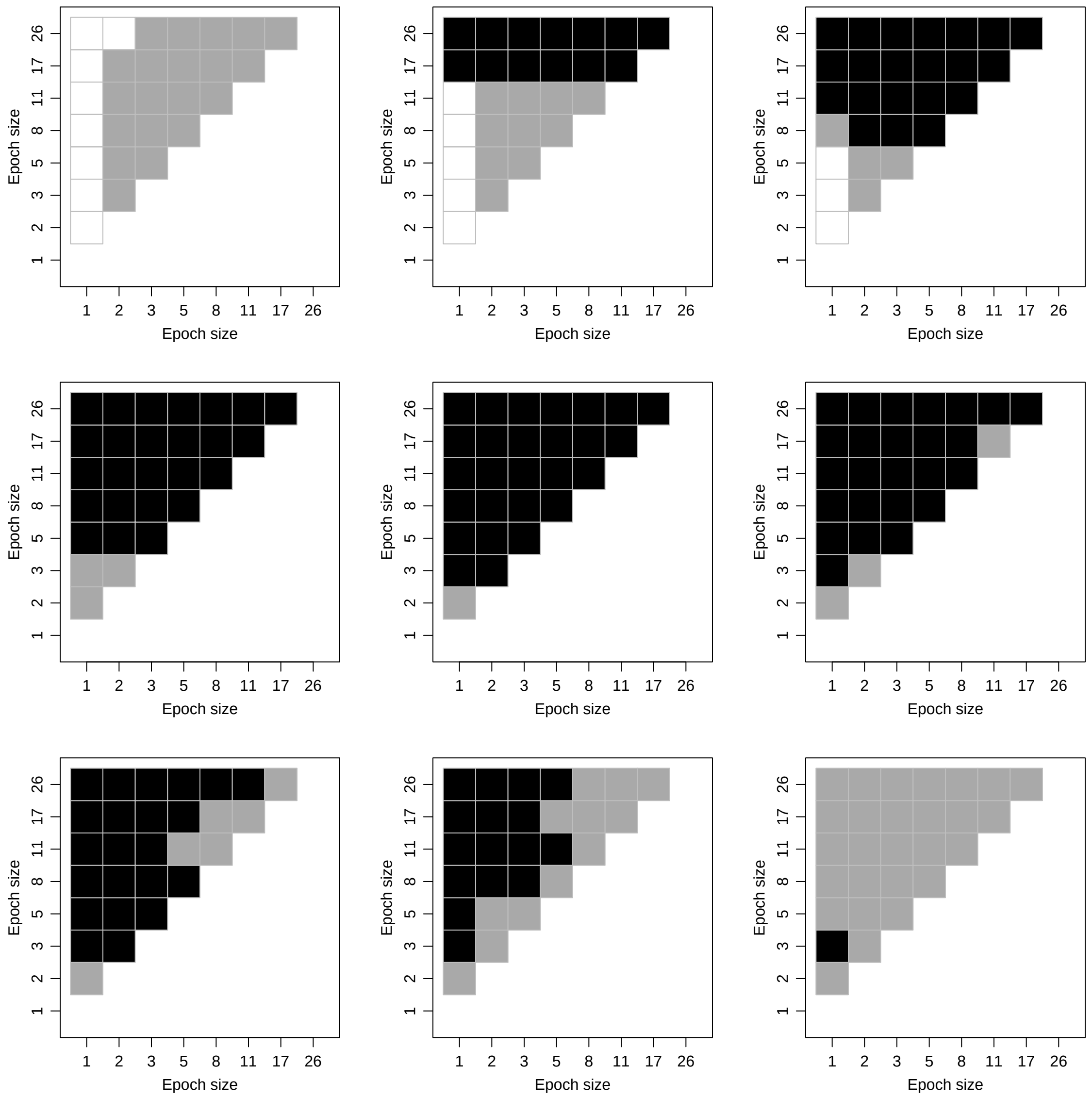


Figure 4.12: $\boldsymbol{BR_{10}^{\alpha}}$. Statistical significance of differences between epoch sizes' performance.

particular epoch sizes generate a statistically significant difference in performance. To achieve that, we use a different plotting technique portrayed in Figure 4.12. Each plot in this figure represents one α value and color-codes the results of all pairwise comparisons of epoch sizes. A square corresponding to epoch sizes es_i and es_j is:

- gray, if we cannot statistically significantly distinguish between the performance of es_i and that of es_j;

- black, if there is a statistically significant difference in performance between es_i and es_j and the smaller epoch size ($min(es_i, es_j)$) performs better; and

- white, if there is a statistically significant difference in performance between es_i and es_j and the bigger epoch size performs better (i.e., the smaller epoch size performs worse).

This definition is symmetrical, since $min(es_i,\ es_j) = min(\ es_j, es_i)$, therefore we only display the upper left triangle. To test statistically significant difference of medians we use the Wilcoxon test (Wilcoxon, 1945) combined with Hochberg's sharper Bonferroni procedure for multiple tests of significance (Hochberg, 1988). The total confidence for each image is at least 95%.

The image for $\alpha = 0$, for example, tells us that epoch size 1 performs worse than any other. It also tells us that epoch sizes 2 and 3 perform worse than 17, but nothing else can be distinguished. Thus, there is a benefit in increasing epoch size from 1 to 2, but to get yet another boost in performance, we would have to increase the epoch size all the way up to 17. For $\alpha = 0.2$ the black squares in the image tell us that epoch size 26 performs worse than anything else and epoch size 17 performs worse than 3 and 8. And while we cannot distinguish between 3, 5 and 8 due to gray, nor between 1 and 2, the white squares tell us that any of 3, 5 and 8 perform better than 1, and 8 also performs better than 2.

Making a parallel with Figure 4.11, white on the left tells us there is an upward part in the performance slope, gray triangles above the diagonal denote a flat part and black on the top shows the existence of a downward part. Reading Figure 4.12 along increasing α, we see that we start with some white on the left and a lot of gray, then the white and the gray areas start diminishing, as the black moves in from the top; the white finally disappears, after which the gray area starts growing again and the black area shrinks until it disappears as well. If we use the above "translation", we discover the same message as conveyed by

Figure 4.11, but now it is backed up by statistical significance.

Our hypothesis is that these performance effects of the communication frequency are due once again to the best-response curves. The dynamics analysis performed in the following section confirms this hypothesis and sheds light on the observed phenomena.

4.2.3 Dynamics Analysis

We perform an analysis similar to the previous ones in order to investigate the performance effects of communication schemes. However, in this case we will be concerned with *best-of-epoch individuals* and their trajectories through the search space. We construct these trajectories in a similar manner to the best-of-generation trajectories. At the end of an epoch during which the X population was active, we plot the best x individual in combination with the y individual used for its evaluation. In this case, due to the single best collaboration scheme, that y is the currently known best individual of the opposite population. At the end of the next epoch, during which the Y population will be active, we plot the new best y coupled with the previous best x. We do this for every epoch and connect the points chronologically. As before, due to the single-best collaboration scheme the trajectory obtained contains only vertical lines (when connecting an X epoch with the following Y epoch) and horizontal lines (when connecting an Y epoch with the following X epoch).

We then combine best-response curves with best-of-epoch trajectories. Figure 4.13 shows examples of best-of-epoch trajectories for individual runs, superimposed on best-response curves. It is immediately apparent that the trajectories are highly influenced by the best-response curves.

The previous chapter and the previous section in this chapter showed that optimization performance is tightly correlated with the three factors below, the first one being problem dependent and the latter two algorithm dependent:

- the relative positions of the best-response curves (mainly whether or not they overlap);

- the accuracy with which the best-of-generation trajectories follow the best-response curves; and

- the length of the trajectories (i.e. the number of communication points).

In the case of $\alpha = 1$, high accuracy in following the best-response curves is bad for

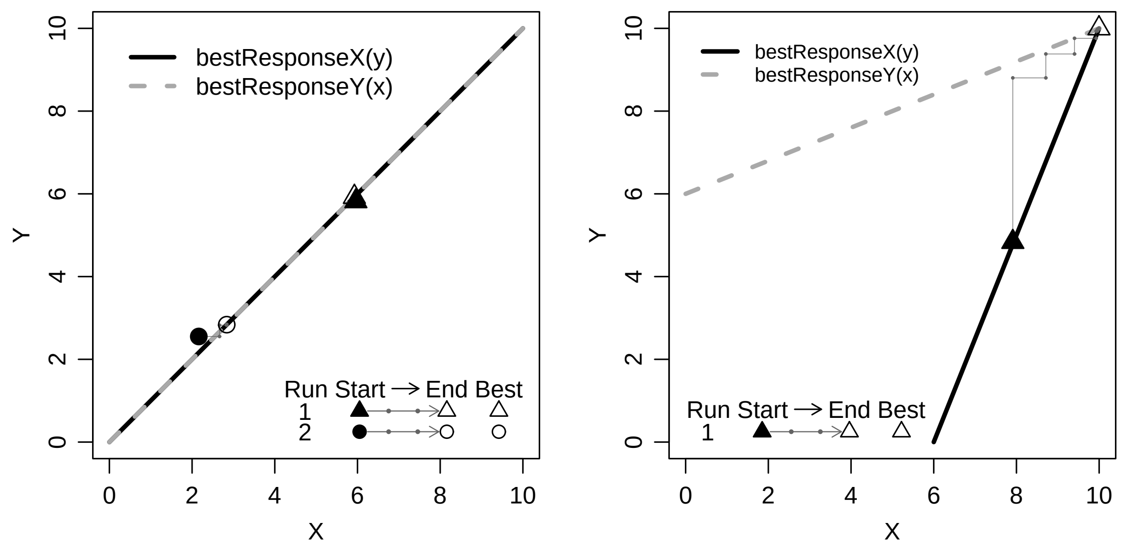


Figure 4.13: Examples of best-of-epoch trajectories and best-response curves. Left: $\alpha = 1$, epoch size 1. Right: $\alpha = 0.4$, epoch size = 3.

performance, since it will cause the trajectory to quickly get stuck in a point on the overlap line (as can be seen in the top of Figure 4.13), which may or may not be close to the optimum (depending on the starting position). For $\alpha \in (0, 1)$, high accuracy will cause the trajectory to climb like on a ladder towards the optimum. However, it constrains the size of the trajectory's steps to the distance between the best-response curves. The closer these are, the smaller the highly accurate steps will be. Low accuracy allows for jumps larger than the distance between the best-response curves, although smaller jumps may occasionally occur as well. Thus, high accuracy is beneficial when the best-response curves are at a big angle, but it starts being detrimental as the angle between them gets smaller.

When the trajectory is not stuck, a bigger length (more steps) will give it more time to get closer to the optimum. However, with a fixed budget, more steps usually imply smaller per-step accuracy. When the trajectory is stuck, more steps will not help (but won't hurt either).

Clearly, since trajectory length is measured in communication points, increasing the epoch size decreases trajectory length. Intuitively, increasing the epoch size should have the effect of increasing trajectory accuracy. One way of testing this is to visually inspect

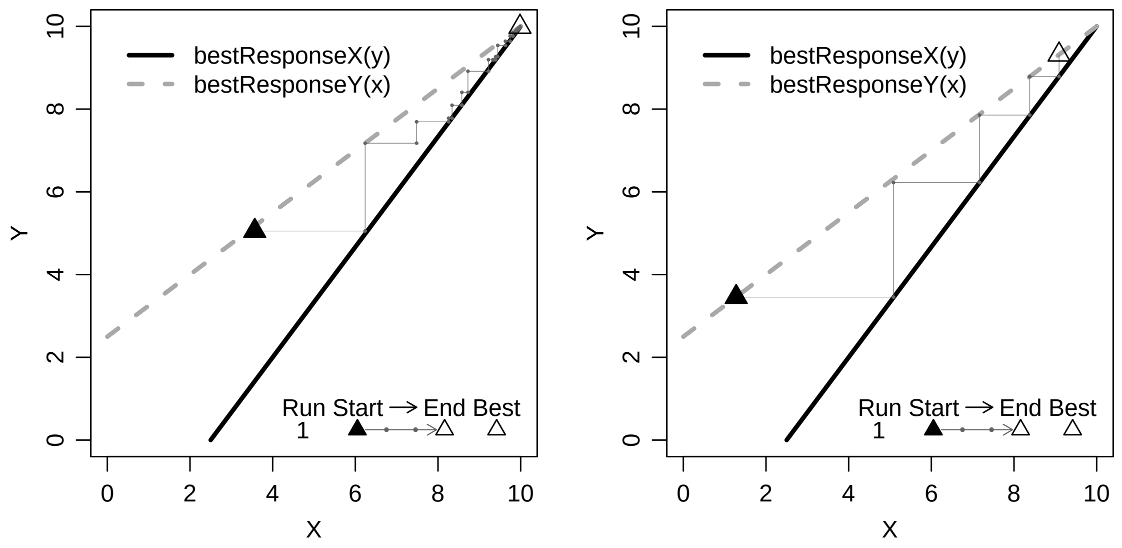


Figure 4.14: Higher epoch size implies fewer communication points and also appears to have the effect of increasing accuracy in following the best-response curves. $\alpha = 0.75$. Left: epoch size 1. Right: epoch size 11.

trajectories. Figure 4.14, portraying epoch sizes 1 and 11 for $\alpha = 0.75$ seems to suggest that is the case. However, we would like a quantitative way of testing this hypothesis.

For that purpose, we define two metrics that compute distance from the best-response curves:

$brDistX(x, y) = |x - bestResponseX(y)|$ and

$brDistY(x, y) = |y - bestResponseY(x)|$.

We compute $brDistX$ at communication points marking the end of an X epoch and $brDistY$ at communication points marking the end of an Y epoch. This gives us a measure of the accuracy of trajectories. We use it to compare the accuracy induced by different epoch sizes.

Figure 4.15 shows statistics of $brDistY$ ($brDistX$ behaves similarly). Each plot portrays for a certain α the accuracies for epoch sizes 1, 5 and 17. At each communication point marking the end of an Y epoch, we plot the mean over 50 runs[7] of $brDistY$ for that

[7]In this case the ones starting with an X generation. Similar plots are obtained when averaging over the 50 runs that start with a Y generation

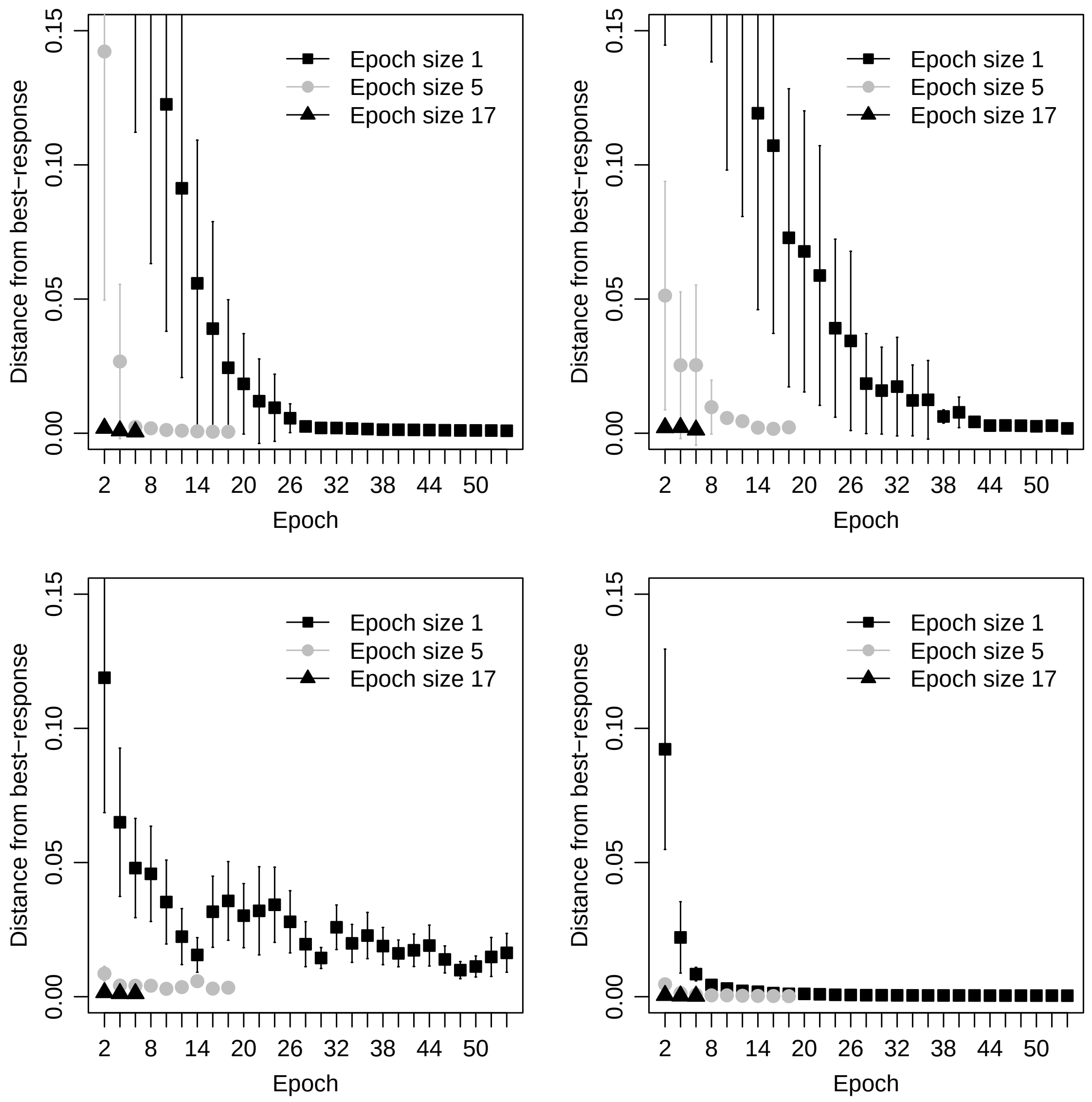


Figure 4.15: $\boldsymbol{BR^{\alpha}_{10}}$. **Accuracy with which best-of-epoch trajectories follow the best-response curves**. $|y - bestResponseY(x)|$ computed for best-of-Y-epoch points (x, y). Top row, left to right: $\alpha = 0, \alpha = 0.6$. Bottom row, left to right: $\alpha = 0.98, \alpha = 1$.

communication point, together with the 95% confidence interval for that mean. These plots confirm our hypothesis that increasing epoch size increases the accuracy of the best-of-epoch trajectories.

Combining this new knowledge with the previous heuristics about how best-response curves, trajectory accuracy and trajectory length affect performance, we can now explain the results displayed by Figures 4.11 and 4.12.

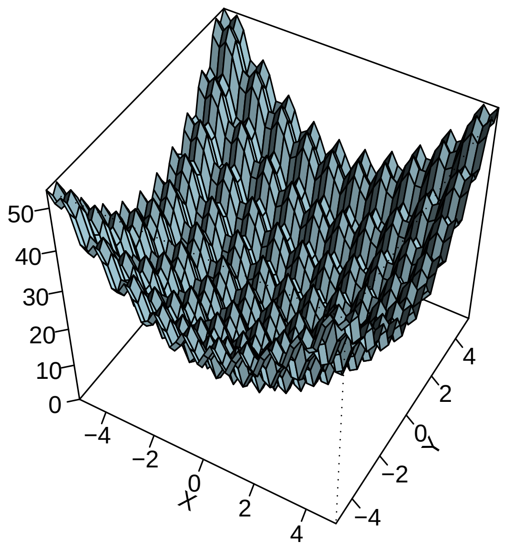
At $\alpha = 0$, as the best-response curves are perpendicular, a deterministic system exactly following them would reach the optimum in just two steps. Even with an epoch size of 26 generations, the trajectories are allowed 4 steps. Thus the trajectory length plays a less important role and it is accuracy that brings in performance benefits. As we increase α, we decrease the angle between the best-response curves, and more steps are needed in order to get close to the optimum, therefore trajectory length becomes increasingly more important. As long as increasing the epoch size increases accuracy while still keeping the trajectory length above what is needed to get close to the optimum, we see improvements in performance (the upward part of the slope). Then there is a range in epoch sizes for which accuracy and trajectory length counter-balance each other and we see a flat performance trend. After that, the disadvantage of having a short trajectory overcomes the benefits of high accuracy and we see diminishing performance.

As the best-response curves get closer, we transition into the phase where high accuracy is detrimental. Thus, increasing the epoch size becomes bad for performance both through high accuracy and through short trajectory length.

At $\alpha = 1$, even the accuracy of epoch size 1 is extremely high (as can be seen in the bottom right plot of Figure 4.15), and so the trajectory already gets stuck. The best-of-run essentially has the same distribution as the best of the first epoch, namely uniform over the interval [10, 20]. Increasing epoch size can't make things worse than they already are.

4.2.4 Predictive Power

We now test to see whether what we have learned on the BR_n^α family can be applied to other functions. We use three functions: the previously introduced $offAxisQuadratic$ and $rosenbrock$ and a third called $rastrigin$ and defined as:



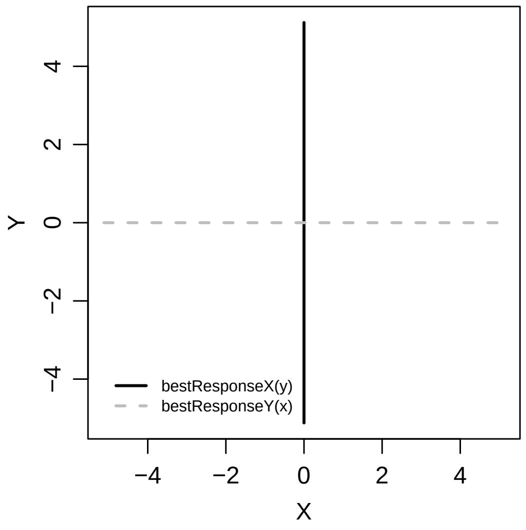


Figure 4.16: ***rastrigin***. Three-dimensional view and best responses.

$$rastrigin(x, y) = 6 + x^2 + y^2 - 3cos(2\pi x) - 3cos(2\pi y),$$
$$x, y \in [-5.12, 5.12].$$

Figure 4.16 shows its three-dimensional surface and its best responses. These are similar to the best responses of BR^{0}_{10}, in that they are parallel to the axes (thus perpendicular to one another) and intersect only at the global minimum, $(0, 0)$. The best-response curves for $offAxisQuadratic$ and $rosenbrock$ were plotted in Figure 3.9.

Based on the similarities with functions from the BR^{α}_{n} family, we expect the following effects of increasing the epoch size within a fixed budget:

- $rastrigin$: performance increases with the epoch size;

- $offAxisQuadratic$: undistinguishable performance for the first few epoch sizes and then decreasing performance;

- $rosenbrock$: similar performance for all epoch sizes, with just a few cases when higher epoch size performs worse.

We performed the same experiments as for BR^{α}_{10}, with the only exception that we adjusted the sigma of the Gaussian mutation according to the size of these new domains. The experiments confirmed our expectations, as can be seen from Figure 4.17.

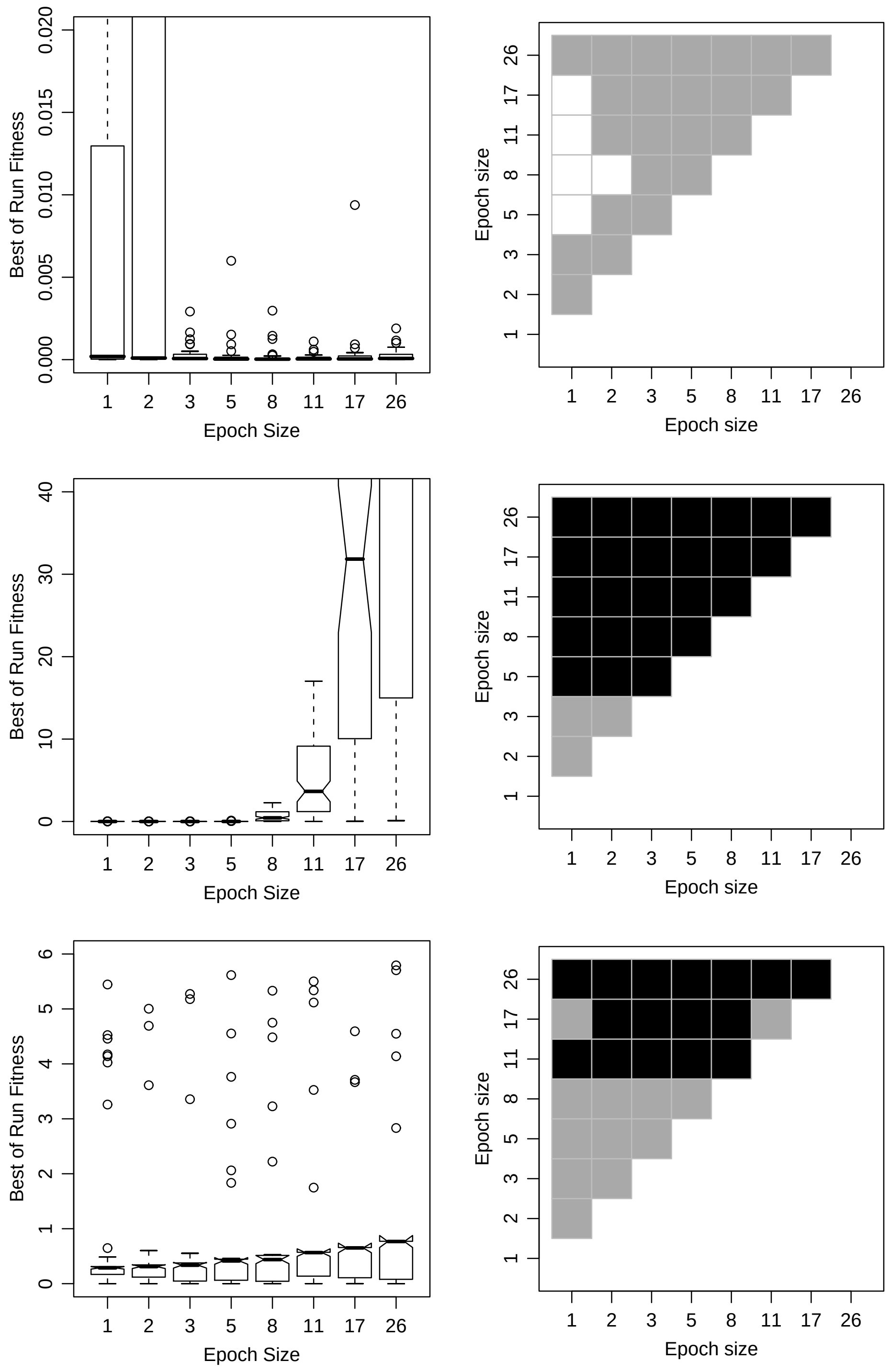


Figure 4.17: Left column: best of run statistics. Minimization: smaller is better. Right column: statistical significance for pairwise comparisons of epoch sizes' performance. Top to bottom, per line: *rastrigin*, *offAxisQuadratic*, *rosenbrock*.

4.3 Communication Flow: Sequential versus Parallel

The parameter under investigation in this section is *communication flow* (also referred to as update timing Wiegand (2004)), which determines the order in which the populations of a CoEA are processed and updated, relative to inter-population communication events. There are two main types of communication flow: *sequential* and *parallel*. In the sequential case the populations are processed in serial order and at any point in time only one population is active. For evaluation purposes, collaborators are chosen from the current state of the other populations. Populations processed and updated earlier will affect populations processed later. In the parallel case the populations are processed in parallel, at any point in time all of them are active[8]. Collaborators are chosen from the saved states of the other populations.

4.3.1 Related Work

A parallel CoEA may be setup to run on multiple processors in order to decrease computation time. Therefore, we would like to know how the optimization performance is affected by the decision of switching from sequential to parallel communication flow. A previous study on this matter was performed by Jansen and Wiegand (2003b) in the context of pseudo-boolean functions. The approach taken in that work was to use traditional run time analysis tools on the simple (1+1) cooperative CoEA. It first showed that when the function is linearly separable and the decomposition used by the algorithm matches this separability[9] there is no difference between the sequential and the parallel CoEA. It then used two carefully crafted inseparable functions to distinguish the two variants of the CoEA. On one function the sequential version performed better than the parallel one, while on the other function the reverse was true.

While this study provided useful insights into the differences between sequential and parallel communication flow, for most practical purposes one uses CoEAs more complex than the (1+1) version and also deals with domains that cannot be modeled as pseudo-boolean functions. Therefore, in this section we take another look at the effects of communication

[8]When simulating a parallel CoEA in a single process, this translates into the fact that the order in which the populations are processed is irrelevant.

[9]See Jansen and Wiegand (2003b) and Wiegand et al. (2002) for explanations of these terms.

flow in a different setting, namely using a generational CoEA with larger population size and functions defined on infinite, continuous real-number domains.

4.3.2 Experiments

The Problems

Once again, the BR_n^α family of functions was used as a test suite. The experiments were performed for $n = 10$ and $\alpha \in \{0, 0.2, 0.25, 0.4, 0.5, 0.6, 0.75, 0.8, 0.85, 0.9, 0.92, 0.94, 0.95, 0.96, 0.98, 0.99, 1\}$.

The Algorithm

The sequential communication flow behaves as described for previous experiments. For the parallel communication flow the populations evolve in parallel. Both of them are active during each generation. At the end of each generation they communicate to each other their respective best individual (bidirectional communication, epoch size 1).

As before, the evolution starts with an interaction point; a population that at this time needs to communicate a best individual, will instead communicate a random one (since the population hasn't been evaluated yet, it doesn't have a best). The number of evaluations was kept constant across experiments, namely set to 1000. Each population had a size of 20. One generation of the parallel case performs twice the number of evaluations of the sequential case, because both populations are active. Therefore we ran the sequential case for twice the number of generations for the parallel case, namely 50 generations for sequential and 25 generations for parallel. Single-best collaboration, elitism and one extra evaluation round per generation were used.

For each setting we performed 100 independent runs. In the sequential case, 50 runs started with the X population active and 50 with the Y population active. This was repeated for each value of α.

The Performance

Figure 4.18 summarizes the optimization performance for sequential and parallel communication flow over all 17 values of α. For each α, the median of best-of-run fitness over

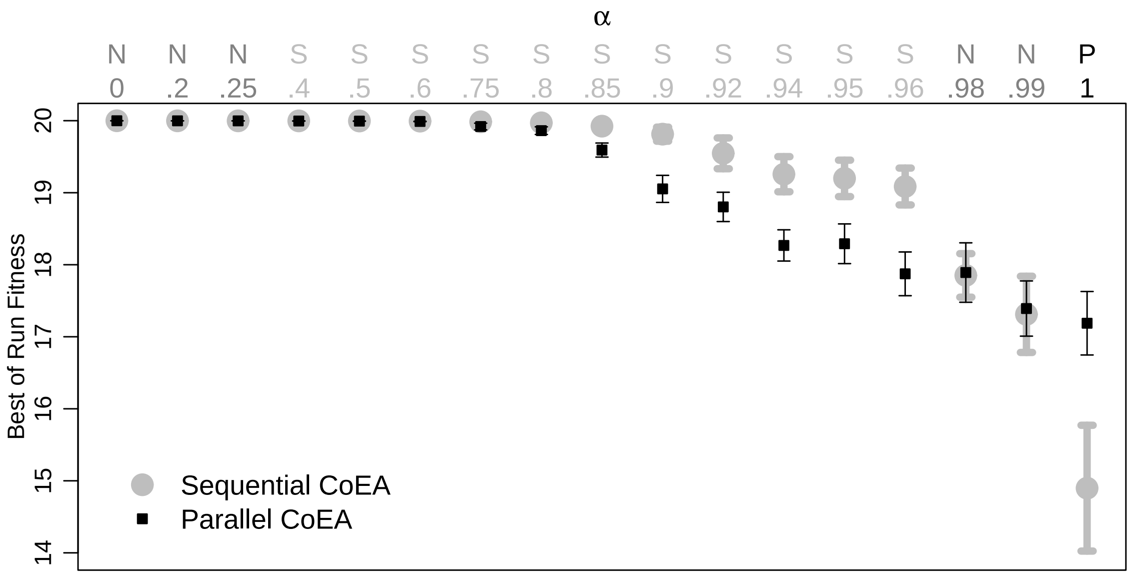


Figure 4.18: $\boldsymbol{BR_{10}^{\alpha}}$. Best-of-run statistics for communication flow. Medians and 95% confidence intervals over 100 runs. Statistical significance of medians' difference: **N** = no difference; S = sequential is better; **P** = parallel is better. Maximization: bigger is better.

100 runs is shown, together with its 95% confidence interval. Different colors and plotting symbols distinguish between sequential and parallel.

The character above each α value on the horizontal axis reflects whether or not the two medians (sequential and parallel) for that α are statistically significantly different with 95% confidence. Three color-coded situations are distinguished: dark gray **N** means no difference, light gray S means sequential performs better and black **P** means parallel is better. To test statistically significant difference we used the Wilcoxon test for medians.

What we observe is that as we increase α from 0 to 1 there is a decrease in performance both for the parallel and for the sequential communication flow (as we increase α the problem becomes harder). However, this decrease happens at different rates for the two settings. The effect is that they start out undistinguishable for $\alpha \leq 0.25$, after which the parallel setting performs progressively worse than the sequential (α from 0.4 to 0.96), then they are undistinguishable again ($\alpha \in \{0.98, 0.99\}$) and finally the parallel becomes considerably better than the sequential for $\alpha = 1$.

As the next section will show, it is once again the best-response curves that are respon-

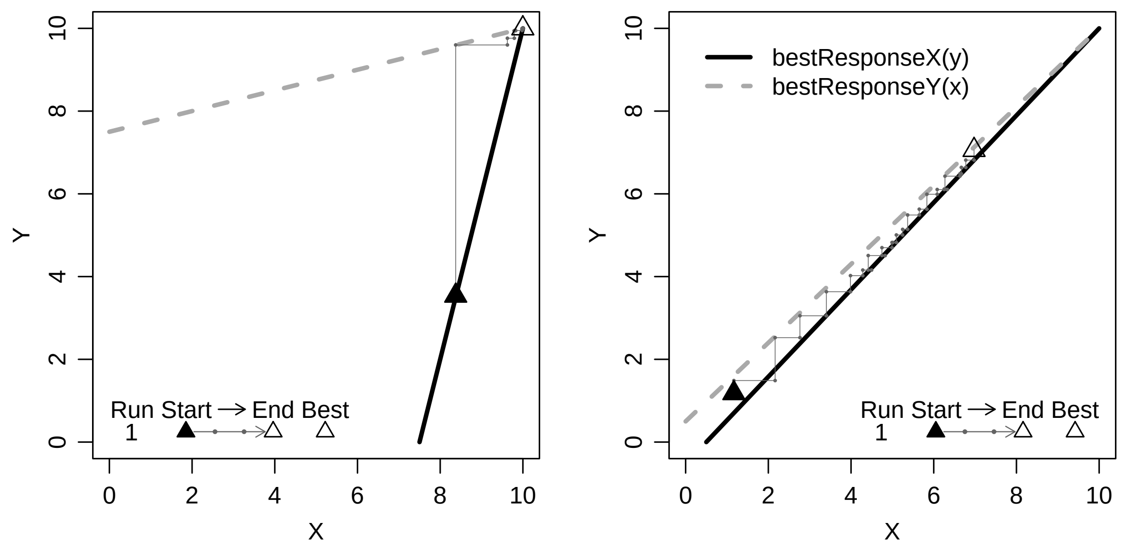


Figure 4.19: Examples of best-of-generation trajectories and best-response curves for individual runs of the sequential CoEA. Left: $\alpha = 0.25$. Right: $\alpha = 0.95$.

sible for the effects on optimization performance of the communication flow.

4.3.3 Dynamics Analysis

For sequential communication flow, we construct the best-of-generation trajectories as before and then overlay them on best-response curves. Figure 4.19 shows examples of individual runs.

For parallel communication flow, the best-of-generation individuals describe two interlaced (and interdependent) trajectories. One consists of: the best x individual at generation one, the best y individual at generation two, the best x individual at generation three, etc. The other trajectory consists of: the best y individual at generation one, the best x individual at generation two, the best y individual at generation three, etc. Figure 4.20 shows an example for $\alpha = 0.75$. The trajectories are distinguished by color and the shape of start/stop points.

As before, the trajectories are highly influenced by the best-response curves. We go on to show that this fact is the key to the performance effects observed earlier. To that extent, let us consider a deterministic system that exactly follows the best-response curves. For

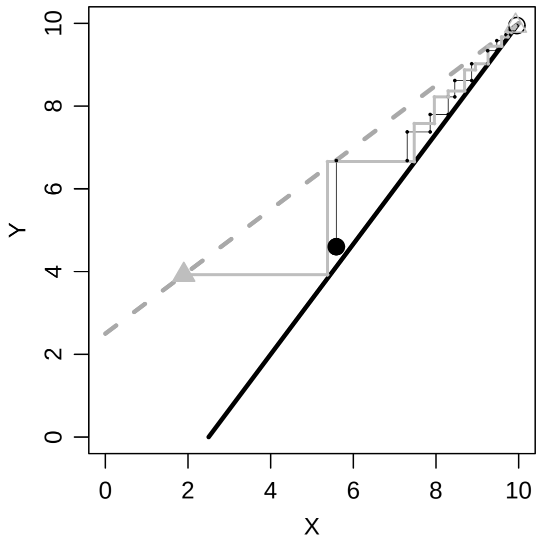


Figure 4.20: $\boldsymbol{BR_{10}^{0.75}}$. The two interlaced and interdependent best-of-generation trajectories of one run of the parallel CoEA. Triangles denote the start/stop of the trajectory started with an X generation evaluated with a random y individual. Circles denote the start/stop of the trajectory started with a Y generation evaluated with a random x individual.

such a system, given the starting point and the number of iterations we can compute the final point and consequently the function value in that point. Specifically, if the system starts with a random y, y_{rand}, and is run for n iterations, then the final point (x_{final}, y_{final}) can be computed as follows:

```
$m \leftarrow n$ div 2
$p \leftarrow n$ mod 2
$x_{final} \leftarrow n - \alpha^{2m}(n - y_{rand})$
$y_{final} \leftarrow n - \alpha^{2m+2p-1}(n - y_{rand})$
```

The end point of the trajectory of the system started with a random x can be similarly computed.

As a model of idealized performance for sequential communication flow, we generate 50 random x values and 50 random y values[10] and use them as starting points. We compute as above the corresponding final points for 50 iterations and then the value of the function

[10]Random in this case means uniformly distributed over [0, 10].

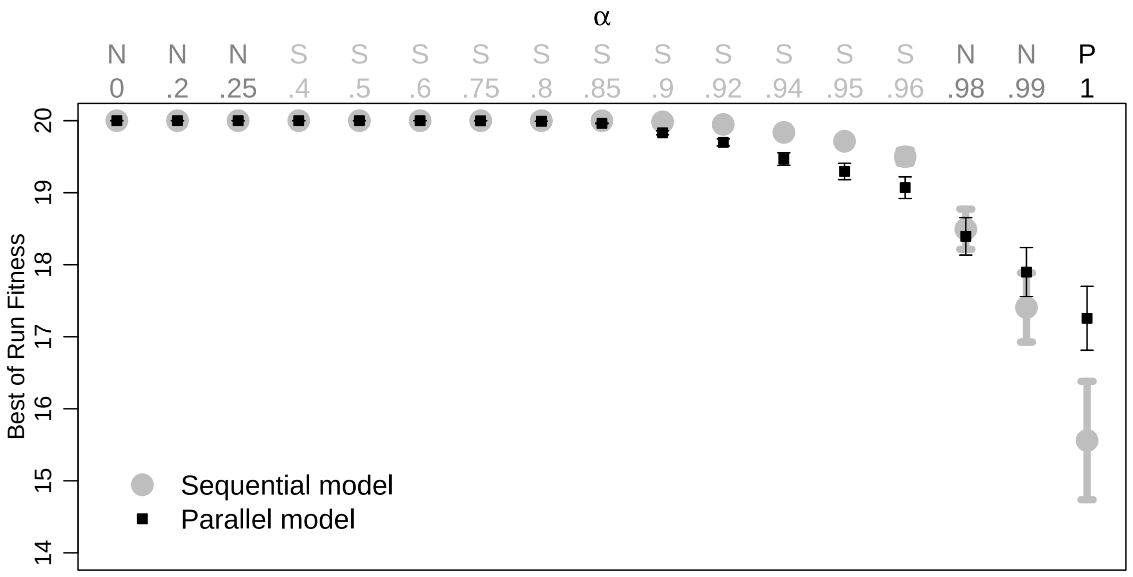


Figure 4.21: $\boldsymbol{BR_{10}^{\alpha}}$. Best-of-trajectory for model. Medians and 95% confidence intervals over 100 runs. Statistical significance of medians' difference: **N** = no difference; **S** = sequential is better; **P** = parallel is better. Maximization: bigger is better.

in those points. We end up with 100 values whose distribution should approximate that of the best-of-run fitness for the sequential setting.

To model parallel communication flow, we generate 100 pairs of random x and y values. For each such pair, we compute the corresponding final points for 25 iterations and the value of the function in those points and then take the maximum of the two. We end up with 100 values whose distribution should approximate that of the best-of-run fitness for the parallel setting.[11]

Clearly, when starting in the same point a shorter trajectory (i.e. with fewer iterations) will attain a smaller function value at the end. However, taking the maximum for two shorter trajectories should somewhat counteract this disadvantage. This is indeed what we see in Figure 4.21, which shows the results of the computations for the deterministic models (sequential and parallel) for the same values of α as before.

We see the effects are remarkably similar to those of the actual algorithm. As α increases,

[11]For the sequential setting the best-of-run is always equal to the best of the last generation (due to elitism and re-evaluation at each generation). For the parallel setting that may not be the case. For deterministic trajectories the value of the function increases monotonically from the first to the last point, thus the best of a trajectory is the last of that trajectory.

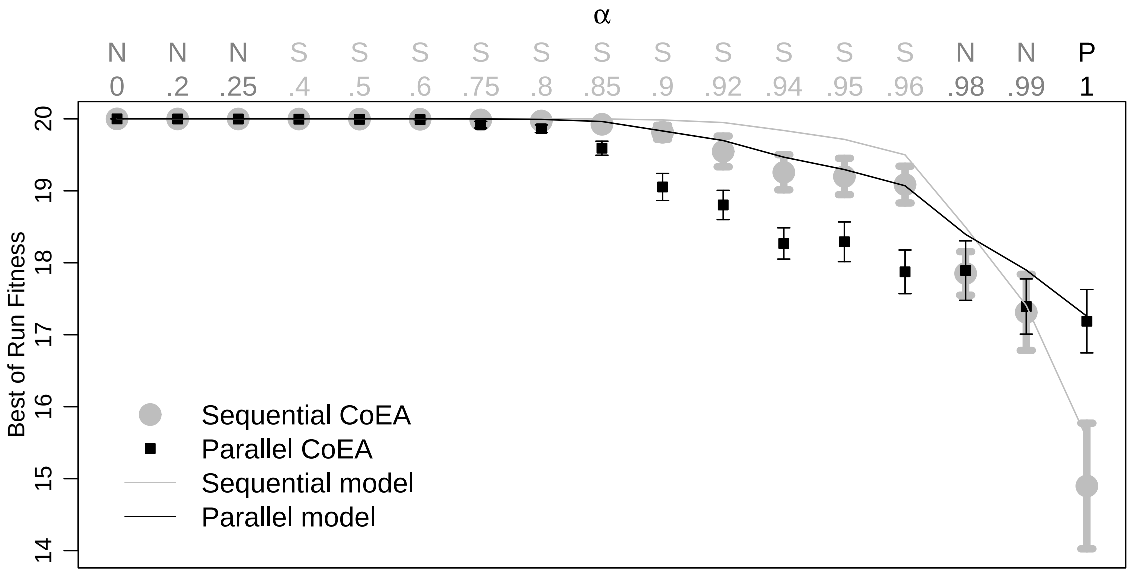


Figure 4.22: $\boldsymbol{BR_{10}^{\alpha}}$. Actual results versus model results. Medians and 95% confidence intervals over 100 runs for CoEA results. Medians only for model results. Statistical significance of medians' difference (sequential versus parallel), identical for model and algorithm: **N** = no difference; S = sequential is better; **P** = parallel is better. Maximization: bigger is better.

performance decreases, both for parallel and for sequential. And we still have the trend: first undistinguishable performance, then sequential performs better, then undistinguishable performance, then parallel performs considerably better. Moreover, for the 17 values of α considered, the categories of statistical significance for the differences between sequential and parallel (**N**/S/**P**) are the same for the model and the algorithm.

To get a better view of how the results of the algorithm relate to the results of the deterministic model, we superimpose the two, as can be seen in Figure 4.22. For visibility, we display the results of the model as lines connecting the medians and omit the confidence intervals.

Both flow settings generally perform worse than the corresponding deterministic model for non-overlapping best responses, this is due to the fact that the best-of-generation trajectories tend to take more steps which are smaller than the actual distance to the corresponding best-response curve than steps larger than this distance. This effect is the result of the difference in gradient on the two sides of each best-response curve. The best-of-generation

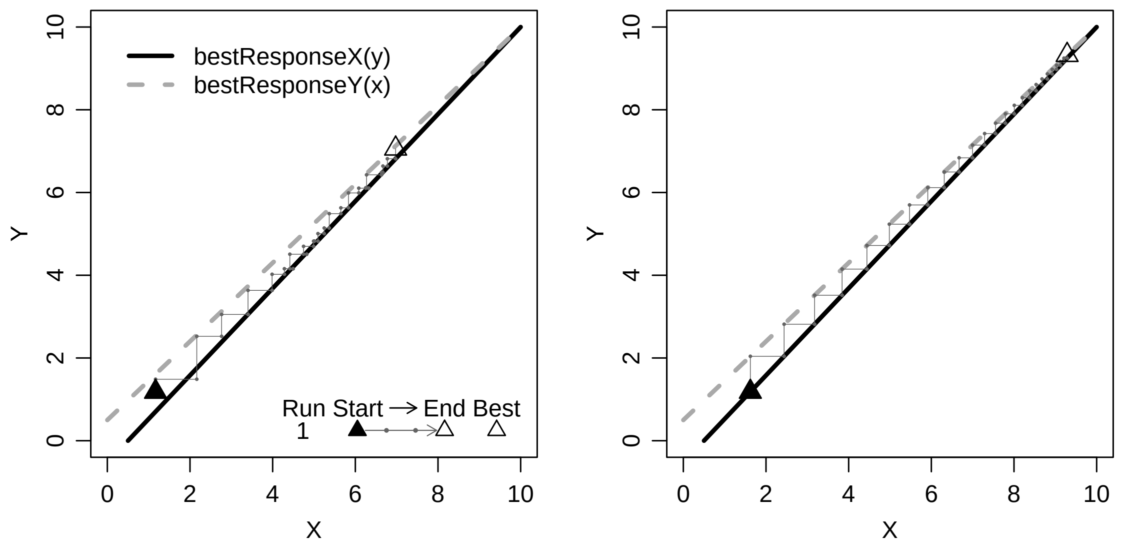


Figure 4.23: $\boldsymbol{BR_{10}^{0.95}}$. Sequential flow. Left: the best-of-generation trajectory for one CoEA run. Right: the trajectory for the corresponding deterministic model started with the same initial conditions as the run on the left.

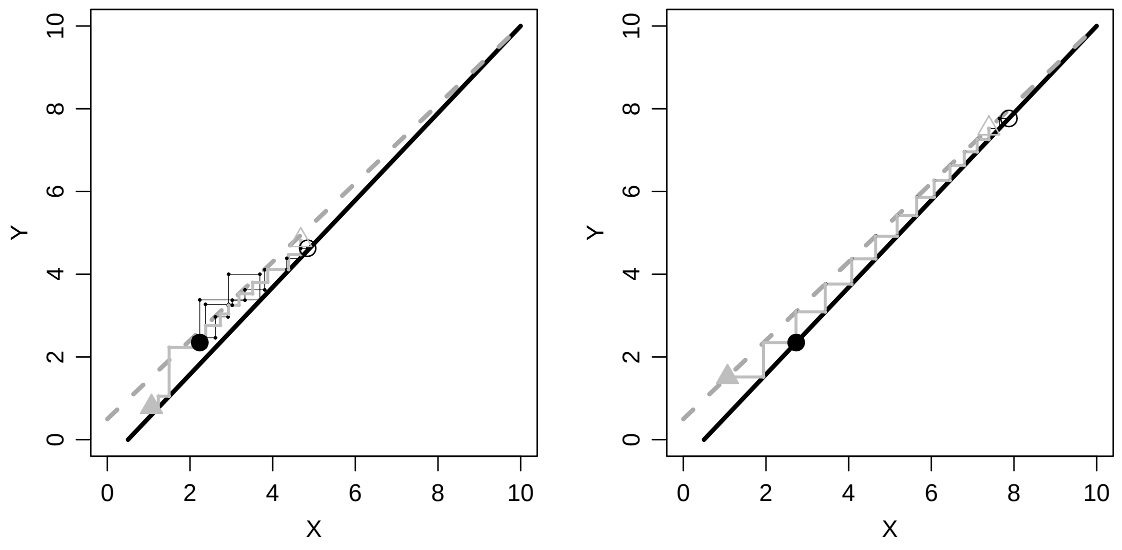


Figure 4.24: $\boldsymbol{BR_{10}^{0.95}}$. Parallel flow. Left: the best-of-generation trajectories for one CoEA run. Right: the trajectories for the corresponding deterministic model started with the same initial conditions as the run on the left. Triangles denote the start/stop of the trajectory started with an X generation evaluated with a random y individual. Circles denote the start/stop of the trajectory started with a Y generation evaluated with a random x individual.

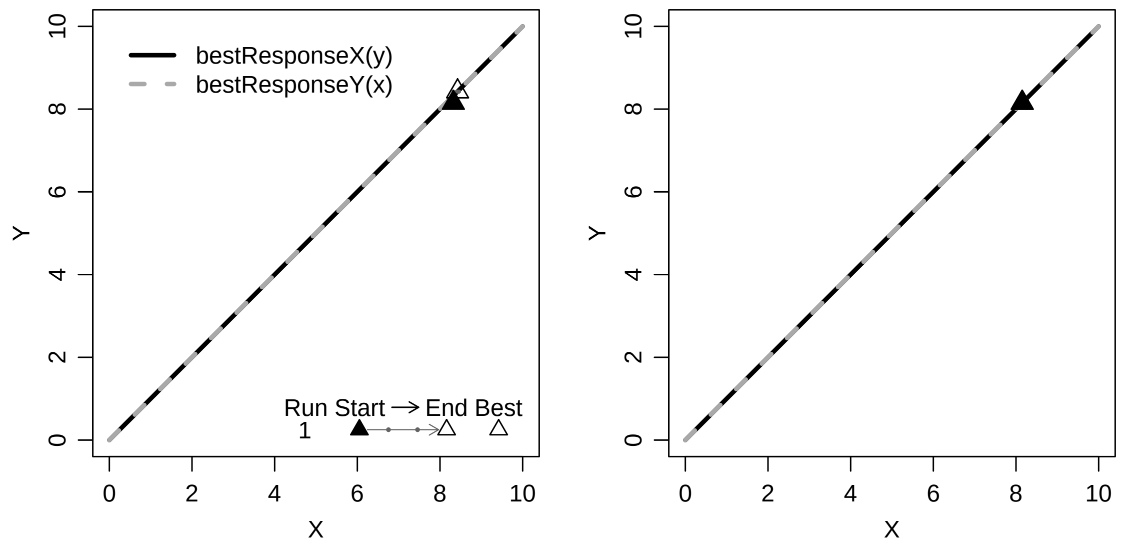


Figure 4.25: $\boldsymbol{BR^{1}_{10}}$. Sequential flow. Left: the best-of-generation trajectory for one CoEA run. Right: the trajectory for the corresponding deterministic model started with the same initial conditions as the run on the left.

trajectories thus do not travel as far as deterministic trajectories would, resulting in smaller function values. This can be observed both for the sequential case in Figure 4.23 and for the parallel case in Figure 4.24. Also note on these plots that the trajectory of the sequential CoEA travels farther than any of the two trajectories of the parallel CoEA.

For overlapping best responses ($\alpha = 1$), the sequential CoEA and the corresponding model behave extremely similar, as portrayed in Figure 4.25. As we know from all studies presented so far, the combination of elitism, single-best collaboration and sequential flow causes the trajectories to quickly converge to a close-by diagonal point. This is in agreement with the undistinguishable performance of the sequential CoEA and its deterministic model for $\alpha = 1$ as can be seen in Figure 4.22.

The trajectories of the parallel CoEA on BR^{1}_{10} behave differently both from those of the corresponding model and from those of the sequential CoEA, as can be seen in Figure 4.26. Parallelism no longer guarantees ever increasing fitness, therefore the trajectories wander around the diagonal without getting stuck on it. This explains the better performance of the parallel CoEA for $\alpha = 1$, as shown in Figures 4.18 and 4.22.

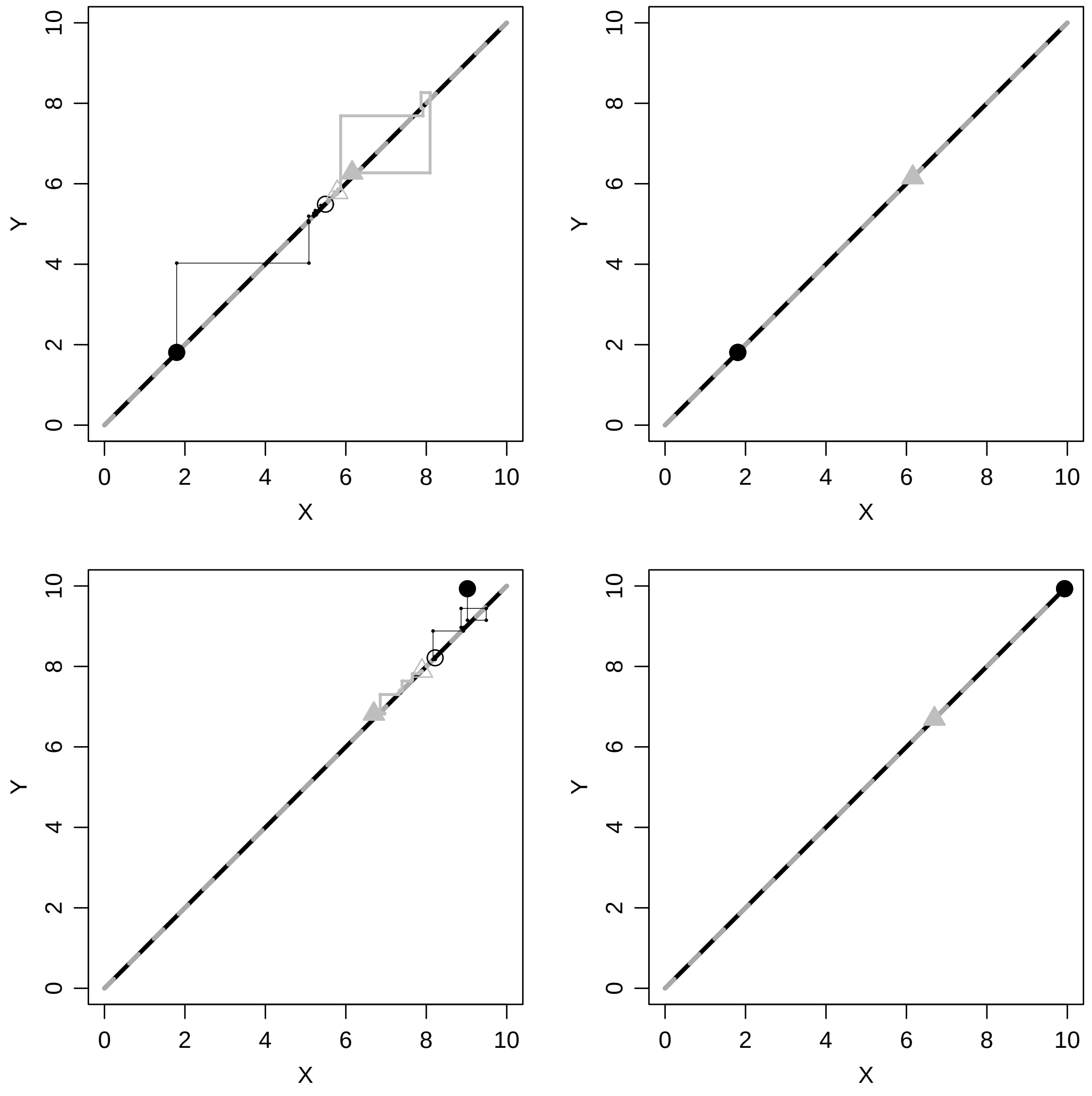


Figure 4.26: $\boldsymbol{BR^{1}_{10}}$. Parallel flow, two different runs, one per line. Left: the best-of-generation trajectory for one CoEA run. Right: the trajectory for the corresponding deterministic model started with the same initial conditions as the run on the left. Triangles denote the start/stop of the trajectory started with an X generation evaluated with a random y individual. Circles denote the start/stop of the trajectory started with a Y generation evaluated with a random x individual.

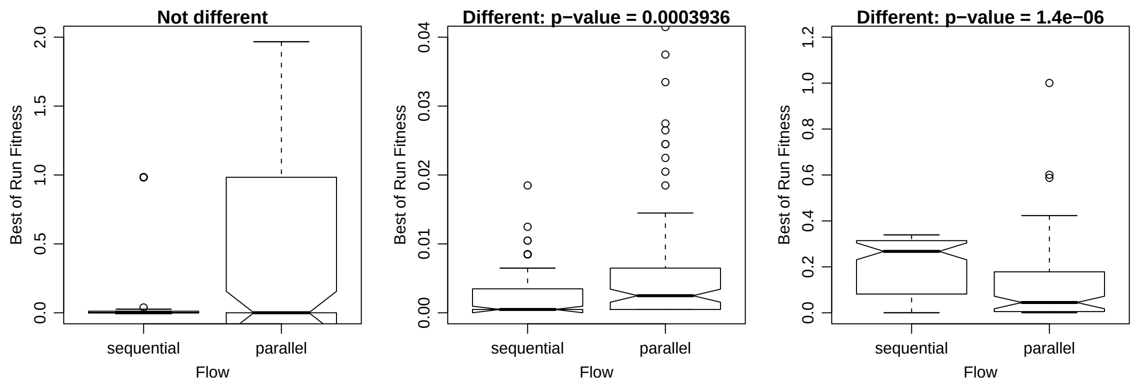


Figure 4.27: Best-of-run statistics for communication flow. Minimization: smaller is better. Left: *rastrigin*. Middle: *offAxisQuadratic*. Right: *rosenbrock*.

The heuristics we have developed through this study on the BR_n^α family of functions are as follows:

- when the best-response curves are overlapping or very close to each other, parallel communication flow has a considerable advantage over sequential communication flow;

- otherwise, the two settings are undistinguishable or the parallel performs worse than the sequential.

4.3.4 Predictive Power

We now test whether what we have learned on the BR_n^α family can be applied to other functions. We use once again the three familiar test functions, *rastrigin*, *offAxisQuadratic* and *rosenbrock*.

Refer back to Figures 3.9 and 4.16 for the best-response curves of the three functions. Based on these curves and the above heuristics, we expect the following effects of the communication flow:

- *rastrigin*: undistinguishable performance;

- *offAxisQuadratic*: undistinguishable performance or parallel does just slightly worse than sequential;

- *rosenbrock*: parallel performs considerably better than sequential.

We performed the same experiments as for BR_{10}^α, with the only exception that we adjusted the sigma of the Gaussian mutation according to the size of these new domains. The experiments confirmed our expectations, as can be seen from Figure 4.27: performance of the two flow settings is undistinguishable for $rastrigin$, sequential outperforms parallel for $offAxisQuadratic$, and parallel outperforms sequential for $rosenbrock$.

Note that both BR_n^α and $rastrigin$ are linearly separable (Wiegand, 2004) with respect to the x–y decomposition. Thus, the fact that for these functions the performance of the parallel and sequential communication flow are undistinguishable is in agreement with the results of Jansen and Wiegand (2003b). The fact that the best-response curves are lines parallel to the x and y axes is a consequence of the separability. The best-response curves are decomposition-dependent and for any function that is separable across a decomposition, the corresponding best-response curves will be parallel to the decomposition's axes.

4.4 Summary

This chapter and the previous one analyzed a wide range of variations of the basic coevolutionary algorithm as introduced by Potter (1997). In all cases, the best-response curves of the problems considered had a strong influence on the results. In addition, the dynamics analysis of best-individual trajectories proved useful for understanding how the interplay of algorithm properties and problem properties affect optimization performance. The following chapters use the same connected analysis approach and same set of tools to investigate problems with more complex best-response relationships.

Chapter 5

Analysis of More Complex Synthetic Compositional Cooperative Problems

The functions investigated in the previous two chapters all have continuous best responses. Moreover, these best responses intersect either in a single point (which is then the global optimum), or in an infinity of points (as is the case for BR_n^1). In this chapter, the analysis is extended to functions with more complex relationships between the best responses.

The first section introduces a new set of functions with discontinuous best responses that have either two or an infinity of intersection points (Nash equilibria[1]). The analysis concerns the performance effects of the population size. The second section introduces another test suite for which the focus is on "locally-best" responses and analyzes the performance effects of collaboration methods.

For both cases, the closeness of the best responses is shown to still have an influence on performance. A few additional problem properties that influence performance are investigated, such as the size of the basins of attraction of best-response intersection points and the relative values of those points. It is shown how some of these problem properties have conflicting influences on the performance effects of the parameters under investigation.

Finally, the third section uses a function from the literature to illustrate the role of the starting population for sequential CoEAs given a certain kind of asymmetry of the best responses.

As before, all functions used in this chapter are given in closed form, have analytically computable best responses and known optima. All these three restrictions are eliminated in the next chapter which features a simulation domain.

[1]As mentioned in chapter 3, the intersection points of the best responses are Nash equilibria for a deterministic system following the best responses.

5.1 Collaboration Methods and Independent Nash Equilibria

5.1.1 Experiments

Test Suite

The BR_n^α family of functions is used as a basis for constructing new, multi-Nash functions by means of concatenation. To analyze collaboration methods the following family of functions is used:

$$DBR_{p,add}^{n,\alpha} : [0,n] \times [0,n] \to \mathbb{R}$$

$$DBR_{p,add}^{n,\alpha}(x,y) = \begin{cases} BR_n^\alpha(n-x, n-y) & \text{if } p = 0; \\ BR_n^\alpha(n-x, n-y) & \text{if } p \in (0,1),\ x \leq (1-p)n,\ y \leq (1-p)n; \\ BR_n^\alpha(x,y) + add & \text{if } p \in (0,1),\ x > (1-p)n,\ y > (1-p)n; \\ BR_n^\alpha(x,y) + add & \text{if } p = 1; \\ 0 & \text{otherwise.} \end{cases}$$

Since n is fixed to 10 for all experiments reported, it will from here-on be dropped from the function's name. $DBR_{p,add}^\alpha$ is basically composed of two BR^α "hills" placed diagonally in the space, and two flat areas of value zero. These four regions are disjunct, and their relative sizes are determined by the p parameter. One hill has the peak (Nash) at $(0,0)$, the other hill at (n,n). The latter hill is "lifted" by *add*.

Figure 5.1 shows three-dimensional views of DBR for $\alpha = 0.75$, $add = 5$ and $p \in \{0.25, 0.5, 0.75\}$ and together with their respective best responses.

The performance effects of collaboration methods were investigated while varying $\alpha \in \{0, 0.05, 0.25, 0.5, 0.75, 0.95, 1\}$, $p \in \{0.25, 0.5, 0.75\}$ and $add \in \{1, 5\}$, thus resulting in $7 * 3 * 2 = 42$ test functions. For space reasons, plots of results are shown only for $\alpha \in \{0, 0.25, 0.75, 0.95, 1\}$, but the two missing values fit well within the trends described by the ones showed.

Algorithm

For each test function 5 collaboration methods were compared: single-best (1B), single-random (1R), one best plus one random (1B1R), two random (2R) and five random (5R).

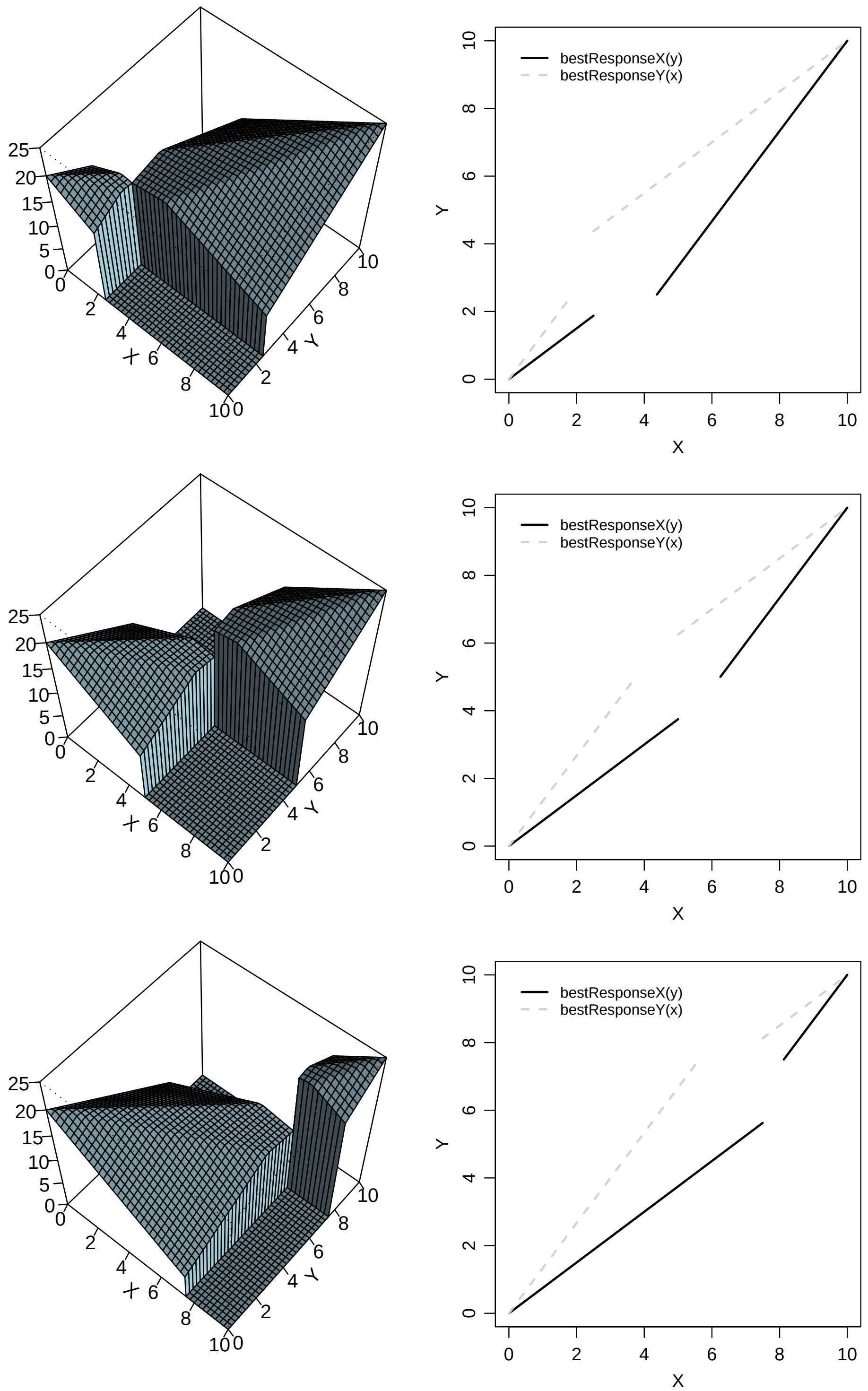


Figure 5.1: $DBR^{\alpha}_{p,add}$ with $add = 5$ and $\alpha = 0.75$. Top to bottom: $p = 0.75, 0.5, 0.25$. Left: three-dimensional view. Right: best responses.

The remaining parameters of the algorithm were set as follows: sequential update timing, epoch size 1 generation with no extra evaluation, 10 single-gene real-number individuals in each population, elitism of 1, binary tournament selection, Gaussian mutation at 90% rate with sigma fixed to 0.2 (10/50). Each run had a budget of 1000 evaluations, resulting in 100 generations for methods using only one collaborator, 50 generations for methods using two collaborators and 20 generations for 5R. For each function and collaboration method 100 independent runs were performed.

5.1.2 Results

Visualization Technique

Best-of-run Statistics Performance is compared in terms of best-of-run fitness distributions. Figures 5.2 and 5.4 summarize the results in the form of "significance plots", as introduced in section 4.2. Figure 5.2 shows all results for $add = 5$ and Figure 5.4 shows all results for $add = 1$. Each column of plots corresponds to a value of p and each line to a value of α. Thus each individual plot represents one function and shows the outcome of all pair-wise method comparisons for that function.

A comparison is represented by a square color-coded as gray, white or black. A square (i, j), with $i, j \in \{$1B, 1R, 1B1R, 2R, 5R$\}$ is:

- gray – if a statistically significant difference between method i and method j could not be detected;
- white – if there is a statistically significant difference between the methods and method i performs better; and
- black – if there is a statistically significant difference between the methods and method i performs worse.

Only the squares (i, j) above the main diagonal are shown, as the ones below can be inferred from them (by complement: gray stays gray while black turns to white and vice-versa).

As in section 4.2, statistically significant difference is tested using the Wilcoxon test (Wilcoxon, 1945) combined with Hochberg's sharper Bonferroni procedure for multiple tests of significance (Hochberg, 1988). The total confidence for each image is at least 95%.

Note that he methods are ordered such that in any pair the method on a higher line uses either *more* collaborators or an equal number of collaborators, but *more* random ones than the method on the lower line. Using this "more" metaphor and due to the fact that the plots only display the region above the main diagonal, one can simply interpret the colors as white means more is better, black means more is worse. Additional ways of easily reading information in the images include: black on a line is bad for the method on that line; white on a line is good for the method on that line; black on a column is good for the method on that column; white on a column is bad for the method on that column.

Convergence Statistics Since the purpose of this chapter is to investigate CoEAs' behavior on problems with multiple Nash equilibria, convergence to the right Nash is an issue of interest. Of course, this is tightly related to performance, however, it cannot be easily inferred from the best-of-run comparison plots. Therefore, a new type of plot is used, as shown in Figure 5.3. Each line of plots corresponds to one α value and each column to one *add* value. Each plot then summarizes the three values of p, with different styles for lines and points. For each p there is a line connecting 5 points, each corresponding to a collaboration method. A point for a particular p and collaboration method displays the percentage[2] of runs that "converged" to the hill of the higher peak/Nash.

The term convergence is somewhat abused here, for reasons of shortness of speech. What it really means is as follows: a run is said to have "conoverged" to a certain peak, if the best individual of that run is located on the hill corresponding to that peak (regardless of how far/close it may be from the peak). This is well-defined due to the fact that the two hills are disjunct[3].

Discussion

Let us start by looking at Figure 5.2 corresponding to $add = 5$. The following color trend is clearly visible: the number of black squares per plot decreases (in favor of gray and white) both from left to right on each line and from top to bottom on each column. This implies

[2]In this case, since 100 runs were performed, the percentage is equal to the actual count.

[3]The point $((1-p)*n, (1-p)*n)$ belongs to the lower peak, except for $p = 0$, when there is no lower peak.

that the performance effects of collaboration methods are affected both by the proximity of the best responses and by the relative sizes of the basins of attraction of the two Nash-es.

Consider the case when the hill corresponding to the higher Nash is larger ($p = 0.75$). On the right side of Figure 5.3, the black lines and bullets show the percentage of runs converged to the hill of the higher Nash. What one can see is that, regardless of α, the single-best collaboration method has a convergence percentage roughly close to 75% (in line with $p = 0.75$), while all methods that employ at least one random collaborator increase the convergence percentage to roughly 100%. If one were to judge only by these numbers, one would expect the random-based methods to always outperform the single-best method. As the left-hand column of Figure 5.2 shows, that is not the case, namely because, as previously mentioned, the closeness of the best-responses also comes into play.

As the analysis in section 4.1 showed, including a random collaborator decreases the accuracy of following the best-responses. This is good when the best responses are close, as it prevents the algorithm from getting stuck or barely crawling on a narrow ridge, but it is bad when the best responses are far apart. Thus, the column corresponding to 1B is completely black (meaning 1B is better than all others) for $\alpha = 0$ and $\alpha = 0.25$, half black and half gray for $\alpha = 0.75$ and completely white (meaning 1B is worse than everything else) for $\alpha = 0.95$ and $\alpha = 1$. At α close to 1, both the proximity of the best responses and the existence of a lower hill work to the disadvantage of single-best. At low α, the negative effect of the lower hill is counteracted by the high precision in reaching the peak when on the higher hill. As the higher hill is large, the latter overcomes the former.

One thing to note when comparing the random-based methods amongst themselves is that, regardless of α, 5R is worse than 1R, 1B1R and 2R. As Figure 5.3 shows, 5R has no advantage over the other random-based methods in terms of convergence to the higher hill. Moreover, remember that with a fixed budget, increasing the number of collaborators decreases the number of generations. Thus, even at α close to 1, while the low accuracy in following the best responses allows the best-individual trajectory to climb the ridge, the number of steps the trajectory is allowed is just too small. We see the same phenomena for $p = 0.5$ and it is only at $p = 0.25$, when the hill of the higher Nash becomes very small, that 5R starts to show some advantage. These two cases are discussed in the following.

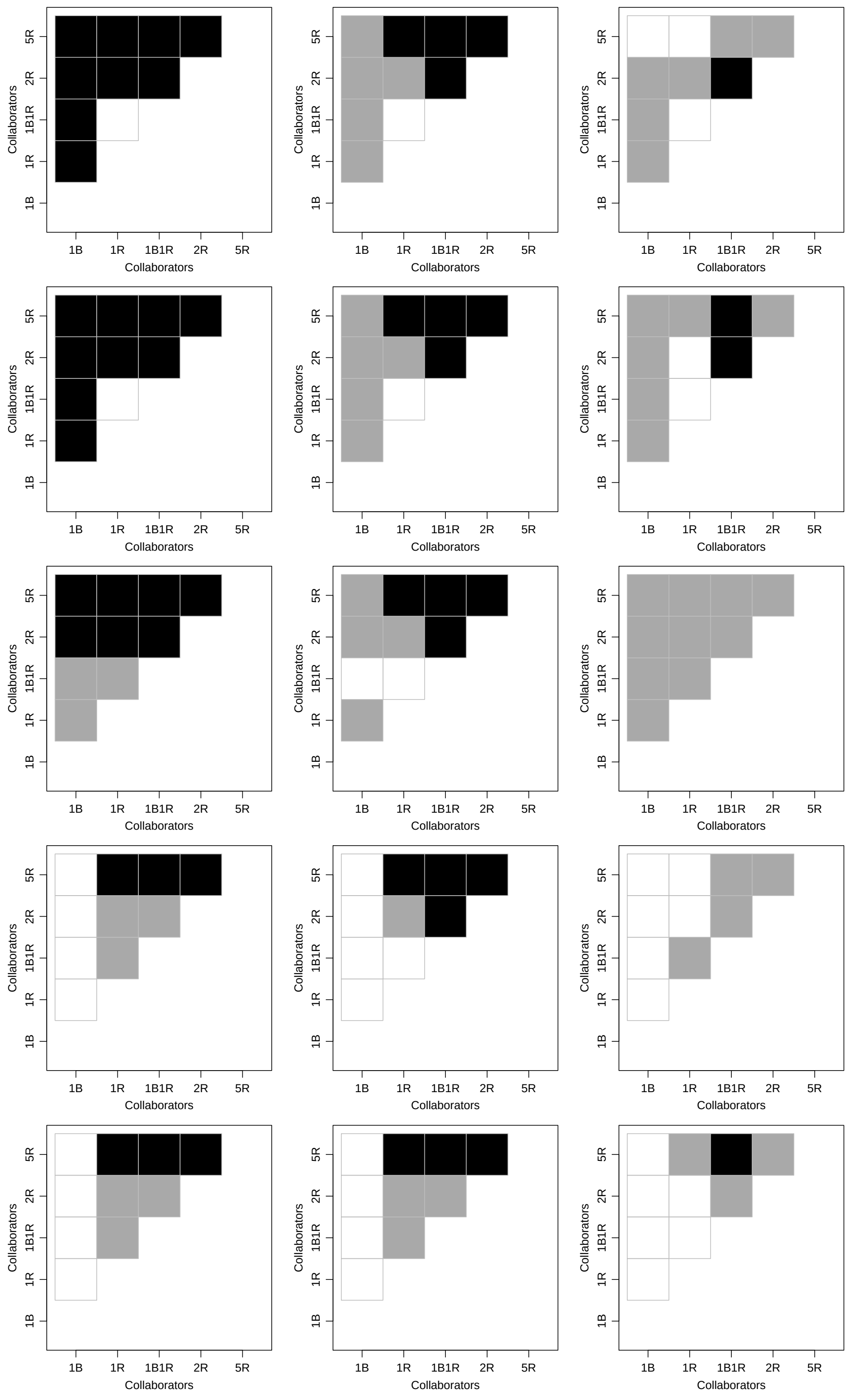


Figure 5.2: Significance of best-of-run fitness differences for $DBR^{\alpha}_{p,add}$ with $add = 5$. From left to right, per column: $p \in \{0.75, 0.50, 0.25\}$. From top to bottom, per line, $\alpha \in \{0, 0.25, 0.75, 0.95, 1\}$.

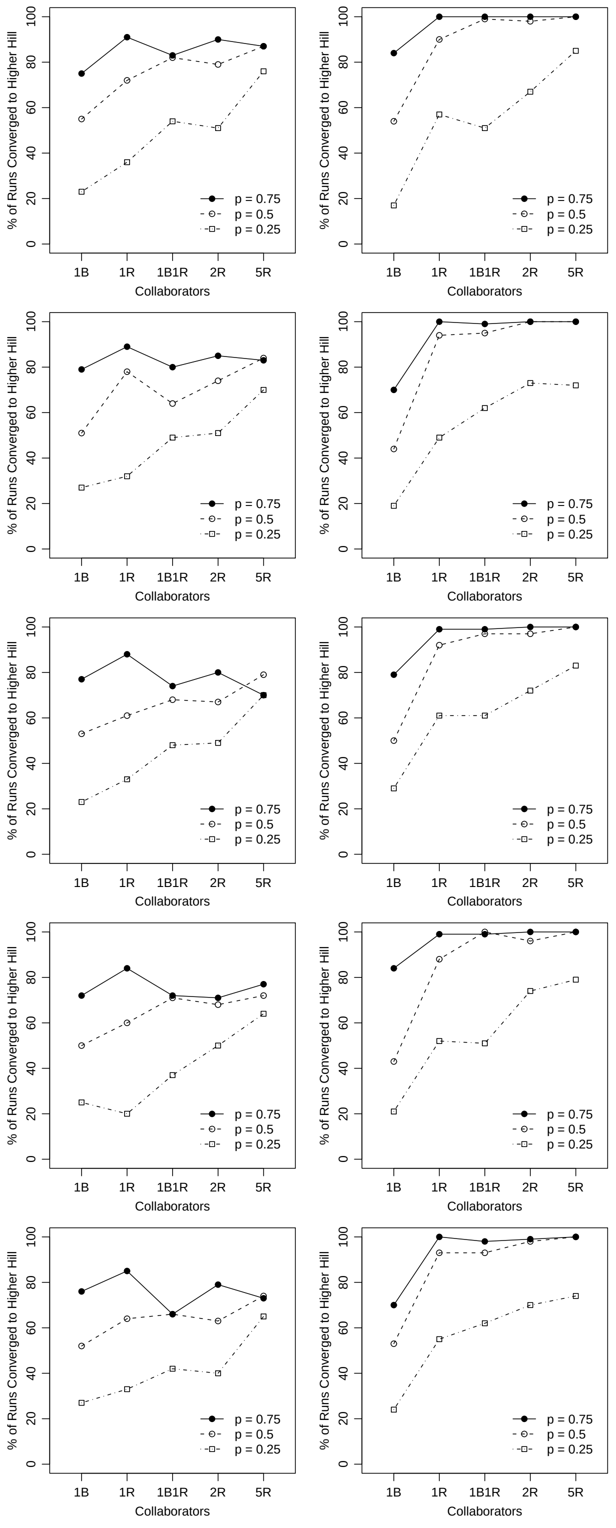


Figure 5.3: Percentage of runs that converged to the hill of the higher Nash for $DBR^{\alpha}_{p,add}$. Left: $add = 1$. Right: $add = 5$. From top to bottom, per line, $\alpha \in \{0, 0.25, 0.75, 0.95, 1\}$.

As previously mentioned, when we move from $p = 0.75$ to $p = 0.5$ to $p = 0.25$, thus decreasing the basin of attraction of the higher Nash, the number of black squares diminishes. In particular, they completely disappear from single-best's column, which they fully filled before for low values of α. This is because the single-best collaboration method doesn't have the ability to jump from one hill to another, it is confined to the hill that it started on by chance. As the size of the higher peak's hill diminishes, single-best will reach that peak less often, and it's good performance (for low α) in reaching the lower peak will not be enough to beat the random-based methods. All these methods have a visibly higher rate of convergence to the higher hill than single-best has, as shown by the dashed and dot-dashed lines on the right-hand side plots of Figure 5.3. Note also the fact that for $p = 0.5$ all random-based methods are in the same ballpark in terms of convergence, while for $p = 0.25$ methods with 2 and 5 random collaborators have higher rates of convergence to the right hill than methods with only one random collaborator. This reflects in performance, as shown in the right-hand side plots of Figure 5.4: 5R is no longer the worst of all random-based methods; most of the time it is undistinguishable from the others and occasionally it even beats 1R. 2R also beats 1R on three occasions and is undistinguishable from it otherwise.

The plot for $p = 0.25$ and $\alpha = 0.75$ shows an interesting case where the conflicting forces driving performance (small hill for the high Nash, requiring many collaborators, and medium distance between best responses, requiring less randomness) cancel each other out and no difference between the five collaboration methods can be detected.

Consider now the case $add = 1$, which means that the difference between the heights of the peaks is diminished. One would expect this to pose an additional challenge to the algorithm's ability to identify the higher peak. Figure 5.3 helps investigate this matter. The left column shows plots for $add = 1$ and the right column the plots for $add = 5$ already mentioned in the discussion above. What these plots show is that both the difference in Nash basin size (controlled by p) and the difference in Nash height (controlled by add) influence the helpfulness of random-based collaboration methods. When the basin size of the higher Nash is small ($p = 0.25$), random collaborators help regardless of difference in peak height (add). As the basin size of the higher Nash increases ($p = 0.5, 0.75$), random collaborators are clearly less helpful in increasing convergence for $add = 1$ than for $add = 5$.

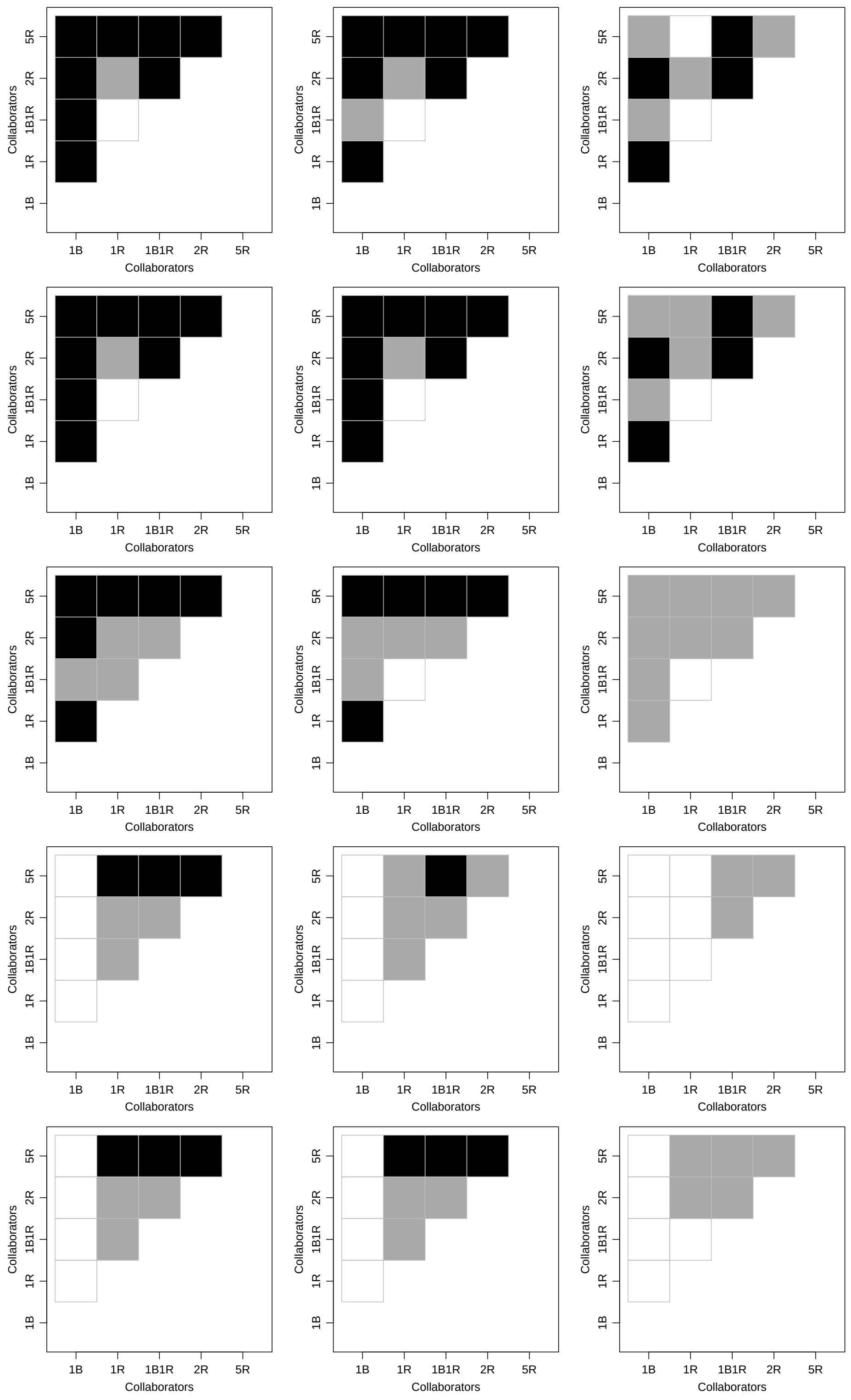


Figure 5.4: Significance of best-of-run fitness differences for $DBR^{\alpha}_{p,add}$ with $add = 1$. From left to right, per column: $p \in \{0.75, 0.50, 0.25\}$. From top to bottom, per line, $\alpha \in \{0, 0.25, 0.75, 0.95, 1\}$.

The effects of this phenomena on performance can be seen by comparing corresponding plots from Figures 5.2 and 5.4. 1B's column shows how it compares with the random-based methods. For α close or equal to 1, this column is always white, regardless of *add*. Even when random collaborators no longer improve convergence to the higher hill ($p = 0.75, 0.5$), they still prevent the algorithm from getting stuck on the ridge and are thus better than single-best.

For medium and low values of α (0, 0.25 and 0.75), improving convergence to the higher Nash was the one thing random-based methods had to their advantage for $add = 5$. This advantage is lost when switching to $add = 1$ and $p = 0.75, 0.5$. Thus, in they fare even worse than before in the comparison against single-best. Out of the 24 squares on 1B columns for $\alpha = \{0, 0.25, 0.75\}$ and $p = 0.75, 0.5$, at $add = 5$ there were 10 black, 13 gray and 1 white. At $add = 1$ there are 18 black, 6 gray and no white.

What remains to be explained is white turning to gray and gray turning to black on 1B'a column for low α and $p = 0.25$. The convergence plots show that random collaborators still improve convergence to the higher peak. Why is it the case that random-based methods still fare worse against 1B at $add = 1$ than at $add = 5$? The reason is that for $add = 1$ being on the higher hill is no longer enough, the algorithm has to reliably climb that hill in order to reach values higher than those that can be found on the lower hill.[4] Many and random collaborators are bad when it comes to climbing far apart best responses.

Finally, note that for $add = 1$ we still have the trend of black disappearing when with increasing α, thus showing that the proximity of the best responses still plays an important role in the performance effects of collaboration methods.

To summarize, there are multiple problem properties that influence the performance effects of collaboration methods in multi-Nash domains. The proximity of the best responses still plays an important role, but so do the relative sizes of the basins of attraction determined by the best responses and the relative heights of the Nash-es. Moreover, these properties can vary independently of one another and thus there exist cases where their influences are conflicting in nature. This can lead to situations like $\alpha = 0.75, p = 0.25, add = 0.5$,

[4] As an example, for $add = 5$ and $\alpha = 0$, the lowest value on the high hill is 20, which is equal to the highest value on the lower peak!

where all collaboration methods fare the same.[5]

5.2 Population Sizes and Local Best Responses

5.2.1 Experiments

Test Suite

To analyze the effects of population sizes, we used a different set of functions, as defined below. They also concatenate two BR_n^α hills, but this time along the Y axis instead of diagonally.

$$VBR_{add}^{n,\alpha} : [0,n] \times [0,2n] \rightarrow \mathbb{R}$$

$$VBR_{add}^{n,\alpha}(x,y) = \begin{cases} BR_n^\alpha(x,y) & \text{if } y \leq n; \\ BR_n^\alpha(x,y) + add & \text{otherwise.} \end{cases}$$

As before, n is fixed to 10 for all experiments reported, so it will be dropped from the function's name. One of the two hills has the peak (Nash) at (n,n), the other at $(n,2n)$. This latter hill is higher by add.

Figure 5.5 shows three-dimensional views and best responses for VBR with $\alpha = 0.75$ and $add \in \{1,5\}$. Note that the best responses do not depend on add, thus the bottom plots in the figure are identical.

The performance effects of population size were investigated while varying $\alpha \in \{0, 0.05, 0.25, 0.5, 0.75, 0.95, 1\}$ and $add \in \{1,5\}$, thus resulting in $7 * 2 = 14$ test functions. For space reasons, plots of results are shown only for $\alpha \in \{0, 0.25, 0.75, 0.95, 1\}$, but the two missing values fit well within the trends described by the ones showed.

Algorithm

For each test function 6 population sizes were compared: 5, 10, 20, 50, 100 and 200. The remaining parameters of the algorithm were set as follows: sequential update timing, epoch size 1 generation with no extra evaluation, single-best collaboration, 10 single-gene real-number individuals in each population, no elitism, binary tournament selection, Gaussian

[5]To be more accurate, we cannot reject the hypothesis that the performance of all methods is the same.

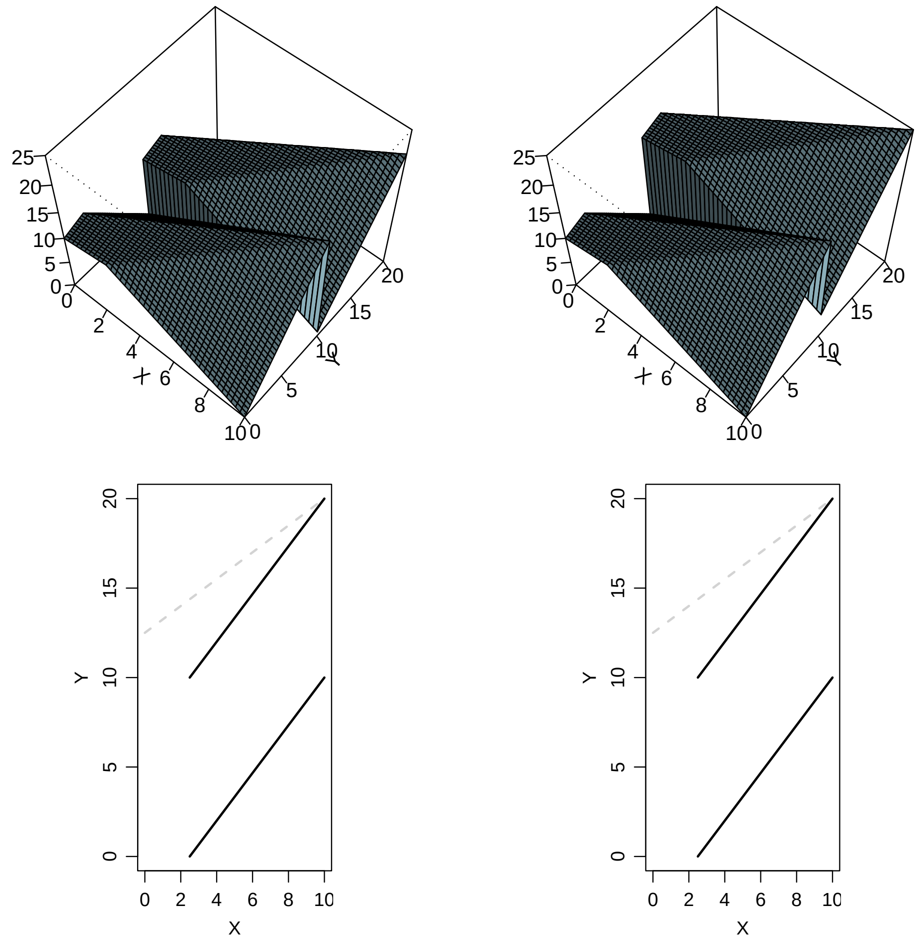


Figure 5.5: VBR^{α}_{add} with $\alpha = 0.75$. Left: $add = 1$. Right: $add = 5$. Top: three-dimensional view. Bottom: best responses; $bestResponseX$ shown in continuous black line, $bestResponseY$ shown in dashed gray line.

mutation at 90% rate with sigma fixed to 0.2 (10/50) for X and 0.4 (20/50) for Y. Each run had a budget of 1000 evaluations, resulting in respective numbers of generations of 200 (for population size 5), 100, 50, 20, 10 and 5 (for population size 200). For each function and collaboration method 100 independent runs were performed.

5.2.2 Results

The results are analyzed using the same types of plots as in the previous section. Figure 5.6 shows how the different population sizes compare in terms of the distribution of best-of-run fitnesses they produce. White means the larger population size is (statistically significantly) better, black means the larger population size is worse and gray means a distinction could not be made. Figure 5.7 shows the percentage of runs that converged to the hill of the higher Nash, with convergence defined as in the previous section.

In terms of best-of-run performance, there is a clear trend when varying α and a more subtle one when varying *add*. I start the discussion with the latter. When moving from $add = 1$ to $add = 5$, the plots get slightly "darker". With one exception ($\alpha = 0.95$) some white turns into gray or even black and some gray turns into black (while black always remains black). To understand why this is, it help to look at the convergence plots. Clearly, convergence is higher for $add = 5$ than for $add = 1$. What is more interesting though is the relation between the slopes of the lines connecting populations. Especially for low α, increasing the population size brings less of an advantage for $add = 5$ than for $add = 1$. This is of course intuitive: as the difference between hills is larger, even small populations will detect it. Since detecting the higher hill was one of the advantages of larger population sizes, when this advantage is diminished their performance relative to smaller population sizes will decrease as well, thus resulting in more gray and black squares, as seen in Figure 5.6.

As far as α is concerned, we note the same effects as for the single Nash study in chapter 3. This shows that the proximity of the best responses still plays an important role in the performance effects of population sizes. As a reminder, what happens is that there is a tradeoff between the accuracy with which best-individual trajectories follow the best responses and the length of these trajectories. For low α, the best responses are at a

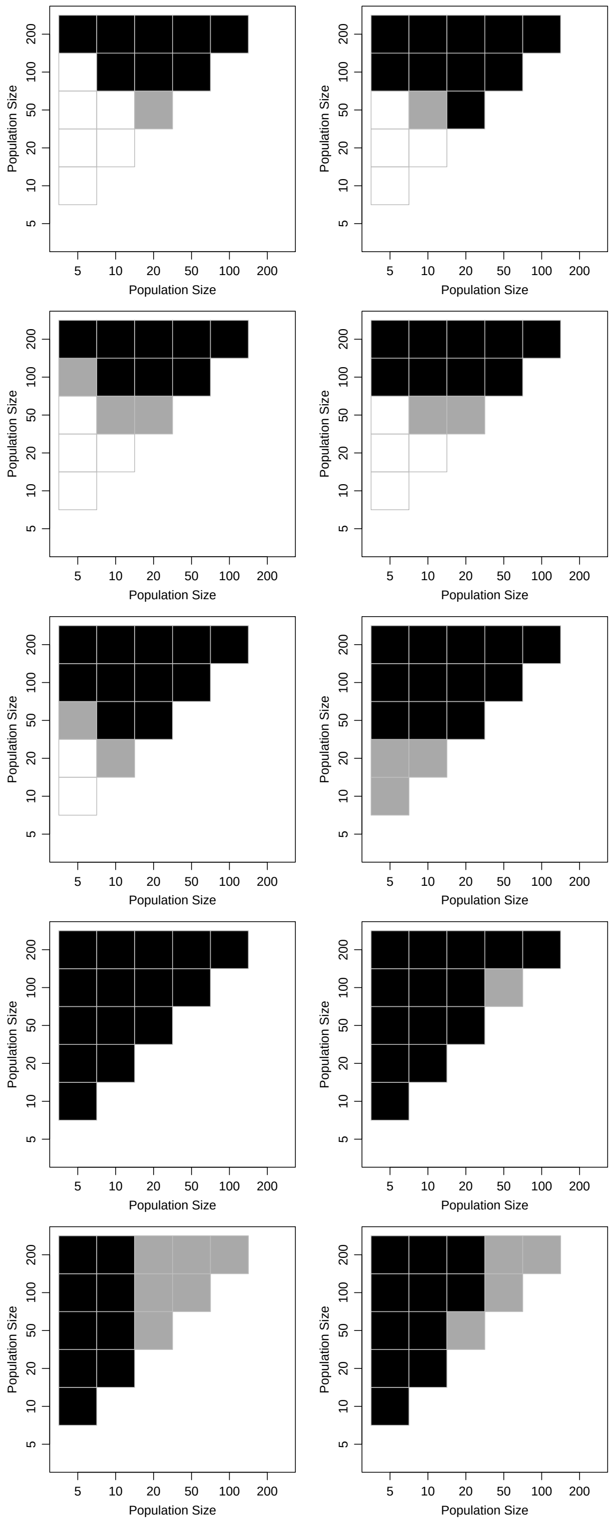


Figure 5.6: Significance of best-of-run fitness differences for VBR^{α}_{add}. Left: $add = 1$. Right: $add = 5$. From top to bottom, per line, $\alpha \in \{0, 0.25, 0.75, 0.95, 1\}$.

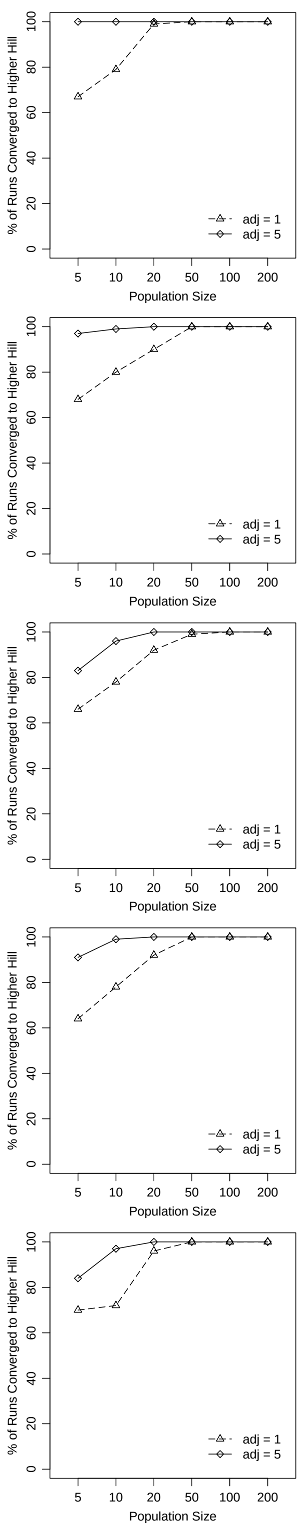


Figure 5.7: Percentage of runs that converged to the hill of the higher Nash for VBR^{α}_{add}. From top to bottom, per line, $\alpha \in \{0, 0.25, 0.75, 0.95, 1\}$.

large angle, thus accuracy is beneficial and few steps are needed to reach the Nash; thus initially performance increases with population size (white squares on the bottom-left), until it reaches a plateau (gray color) after which it decreases (black squares on the top-right). As α increases, the balance shifts in favor of low population sizes, thus the white squares get fewer and fewer and eventually disappear at α close to 1.

In general, the performance effects of population size will depend on the interaction between the influences of best-response proximity and best-response locality. In particular, when there are multiple local best-responses and the best responses for the different populations are close/overlapping, there is tradeoff to be made between increasing the chances of identifying the correct best response (requiring larger population sizes) and climbing the best responses (requiring lower population sizes). In the case presented here, low population sizes came out with the victory, but as the number of local best responses increases, the balance may shift either in favor of larger population sizes or such that no distinctions can be made. Already on the bottom left plot of Figure 5.6 there is a lot of gray denoting that population sizes 20, 50, 100 and 200 cannot be distinguished!

5.3 Starting Population and Asymmetric Best Responses

This section investigates some effects of best-response asymmetry. The function used is *AsymmetricTwoQuadratics* (Wiegand and Sarma, 2004; Wiegand, 2004), reproduced below:

$$asymmTQ(x,y) = max \begin{cases} h_1 - [s_{x1}(x-x_1)^2 - s_{y1}(y-y_1)^2] \\ h_2 - [s_{x2}(x-x_2)^2 - s_{y2}(y-y_2)^2] \end{cases} .$$

The function is the maximum of two quadratic hills, centered at (x_1, y_1) and (x_2, y_2) with respective heights h_1 and h_2. s_{x1} and s_{y1} control the relative widths of the first hill along the two axes. s_{x2} and s_{y2} do the same for the second hill. Their interplay of all four values affects which hill covers more area.

For parameter settings $x_1 = 1/4$, $y_1 = 1/4$, $s_{x1} = 500$, $s_{y1} = 25600$, $h_1 = 50$, $x_2 = 3/4$, $y_2 = 3/4$, $s_{x2} = 25600$, $s_{x2} = 500$, $h_2 = 150$ there are two peaks, one at $(1/4, 1/4)$, of height 50, and one at $(3/4, 3/4)$, of height 150. The best-response curves look like in Figure 5.8. The two peaks correspond to the two intersection points.

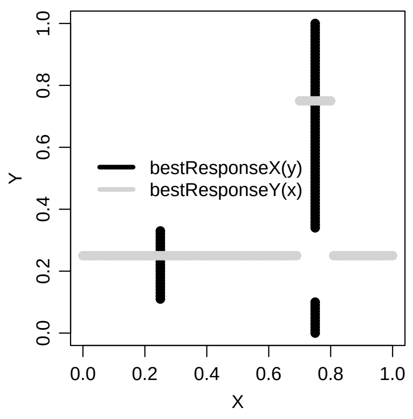


Figure 5.8: Best-response curves for $asymmTQ$ with settings $x_1 = 1/4$, $y_1 = 1/4$, $s_{x1} = 500$, $s_{y1} = 25600$, $h_1 = 50$, $x_2 = 3/4$, $y_2 = 3/4$, $s_{x2} = 25600$, $s_{x2} = 500$, $h_2 = 150$.

It is obvious from this image that a sequential CoEA with a single-best collaboration strategy will in just a couple of steps be drawn to one of the two intersection points (one of which is a local optima and the other the global optima). Which point is picked highly depends on the random y individual used for evaluation in the first X generation, or a random x individual, if we were to start with a Y generation.

If we start with an X generation evaluated in combination with a random y individual, there should be a higher probability of reaching the top-right peak. If we start with a Y generation evaluated in combination with a random x individual, there should be a higher probability of reaching the bottom-left point.

To verify these hypotheses, we ran two experiments. For both we used population size 10, elitism, and standard deviation for the Gaussian mutation set to 0.02. In one case we started with an X generation (evaluated with a random y individual) and in the other case with a Y generation (evaluated with a random x individual). All other settings were as before. We performed 100 runs and for each we tracked to which of the two peaks the best-of-run individual was closer. For the first case, 75 runs converged to the top-right peak, which has more Y-coverage. The remaining 25 runs converged to the bottom-left peak. For the second case, the situation was reversed: 81 runs converged to the bottom-left peak, which has more X-coverage, and the remaining 19 runs converged to the top-right peak.

5.4 Summary

This chapter used the knowledge gained from the studies in previous chapters to investigate more complex problems. It has shown that the best responses are still a driving force of CoEA performance, albeit not the only one. The interplay of best-response proximity, Nash value and Nash basin size can be such that no simple change to the basic CoEA can improve performance (e.g. $DBR^{0.75}_{0.25,5}$). One may need to adjust multiple parameters simultaneously to tackle such problems.

Chapter 6

Analysis of a Complex Simulation Compositional Cooperative Problem

The previous three chapters were all concerned with synthetic test suites, specifically designed to be tunable and to isolate certain properties. Studies on such functions are important, because having knowledge of and control over the problem gives us a better chance to properly identify the causes for observed performance or dynamics.

However, the question that arises is how much of the knowledge gained from such studies actually transfers to real problems. The current chapter investigates this question (and gives an encouraging answer) by focusing on a problem that is not available as a function in closed form. Instead, the values of the function to be optimized are the result of a simulation. Thus, the best responses and the global optima are unknown.

The first section describes the problem. The second section is concerned with uncovering the nature of the best responses for this problem. It reveals that the best responses combine properties of the test suites from the previous chapter. The third section analyzes the effect that combining all these best-response properties in one function has on the role of population size and collaboration methods.

6.1 Domain Description

Consider a bounded rectangular field in which there are a number of *food* items and a *ball* that bumps from the walls (the edges of the rectangle). When the ball comes into contact with a food item, it "eats" the food and enlarges its size by the size of the food. The ball has a starting location (x, y), and a starting angle $\theta \in (0, 360°)$. The goal is to have the ball eat as much food as possible while traveling as little as possible. This will be referred to as *efficiency* and defined as final-ball-radius $*$ maximum-allowed-distance / total-traveled-distance, where maximum-allowed-distance is a bound set on how much the ball can travel, so that it does not bounce around indefinitely without eating any food.

Physics and Settings

In all the experiments, the field is a square of side 600. The ball and the food items have circular shape. In bouncing off the walls, the ball behaves more like a spinning top or thumbtack bouncing off walls of lower height than its surface, in the sense that it bounces at the center rather than on the girth. The ball has no acceleration.

The ball does not bounce off the food. Instead, the food is also considered to be at lower height than the thumbtack's surface. This surface is considered to "hover" over a food item when the distance between the center of the food and the center of the ball is less than or equal to the sum of the two radiuses minus some ϵ (which was set to 1 for all experiments). Eating the food consists of removing it from the field and enlarging the radius of the ball such that its new surface area equals the sum of its old area and the area of the food. This is equivalent to setting $r_{ball}^{new} = \sqrt{(r_{ball}^{old})^2 + (r_{food})^2}$. The ball stops either when it has eaten the last piece of food on the field or when it has traveled the specified maximum distance, whichever happens sooner. The maximum-allowed-distance used in all experiments was 10000.

The A_{34} Problem

Different problems can be defined in this domain. The one considered here is finding the pair (x, y) that maximizes efficiency, given a fixed starting angle for the ball. Also, different problem instances can be constructed by changing the number, size and location of the food items, the initial size of the ball and the starting angle. The resulting problems are likely to have different characteristics and different degrees of difficulty.

The starting angle was arbitrarily set to $34°$. Thus the ball would start moving up and towards the right. Its initial radius was 4. The field was initialized to contain 9 identical food items symmetrically placed as can be seen in figure 6.1. Each food item had a radius of 8.

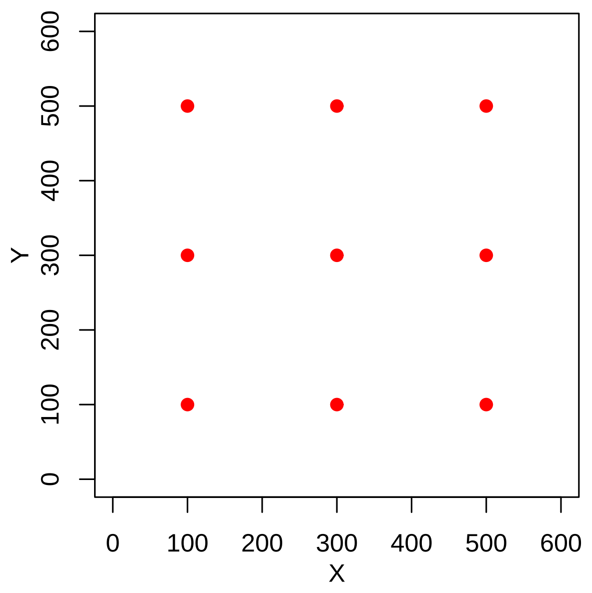


Figure 6.1: Food distribution for the A_{34} problem.

6.2 Probing for Best Responses

Given a pair (x, y), the efficiency value cannot be derived mathematically, in closed form, but must be computed by actually simulating the ball's movement. Therefore, the best responses can no longer be determined by solving derivative equations. To see whether the problem resembles at all the synthetic test suites from previous chapters, the best responses must be empirically estimated.

Probing for best responses occurs in two steps. The first consists of random search (RS). It is meant to give a rough (but broad) view of the landscape. It may hint at the nature of the best responses. The second step consists of runs of coevolution and is meant to give better information about the best responses.

6.2.1 Random Sampling

Figure 6.2 shows 12000 sample points with color-coded fitness: the darker the shade of gray, the higher the fitness value. Black encodes for the highest fitness out of these 12000 (the actual maximum is not known) and white encodes the theoretical minimum, which is equal to inital-ball-radius. The minimum would be achieved if the ball traveled the maximum distance without eating any food. Judging by figure 6.2, this seems never to happen on this particular problem. This type of plot will be referred to as a *landscape-exploration plot*.

On examining the figure, we clearly see some diagonal stripes of similar fitness, running at an angle of about 34°. This could have been expected, given the definition of the problem.[1]

What one might not have realized from the problem description alone is how narrow the stripes of high fitness are. It is within these stripes that the best responses must lie. Figure 6.3 shows intuitively why this is the case. To get an idea of $bestResponseX$, we slice with a fixed y value and look for areas of high fitness that the slice intersects. These are pictured in red and generally appear to be two. Similarly, to get an idea of $bestResponseY$, we slice with a fixed x value and look for areas of high fitness along that slice. These are pictured in blue and generally appear to be three.

The figure suggests a number of possibilities. It may be that we are dealing with non-unique best responses, or, alternatively, "local" best responses, as described in section 5.2. It may also be the case that the same stripe region contains parts of both best responses, in which case they are very close or maybe even overlapped.

In order to extract more information out of the random samples, we enhance the differences between their fitnesses by raising them to the power 20 before transforming into color gradient. This has the effect of enlarging the differences between high values. Figure 6.4 shows the results. What we see is that fitness on some stripe portions on the left (i.e. with x smaller than 150) seems to be largest, followed by stripe portions on the right ($x > 500$). Within each such portion, fitness appears to be non-constant.

Figure 6.5 displays examples of slices in this "power 20" view of the landscape. The y slices now intersect only one region of high values, either on the left or on the right, depending on the value of y. This seems to suggest that the $bestResponseX$ is unique

[1]Consider two points in the search space, P_1 and P_2, close to each other and such that the line determined by them is at an angle of 34°. Let Q_1 and Q_2 be the corresponding stop points. Then, except for the segments $\overline{P_1P_2}$ and $\overline{Q_1Q_2}$, the ball would have the same trajectory when leaving from P_1 and when leaving from P_2. The lengths of $\overline{P_1P_2}$ and $\overline{Q_1Q_2}$, and whether or not any food is eaten while traveling along these segments are the only things that would cause a (small) difference in fitness between P_1 and P_2. Note that Q_1 and Q_2 may be one and the same point if both for P_1 and for P_2 the ball stops because it ate the last piece of food.

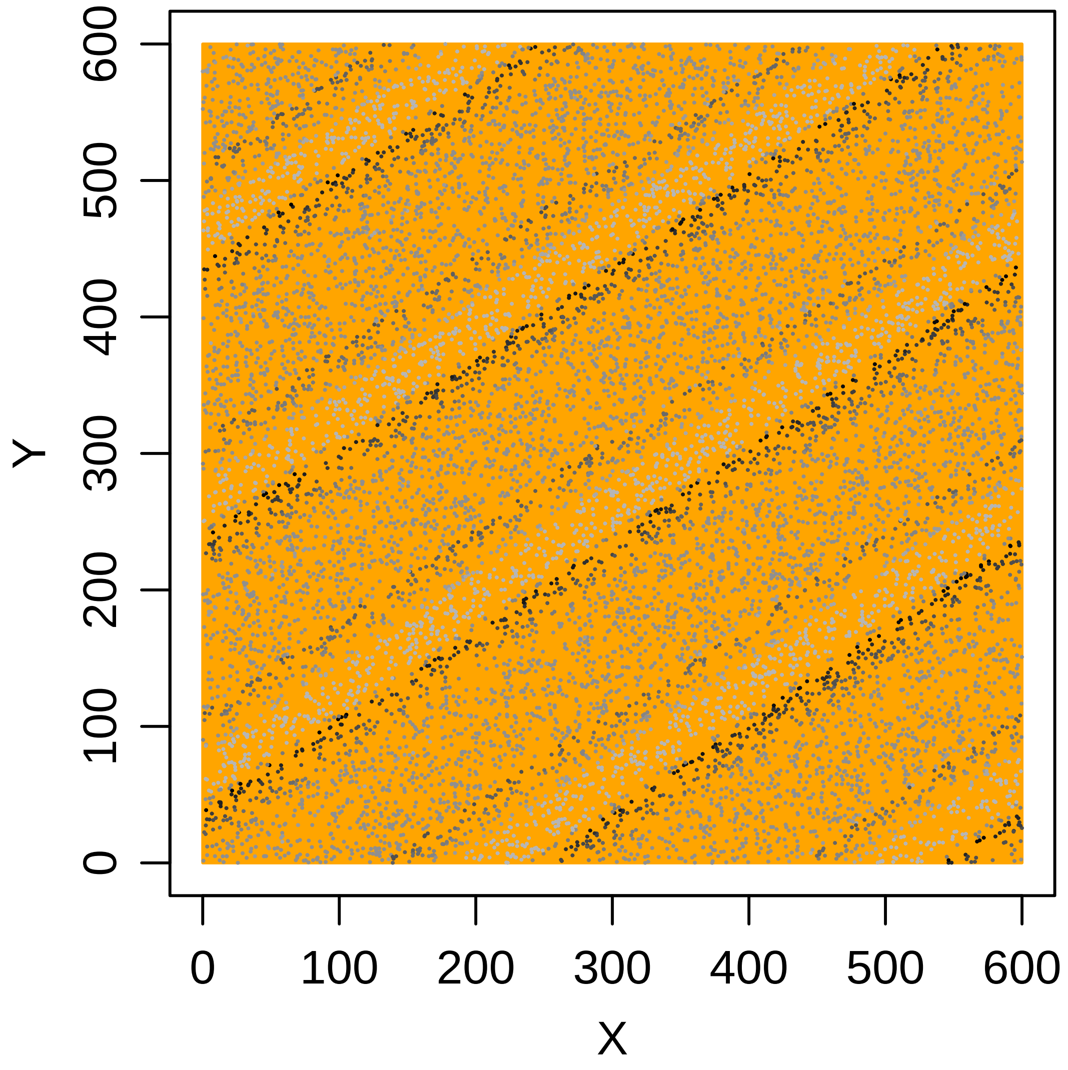


Figure 6.2: 12000 random samples for the A_{34} problem. Color-coded fitness: the darker the shade of gray for a point, the higher its fitness is.

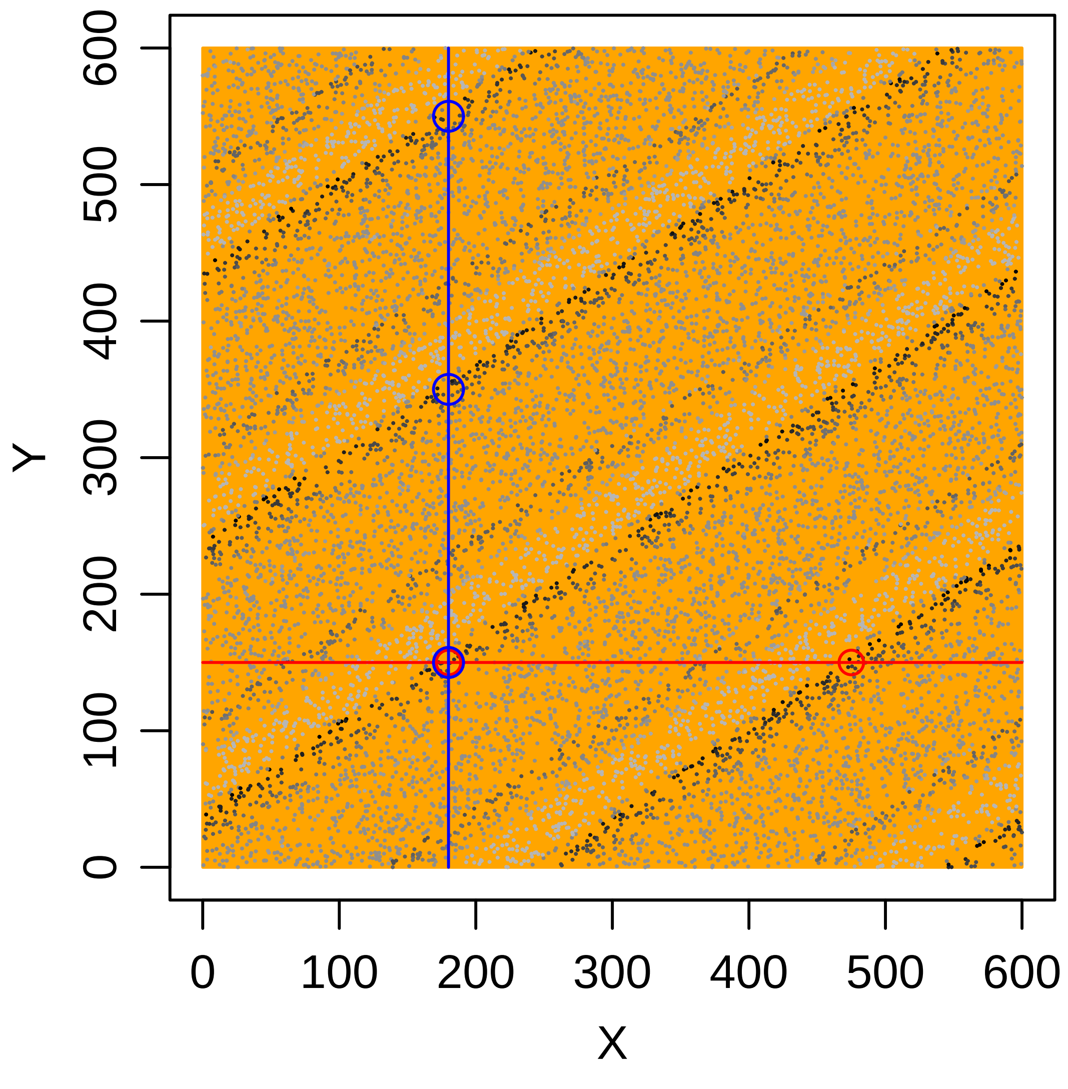


Figure 6.3: Intuition for best responses. Lines represent slices; circles represent areas of high fitness along slice. Same 12000 random samples as in figure 6.2.

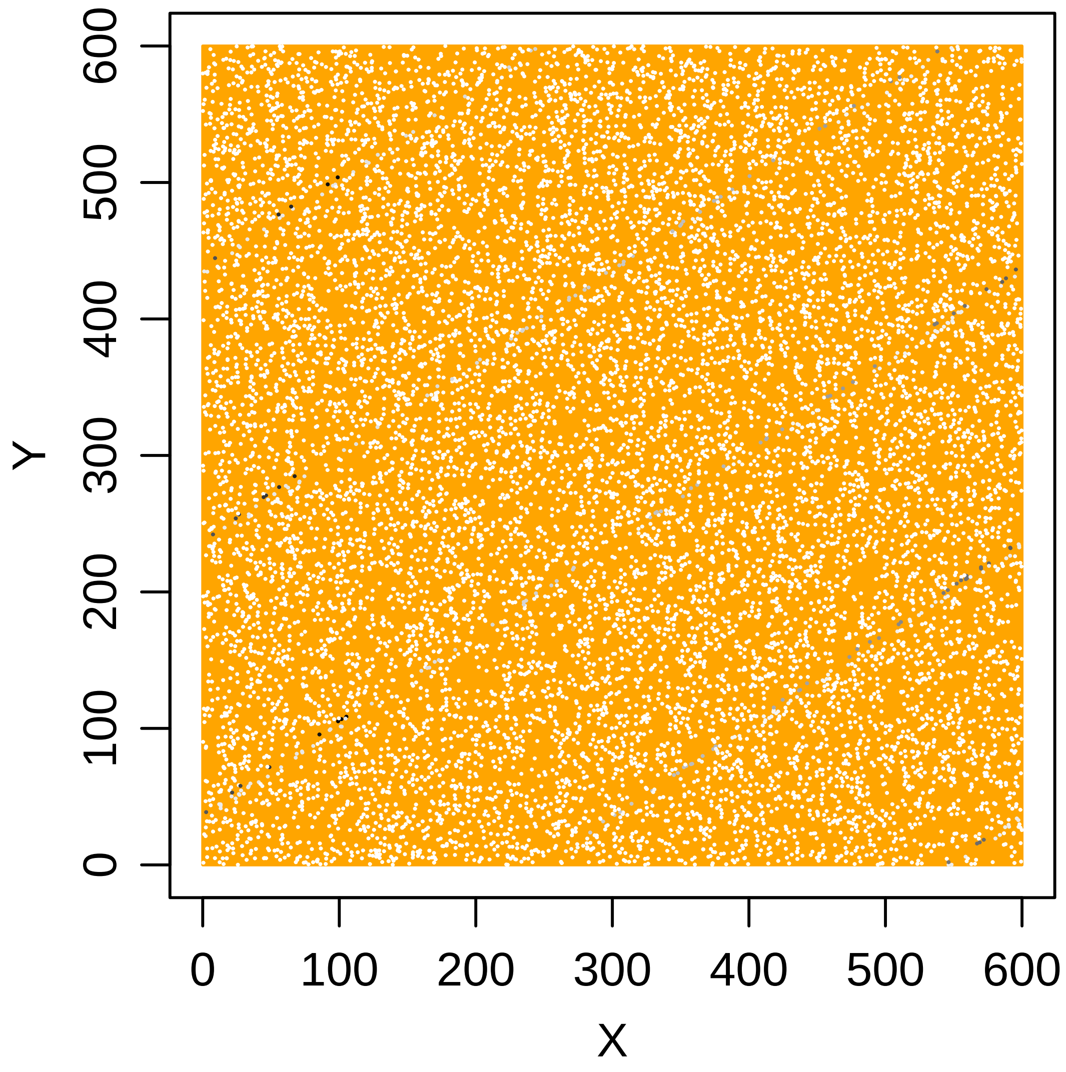


Figure 6.4: Enhanced differences in fitness by raising to power 20. Same 12000 random samples as in figure 6.2.

(unless there are points of equal fitness within the small dark region intersected by the slice). When slicing with x we still cannot differentiate between different regions of high fitness.

To summarize, it appears that there are multiple narrow stripes (ridges) of high fitness, some of which are better than others. The optimum(a) will be on the higher one(s).

6.2.2 Exploiting the Best Responses with a CoEA

In order to obtain more information about the nature of the best responses, a second domain-probing step is performed, consisting of 10 sample runs of coevolution. The knowledge from previous chapters is used to set up the algorithm's parameters such that it approximates the best responses with high accuracy. Namely, a budget of 1200 evaluations per run is split as follows: population size 40, single-best evaluation, sequential flow, 6 epochs with 4 generations and an extra round of evaluation at the end of each,[2] giving $1200 = 40 * 1 * 5 * 6$. In each population, the EA used elitism of 1, tournament selection of size 4 and Gaussian mutation with $\sigma = 12$ (600/50) operating at a rate of 90%. Figure 6.6 shows the landscape-exploration plot for this algorithm (CoEA-1). Data from all 10 runs is superimposed.

The main purpose for displaying this image is to show that (as intended) this particular coevolutionary algorithm focuses a lot on the high fitness ridges. However, it does so in an unbalanced fashion, by thoroughly exploring some portions of the ridges while leaving other portions completely unvisited. For now, the goal is to find out more about the best responses, and a different type of plot is used, namely an aggregate best-of-epoch plot, which is the equivalent of the aggregate best-of-generation plots introduced in section 3.2 and featured in Figures 3.7 and 3.8. The reason for this is that best-of-epoch individuals tend to approximate the best responses. Figure 6.7 displays best-of-epoch points (6 per run) from all 10 runs superimposed, together with a zoom-in on a part of the space for better visibility.

What these figures show is that the best-of-epoch points corresponding to X and Y, while very close, do not actually overlap, which means that by approximating the best

[2]I.e. 5 rounds of evaluation per epoch.

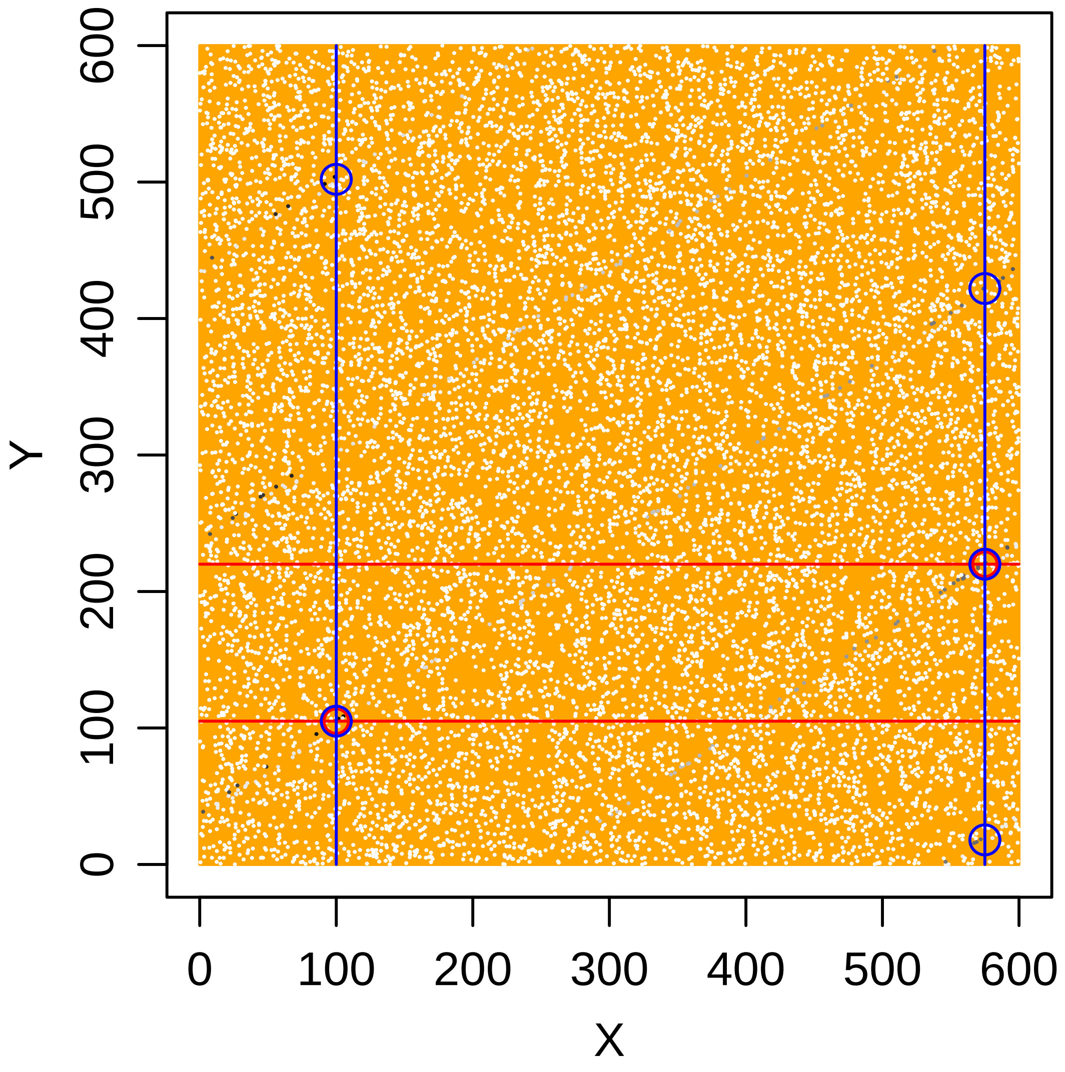


Figure 6.5: Intuition for best responses. Lines represent slices; circles represent areas of high fitness along slice. Enhanced differences in fitness by raising to power 20. Same 12000 random samples as in figure 6.2.

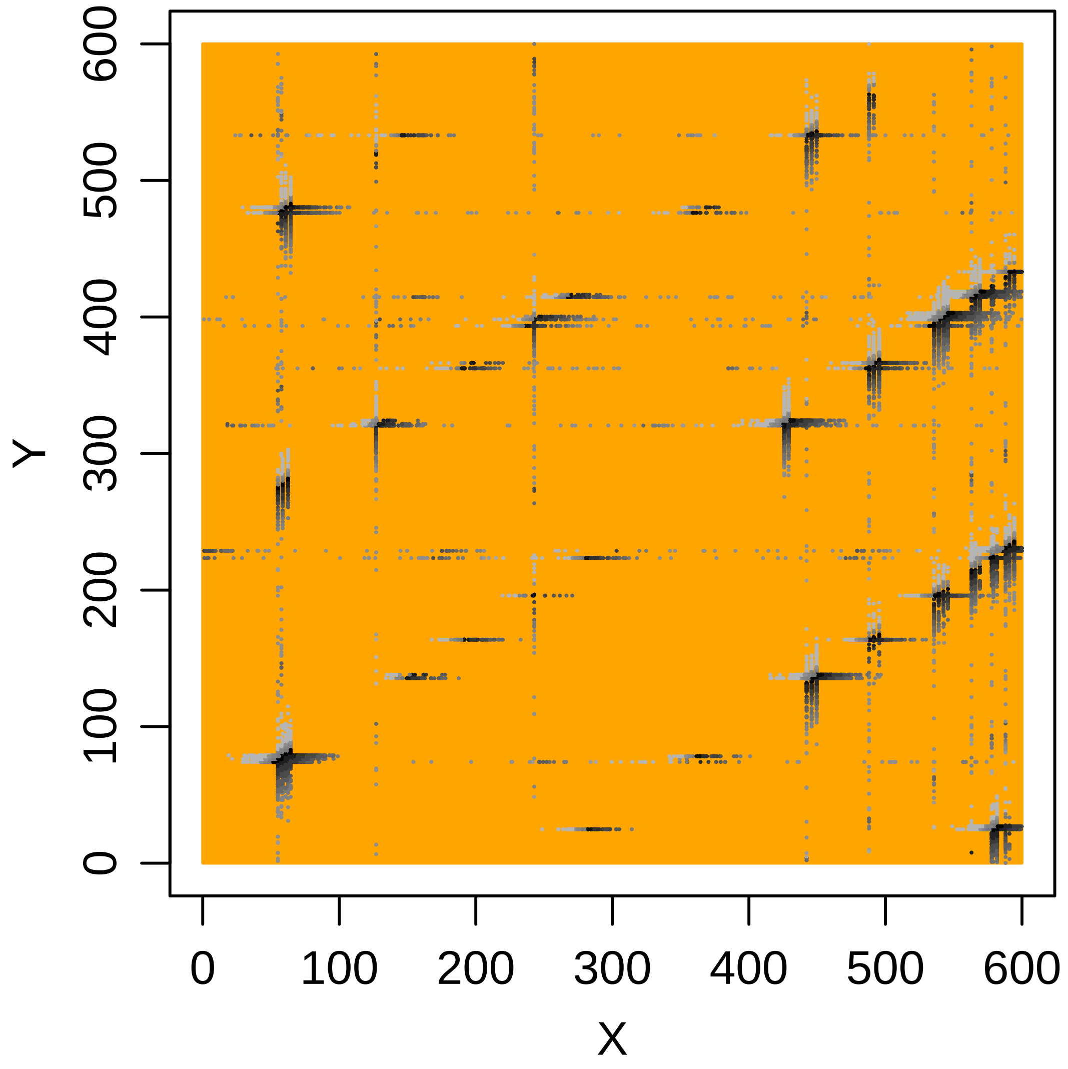
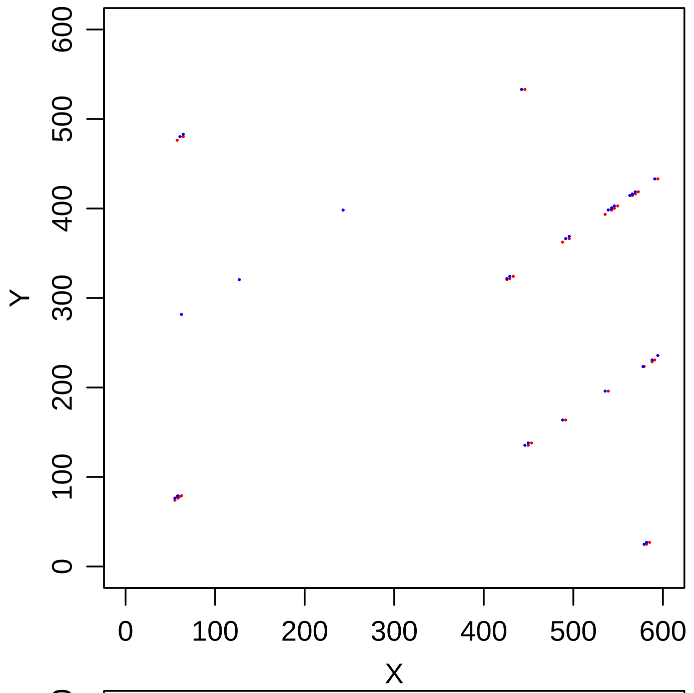


Figure 6.6: All evaluated points from all 10 runs (12000 points total) of CoEA-1. Gray-coded fitness.



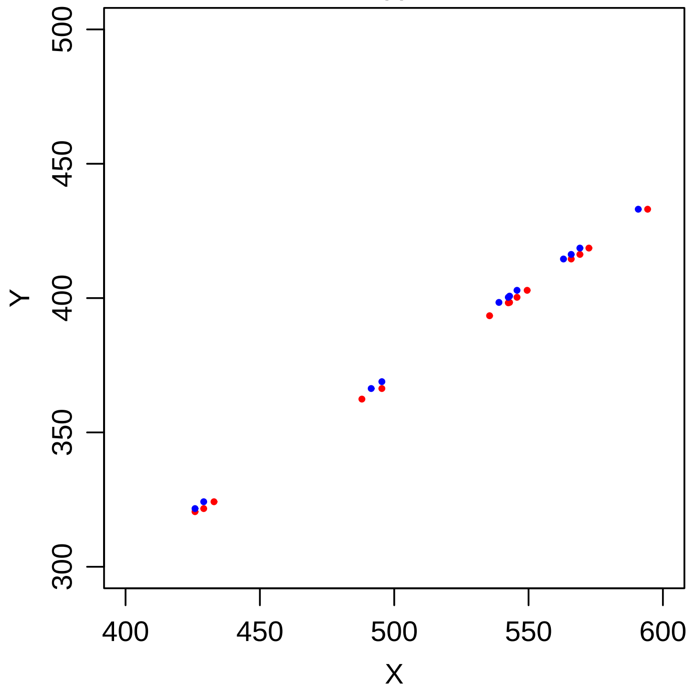


Figure 6.7: Best-of-epoch points (best-of-epoch individuals with corresponding collaborators). X epochs are shown in red and Y epochs in blue. Bottom image displays zoom-in.

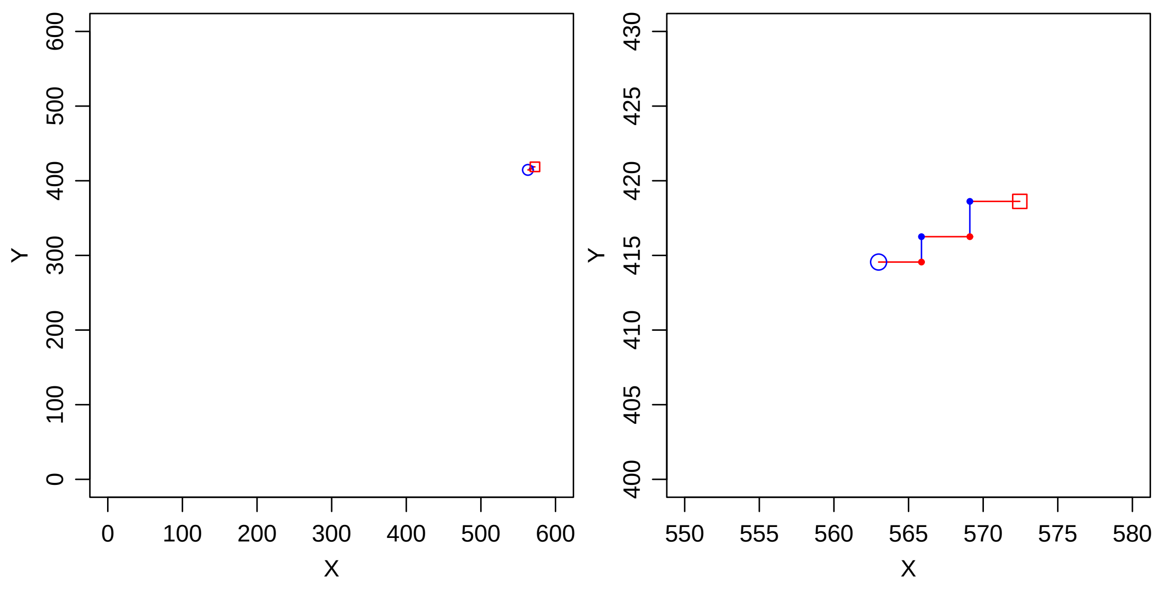


Figure 6.8: Best-of-epoch trajectory for one individual run (in particular, one started with the Y population active). X epochs are shown in red and Y epochs in blue. Start of trajectory (best of first epoch) is shown with an empty circle. End of trajectory (best of last epoch) is shown with an empty square. Left: full space. Right: zoom-in.

responses coevolution may actually "climb" towards higher values. This climbing can be seen in action in figure 6.8, which displays the best-of-epoch *trajectory* for an individual run, by connecting that run's best-of-epoch points in chronological order. The right side plot shows a zoom-in for convenient visualization. This type of plot was first introduced in section 4.2.3.

To summarize again, it appears that:

- $bestResponseX$ is discontinuous in some points, which could lead to independent Nash-es;
- $bestResponseY$ is either non-unique or a case of local best responses;
- segments of $bestResponseX$ and $bestResponseY$ are very close to one another, forming very narrow angles;
- the fitness along the best responses is not constant, but increases at a small rate.

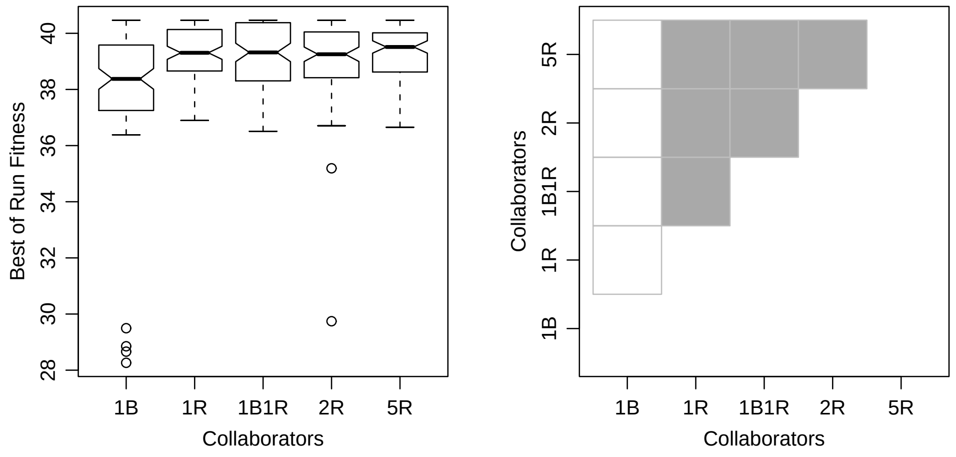


Figure 6.9: Comparison of best-of-run performance for different collaboration methods on the A_{34} problem. Left: boxplots of best-of-run fitness distributions. Right: statistical significance of differences between distributions.

All these observations point towards a problem that could have both multiple independent Nash-es and multiple local Nash-es, thus a combination of the properties studied in sections 5.1 and 5.2. The next section studies the effects of collaboration methods and population sizes on performance. The launched hypothesis is that these effects will be similar to the ones observed on the synthetic functions in the previous chapter.

6.3 Analysis

To analyze the performance effects of collaboration methods and population sizes, the same type of experiments as in sections 5.1 and 5.2 are performed. The only adjustments are the ranges for X and Y, set now to $(0, 600)$, and the standard deviation for the Gaussian mutation, set to 12 (600 / 50). All other algorithm parameters were the same.

6.3.1 Collaboration Methods

Figure 6.9 summarizes the effects of collaboration methods on performance. The significance plot looks remarkably similar to the plot in figure 5.4 for $DBR^{\alpha}_{p,add}$ with $p = 0.25$, $\alpha = 1$ and $add = 1$. Indeed the probing of the domain in the previous section showed that the

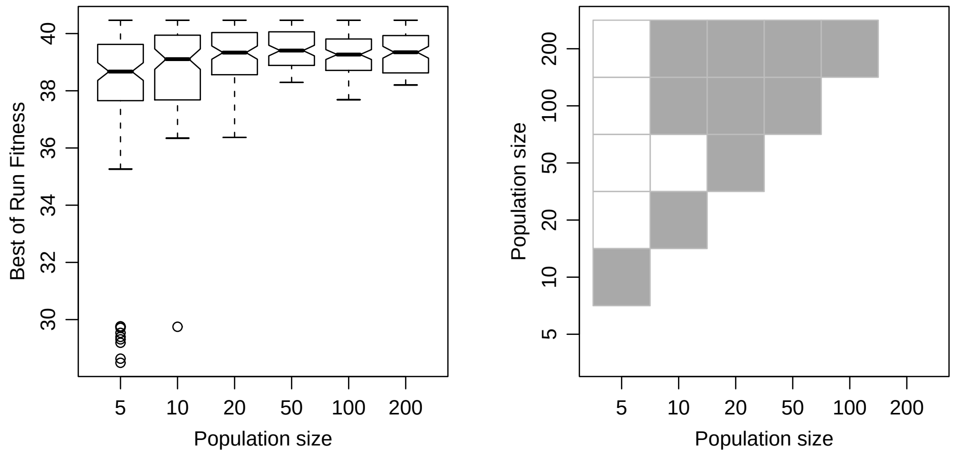


Figure 6.10: Comparison of best-of-run performance for different population sizes on the A_{34} problem. Left: boxplots of best-of-run fitness distributions. Right: statistical significance of differences between distributions.

difference between the stripes on the left and the ones on the right was fairly subtle. The significance plot is also generally consistent with all DBR plots for $\alpha = 1$, in terms of the single-best being the worst-performing collaboration method. This is in line with the proximity of this problem's best responses, as detected during the probing phase in section 6.2.

6.3.2 Population Size

Figure 6.10 summarizes the effects of population size on performance. This is similar with the plot for VBR_{add}^{α} with $\alpha = 1$ and $add = 1$ (bottom left of figure 5.6) in terms of having a large number of gray for squares for big population sizes. The difference is that for the current problem small population sizes are worse than big population sizes (5 is worse than all above and including 20 and 10 is worse than 50), rather than better, as it was the case for VBR_1^1. This shift of balance is not surprising, as the A_{34} problem has local best responses in both dimensions, and more than two of them for Y! Population size 5 just isn't enough to determine the right answer. Also note that, unlike for VBR, fitness along the best responses increases at a fairly low rate. Because of this, it is more advantageous to be stuck (or crawl) on a higher ridge (which happens with large population sizes) than to move (imprecisely!) along a lower ridge (which happens with a small population).

6.4 Summary

The results in this chapter argue that knowledge gained from studies on synthetic problems *can* transfer to real problems. Additionally, section 6.2.2 suggests heuristics for tuning CoEAs to problems. Specifically, one can first approach a problem with a CoEA setup to exploit the best responses, in order to extract some information about them. This information is determined by tracking the trajectories of best individuals while running the algorithm. Based on the nature of these trajectories and on the knowledge generated by the studies in this dissertation, informed decisions can be made on how to adjust the algorithm's settings to improve performance.

Chapter 7

Analysis of Simple Synthetic Test-based Competitive Problems

The previous four chapters all studied compositional problems defined over cooperative domains with ideal partnership as the solution concept. This chapter focuses on test-based problems defined over competitive domains and with optimum-average-payoff as the solution concept.

The analysis targets the relationships between the internal subjective metric used as fitness by a coevolutionary algorithm and the external objective metric measuring the algorithm's progress towards the envisioned goal. The CoEC setups analyzed consist of a basic CoEA and simple domains with a computationally testable goal. I believe one needs to understand such simple setups first before moving on to more complex (e.g. untestable) goals or more complex algorithms. In fact, as the reader will see, even in basic setups there is a wide range of possible relationships. The dynamics analysis provides insight into the causes for these relationships and confirms once again that the best responses problem property is a strong driving force for CoEA behavior.

7.1 Related Work

The only previous analysis of relationships between internal and external metrics was (Watson and Pollack, 2001), in the context of some number-game domains. However, that research cannot be directly extended to other domains, as the external objective metrics are problem dependent. Also, it did not provide explanations for the observed behavior. By contrast, both these issues are addressed in this chapter.

Of the variety of goals for which coevolution was applied, the investigation here concerns one which is characteristic of test-based problems with two asymmetric roles and competitive payoff, such as evolving sorting networks vs. input sequences (Hillis, 1990) or cellular automata vs. initial conditions (Juillé and Pollack, 1998). The goal is to find the individual in the first role with the best average payoff over interactions with all individuals in the second role (e.g. networks that sort all binary input sequences; CAs that correctly classify the maximum number of initial conditions). In other words, the solution concept is optimum-average-payoff.

7.2 Experiments

7.2.1 The Problems and the External Objective Metric

All domains have two asymmetric roles whose behavior sets are subsets of $\mathbb{R}$. A domain is therefore expressed by a two-parameter function $f : D_X \times D_Y \rightarrow \mathbb{R}$; $D_X, D_Y \subset \mathbb{R}$. The function f defines the metrics of behavior (payoffs) for both roles in the following way: for any event $(x, y) \in D_X \times D_Y$, the payoff for x is $f(x, y)$, while the payoff for y is $max_{(x',y') \in D_X \times D_Y} f(x', y') - f(x, y)$. The domains are thus competitive, constant-sum.

For all problems, the space $\mathcal{S}$ of potential solutions is D_X and, as previously mentioned, the solution concept is optimum-average-payoff. Formally, this means the goal is to find x_0 which maximizes $e(x) = \int_{D_Y} f(x, y)\, dy$. $e(x)$ represents the cumulative payoff for x, as it was defined in section 2.2.4. The average (or expected) payoff is given by $avg(x) = e(x)/size(D_Y)$. $avg(x)$ will be the *external objective metric* of performance with respect to reaching the goal.

Four different domains were used, corresponding to four functions defined on $[0, n] \times [0, n]$. In all the experiments presented here $n = 8$ was used (i.e. $D_X = D_Y = [0, 8]$). Three dimensional views of all four functions are shown in Figures 7.1 and 7.2. These figures thus show what the landscapes look like from the point of view of individual interactions (and therefore internal fitness). They are very much alike with regard to ruggedness/modality. By contrast, Figures 7.3 and 7.4 show us what the landscapes look like from the point of view of our external objective metric, the one we really hope to optimize. They plot for each of the four functions $avg(x)$ vs x. These plots confirm that the domains are similar. They all have a single global maximum, no local maxima and a smooth gradient. We shall see

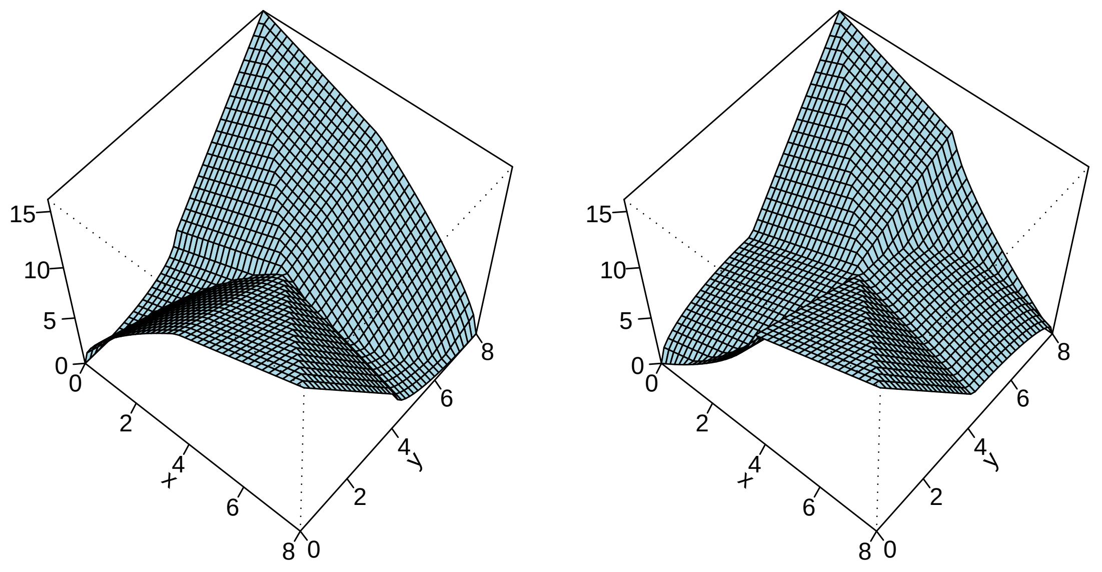


Figure 7.1: Internal landscape perspective (individual interaction payoff surfaces); left: *collapsingRidges*, right: *expanding-Ridges*

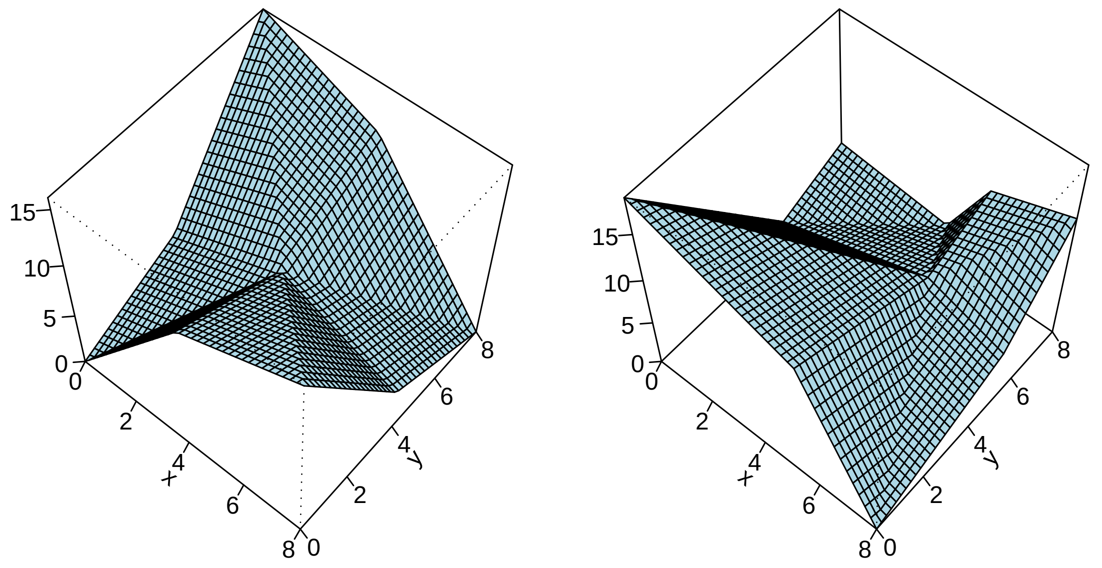


Figure 7.2: Internal landscape perspective (individual interaction payoff surfaces); left: *cyclingRidges*, right: *chaoticRidges*

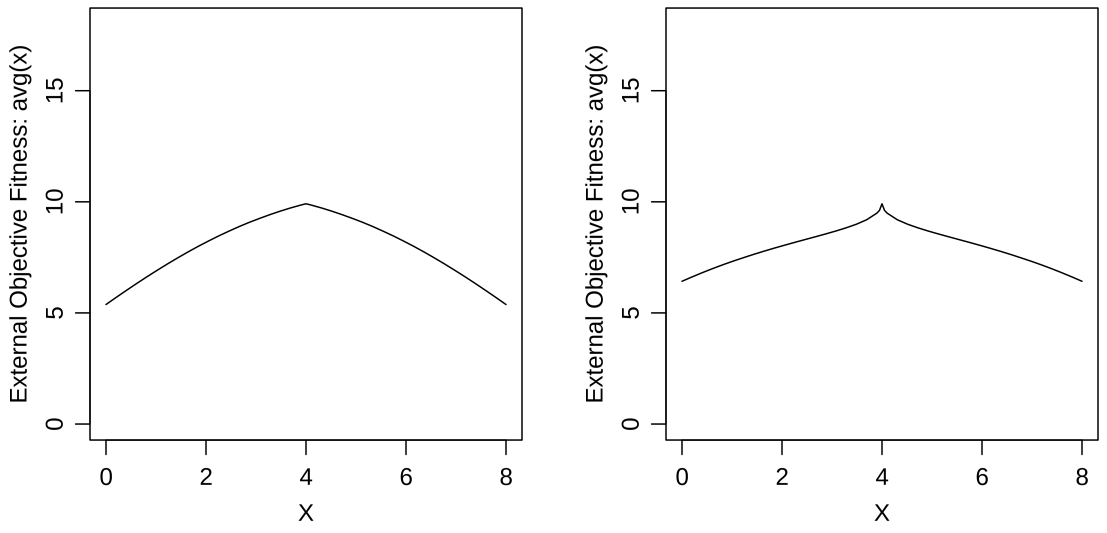


Figure 7.3: External landscape perspective (average payoff curves) for *collapsingRidges* (left) and *expandingRidges* (right).

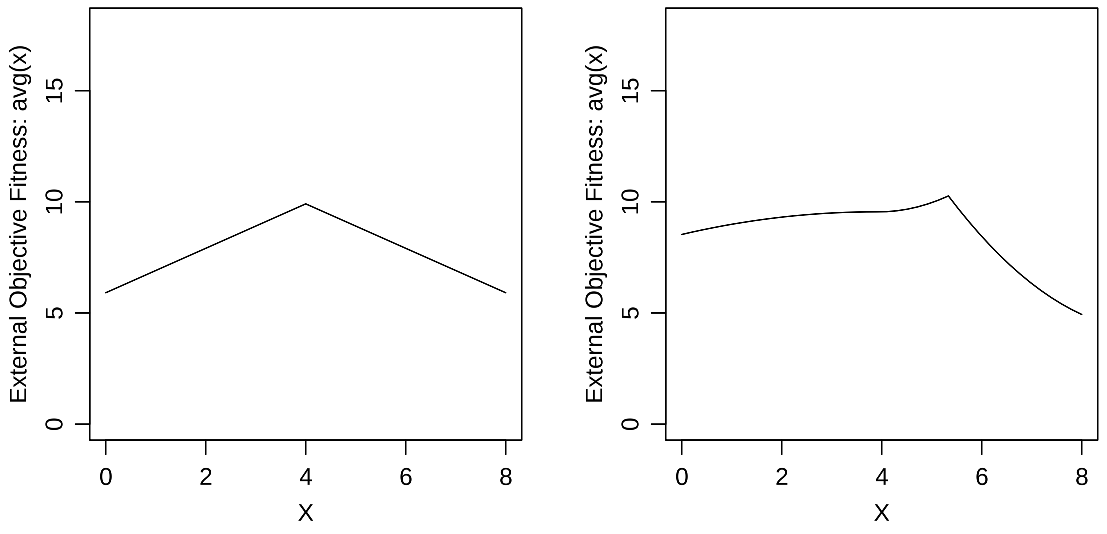


Figure 7.4: External landscape perspective (average payoff curves); left: *cyclingRidges*, right: *chaoticRidges*

however that relationships between the external objective metric and the internal subjective one (for best of generation individuals) can vary widely across these domains.

For the interested reader, the mathematical expressions of these functions are as follows:

$$\mathit{chaoticRidges}_n(x,y) = \begin{cases} n + 2|\frac{2n}{3} - y| - |x-y| & \text{if } (x-\frac{2n}{3})(y-\frac{2n}{3}) \geq 0; \\ \frac{n+x}{3} + |2\frac{n+x}{3} - y| & \text{if } x \leq \frac{n}{2}; \\ -n + 3x + |2(n-x) - y| & \text{if } x \leq \frac{2n}{3}; \\ 3(n-x) + |2(n-x) - y| & \text{otherwise.} \end{cases}$$

$$\mathit{ridges}_n(x,y) = \begin{cases} \mathit{ridges}_n(n-x, n-y) & \text{if } x+y > n; \\ n + x - y & \text{if } x \geq \frac{n}{2}; \\ -n + x + 3y & \text{if } y \geq \frac{n}{2}; \\ 2x - y + f_n(x) & \text{if } y \leq f_n(x); \\ x + f_n^{-1}(y) & \text{otherwise.} \end{cases}$$

$\mathit{cyclingRidges}_n$, $\mathit{collapsingRidges}_n$ and $\mathit{expandingRidges}_n$ are defined as instances of ridges_n for different functions f_n. For $\mathit{cyclingRidges}_n$, $f_n(x) = f_n^{-1}(x) = x$. For $\mathit{collapsingRidges}$, $f_n(x) = x(n-x)$, $f_n^{-1}(x) = \frac{n-\sqrt{n^2-4y^2}}{2}$. For $\mathit{expandingRidges}$, $f_n(x) = \frac{n-\sqrt{n^2-4y^2}}{2}$, $f_n^{-1}(x) = x(n-x)$.

7.2.2 The Algorithm and the Internal Subjective Metric

The CoEA used in this chapter is a variation of the basic one used throughout the dissertation.

It has two populations, one evolving values for x and one for y. Both populations try to maximize payoff. Each population uses a non-overlapping generational model with elitism of 1 and the communication flow is sequential (i.e. populations take turns evolving). Evaluation uses the single-best interaction method, which means an individual is evaluated by having him interact with the best individual in the other population at the previous generation. The payoff from this interaction is assigned as fitness. This defines the *internal subjective metric.* Binary tournament selection is followed by Gaussian mutation with sigma fixed to 0.25, altering each individual with probability 0.75. Each population had a size of 100 and the CoEA ran for 100 generations.

7.3 Results and Dynamics Analysis

Since we are set in a context of applying coevolution for optimization, we will instrument our algorithm from the perspective of change in best individuals throughout the run.

For each experiment we conducted 100 runs and generated plots for all of them. These plots were then visually inspected. If all had the same trend, we picked to show in the paper the one that displays the trend most clearly. If there were several different trends, we similarly picked one from each category, regardless or which category was more heavily represented. Trends of fitness averaged over all runs (not shown) were either presenting nearly no variance (for *collapsingRidges* and *expandingRidges*) or they were obscuring the true phenomena, because they were averaging very different values (for *cyclingRidges* and *chaoticRidges*).

We grouped the results into three categories, based on the relationship between the internal and the external metric. In comparing the metrics, we looked both at the relationship between their respective values and at the nature of their change over time. For each case, we show how the dynamics analysis explains the (otherwise mysterious) trends.

7.3.1 Opposite Monotonic Trends

The Fitness The top row of Figure 7.5 shows a typical run for the *collapsingRidges* function. On the left hand side, the fitness curves plot (only) the points corresponding to the best individual in generations in which X was the active population. For each such point, the internal subjective metric used by the algorithm is the payoff that the best x-individual in that generation obtained from interacting with the best y-individual from the previous (non-plotted) generation. The external objective metric is $avg(x)$. Note that as the external metric is an average over the domain, its range of values is more restricted than that of the internal metric.

In this case the internal subjective metric decreases from its initial value quite fast in the first few generations and then stabilizes for the rest of the run at a value in the middle of its range. This might suggest very poor performance, but in fact the external metric has the exactly opposite trend, i.e. it increases quickly at the beginning and then stabilizes at

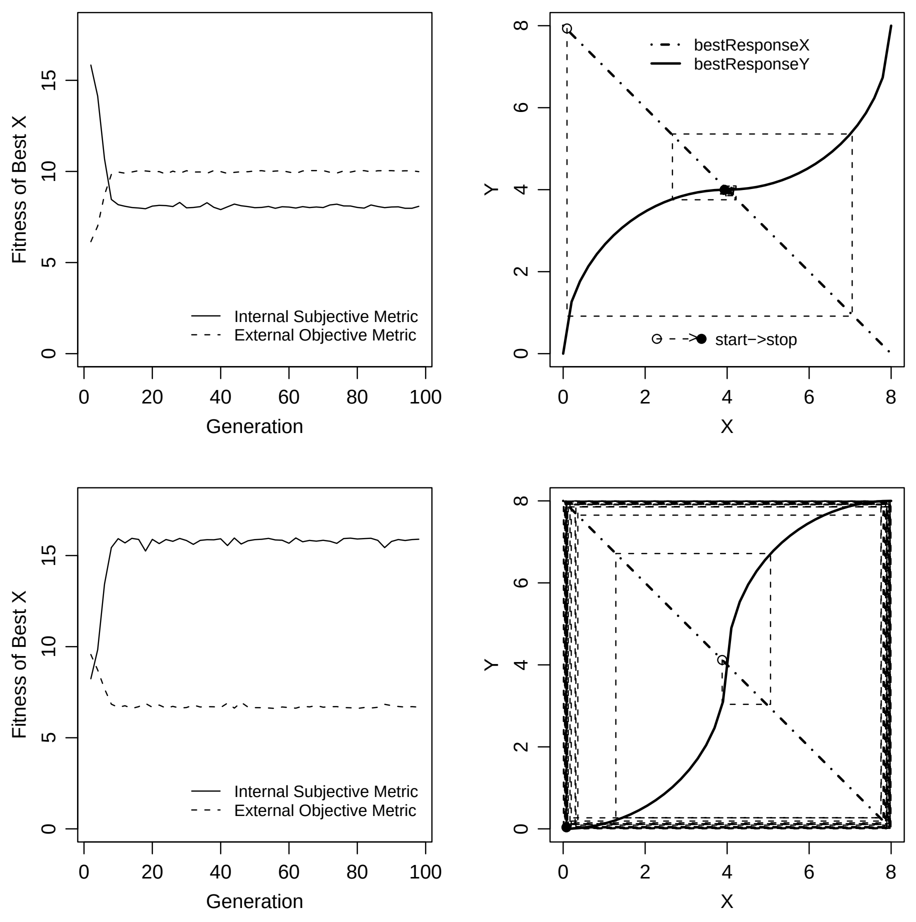


Figure 7.5: *collapsingRidges* (top) and *expandingRidges* (bottom). One selected run each. Left: best-of-generation metrics. Right: best-of-generation space dynamics.

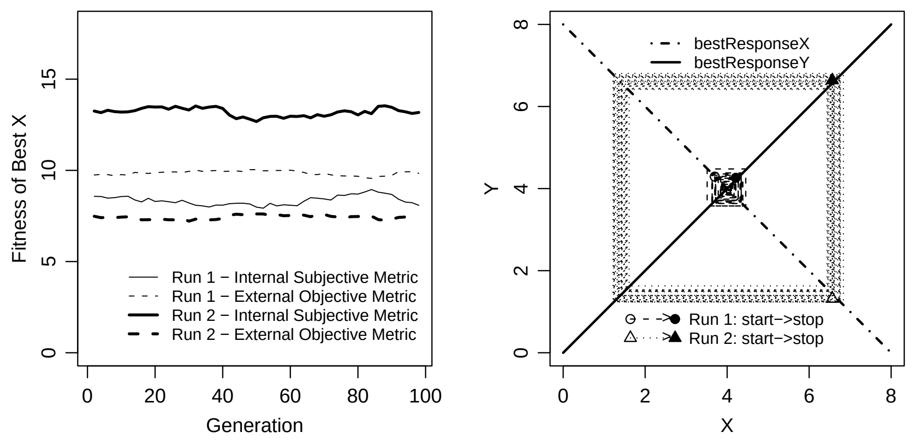


Figure 7.6: Two selected runs for *cyclingRidges*. Left: best-of-generation metrics. Right: best-of-generation space dynamics.

what is actually its maximum value.

A completely reversed situation happens for the *expandingRidges* function. As can be seen in the bottom row of Figure 7.5, initial generations display rapid increase in the internal metric and rapid decrease in the external metric, after which they both stabilize (the internal one at its maximum and the external one at its minimum). What internally looks like progress, from an external perspective is actually regress.

At this point the situation appears confusing. The two functions are very similar from several perspectives. First, the 2D surfaces generated by the payoff of individual interactions have the same degree of ruggedness (same global optima, no local optima, same number of ridges, smooth gradient). Second, the curves generated by the average payoff over all interactions have the same degree of ruggedness (single global optima, no local optima, smooth gradient). Why does the same algorithm have opposite behaviors?

The Dynamics To elucidate this, we look at the dynamics of best individuals from the perspective of their movement across the search space, using the technique introduced in chapter 3. The right hand side of Figure 7.5 shows the trajectories of best individuals across the space (for the same runs) superimposed on the best responses.

To understand the fitness plot for a function, one needs to collectively analyze the trajectory plot, the individual interactions surface plot and the external metric plot for that particular function.

For the system $x \rightarrow bestResponseX(bestResponseY(x))$, that deterministically follows the best-response curves, a point of intersection of the two best-response curves is a fixed point for best individual trajectories. In the case of the *collapsingRidges* function, due to the fact that in this point the absolute value of the slope of the $bestResponseY$ curve is smaller than that of the $bestResponseX$ curve ($|0| < |-1|$), this fixed point has attracting behavior. Regardless of where a trajectory starts (one starting quite far away from this point is shown), it will end up in the proximity of the fixed point. The fitness plot basically reports internal and external metric values for the points of the trajectory on or near the $bestResponseX$ curve. These points move closer and closer to the center of the space. The internal fitness values correspond to the values associated with these points on the 2D surface of individual interactions payoff (left in Figure 7.1). We can see on this surface that moving closer to the center means decreasing values. Once at the center there is only small variation due to stochastic effects. The external fitness values correspond to the values associated with the X coordinate of these points on the external fitness chart (left in Figure 7.3). We can see on this curve that moving from the sides to the center generates increase in values up to the maximum in the fixed point (again, with small perturbations due to stochastic effects). And this is exactly what we see on the fitness plot.

In the case of the *expandingRidges* function, the relationship between the absolute values of the slopes of the best-response curves at the fixed point is reversed ($|\infty| > |-1|$). This makes the fixed point unstable, so trajectories are repelled from it (even when they start very close to it, as in the example run presented). Following the same type of analysis as for *collapsingRidges*, we now understand the shapes of the internal and external fitness curves and why they are different from the previous function (both the 2D surfaces and the external metric curves have the same monotonicities, but the trajectories move in one case from the center to the extremes and vice versa in the other case).

7.3.2 Flat Trends

On the *cyclingRidges* function the two metrics seem to agree in their trends, in the sense that both of them stagnate from beginning to end. However, they disagree in the type of values they report. Figure 7.6 shows two runs superimposed on the same plot(s) (runs are distinguished by line type/width). In one of them the internal metric reports high values for its range while the external metric reports low values for its own range. In the other run, the internal metric shows average values, while the external one is almost at its optimum.

Armed with the technique described above, we can explain these results. The dynamics plot shows that the two best-response curves have the same slope in absolute value. This causes the fixed point to be neither attracting nor repelling. Additionally, it causes the appearance of an infinite number of size two periodic orbits. Trajectories wander about such orbits close to the point where they started (they do not follow a single orbit due to stochastic effects). The periodic orbits intersect the *bestResponseX* curve in points symmetric with respect to the center, and the corresponding fitness values are equal both for the 2D surface of individual payoffs (internal metric) and for the average payoff external metric. This explains the flat trend of the fitness curves.

Whether the internal metric claims better performance than the external or vice versa depends on the point where a particular run's trajectory starts and is cycling close to. Trajectories wandering close to the center will have an internal metric close to 8 (the value at the center) which is in the middle of the 0-16 range and an external metric close to 10, which is the maximum in a range of 6-10. Trajectories wandering closer to the extremes of the space will have the internal metric closer to its maximum of 16 and the external metric closer to its minimum of 6.

7.3.3 Complex Trends

In the three examples examined so far, the trend of each metric throughout the run was monotonic, apart from minor oscillations. The fourth function paints quite a different picture, as can be seen in Figure 7.7 which displays three different runs, each with its own set of plots. Both the internal and the external fitness curves are very rugged, with numerous spikes. For the first run, the spikes are irregular in shape, size and frequency throughout the run. The second run displays two phases, first one of irregular spikes and then one of very regular spikes. During this second phase, the two metrics go up and down synchronously, whereas in the first phase (and more clearly visible in the plot of the third run) when external fitness increases, internal fitness decreases and vice-versa. Finally, the third run displays fairly regular spikes and the two metrics are negatively correlated.

Next, we investigate the dynamics and tie them to the fitness. The right hand side plots in Figure 7.7 show clearly more complex behavior than previously seen. Considering the system $x \rightarrow bestResponseX(bestResponseY(x))$, that deterministically follows the best-response curves, the following can be mathematically proven: 1) it has a repelling fixed point at $(\frac{16}{3}, \frac{16}{3})$; 2) it has periodic orbits of any size; 3) all periodic orbits are sources (i.e. repelling) 4) it has chaotic orbits; and 5) the whole interval $[0, 8]$ is a chaotic attractor. The proof is not included, but for the interested reader it is similar to those for the logistic and tent maps presented in chapters 1 and 3 of Alligood et al. (1996).

Of course, when we introduce stochasticity, as in our CEA, we no longer have clear

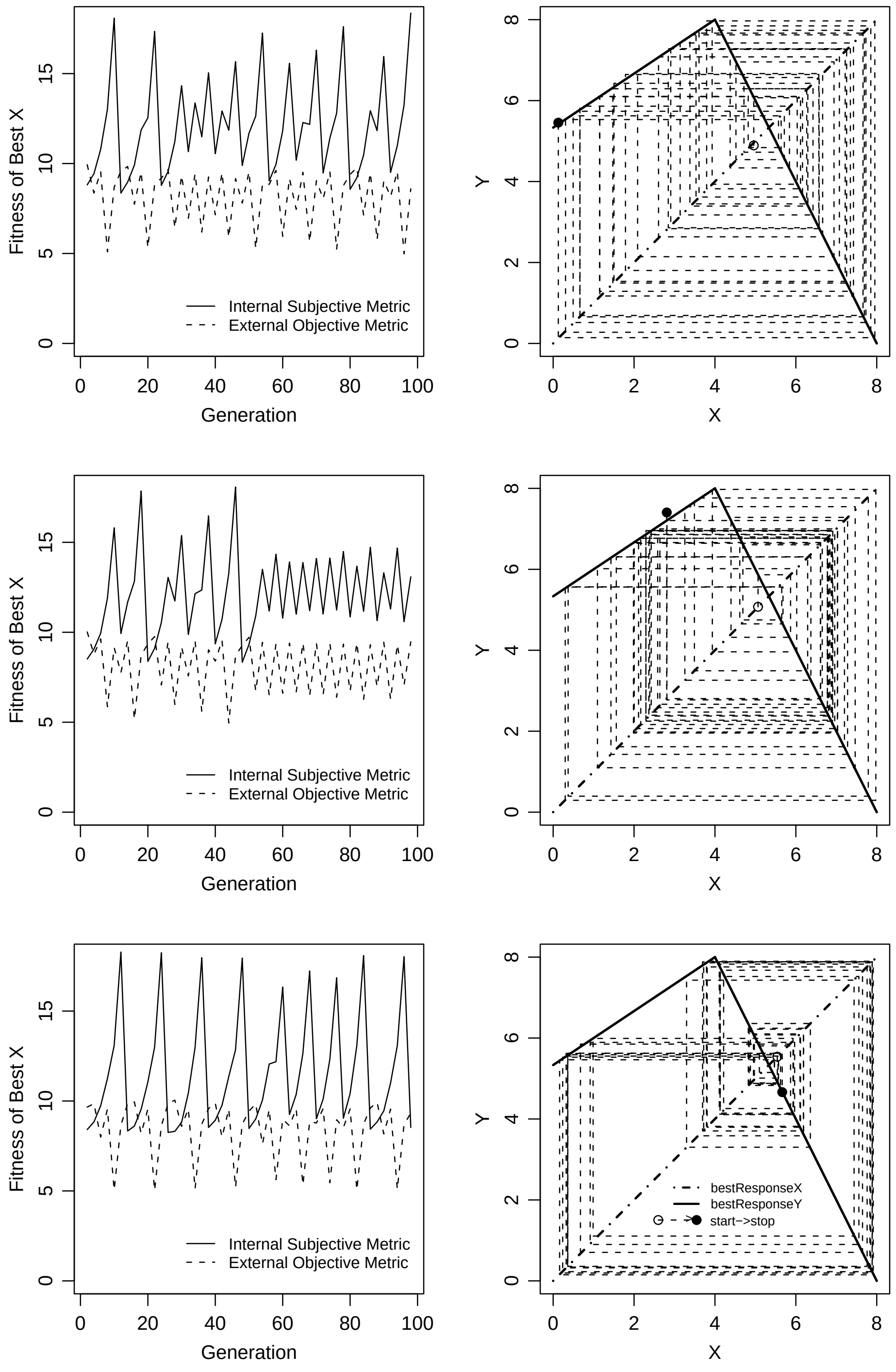


Figure 7.7: Three qualitatively different runs for *chaoticRidges*; left: best-of-generation metrics, right: best-of-generation space dynamics. (To avoid clutter, the legend for the dynamics plots is shown only for the third run - bottom right.)

definitions of the above mentioned dynamical systems concepts. However, visually at least, the CEA using a population size of 100 approximates quite closely the deterministic system.

The first run is the analog of a chaotic orbit for the corresponding deterministic system (which would come infinitesimally close to any point in the $[0, 8]$ interval an infinite number of times, without periodicity). This explains the irregular nature of the fitness curves for this run. The first part of the second run is fairly similar to it, then the trajectory spends some time around a period two orbit, generating the saw-like shape of the fitness. In the third run the trajectory wanders around a higher period orbit, which explains the wider regularly shaped teeth in the fitness plot.

Note first that values in the fitness plot come from points on on near the $bestResponseX$ curve (the main diagonal of the space). Along this diagonal, the minimum payoff is 8 (see Figure 7.2), therefore the internal fitness never goes below this value. At the same time, the maximum average payoff is 10 (see Figure 7.4). Points close to the left/bottom end of the diagonal (x close to 0) have close to maximum internal fitness and high but not quite optimum external fitness. Points near the fixed point have very low internal fitness and close to optimum external fitness. Points close to the right/top end of the diagonal have medium internal fitness and very low external fitness. The rest is somewhere in between.

As can be seen on any of the trajectory plots, $bestResponseX(bestResponseY(x))$ has the following general effects: 1) it moves points near the top/right end to points near the bottom/left end; 2) moves points close to the bottom/left end to points near the fixed point, but above it; 3) moves points near the fixed point, but above it to points near the fixed point, but below it; and 4) moves points near the fixed point, but below it to points further away from the fixed point and closer to the top/right end.

Consulting Figures 7.2 and 7.4, we can see that: case 1) corresponds to some increase in the internal metric and considerable increase in the external one (with respect to their own ranges); 2) entails deep drop in the internal metric and small increase in the external one; 3) gives small increase in the internal metric (gradient to the left of the fixed point is bigger than gradient to the right) and almost no change in the external one; and 4) corresponds to some increase in the internal metric and some decrease in the external one.

This causes a wide range of metric relationships. First of all, the metrics are no longer monotonic throughout the whole run, although they can be for smaller periods of time (e.g. the upward slopes in internal fitness on the spikes for the third run). Second, the two metrics may no longer agree or disagree throughout the whole run, instead there can be periods when they are positively correlated (e.g. second half of second run) and periods when they are negatively correlated (e.g. the teeth in the third run). Moreover, correlation can change sign very often and irregularly (e.g. first run).

7.4 Summary

This chapter showed evidence that the best responses can be an influential problem property in test-based competitive CoEC setups as well, thus showing their cross-area utility and providing a bridge between previously separate CoEC research areas.

Specifically, a new test suite was introduced, featuring problems with competitive-specific relationships between the best responses. These problems, along with the best-individual trajectories tool, were used to gain insight into one of the main causes of failure when using CoEAs for optimization in competitive domains, namely the disconnection between the envisioned external solution concept (optimum average payoff) and the internal behavior of the algorithm.

Chapter 8

Conclusions

8.1 Summary and Contributions

The main hypothesis of this dissertation was that dynamics held the key to understanding coevolutionary algorithms and therefore their analysis should provide insights into the dependency **problem properties + algorithm properties → performance**. Five chapters of **"connected" CoEC analysis** provided evidence supporting this hypothesis in an incremental manner.

Chapter 3 introduced **new tools** for analyzing the dynamics of two-population CoEAs, based on *best-individual trajectories*. Their use exposed a **problem property**, which I named *best-response curves* (or simply best responses). This property proved to have a strong influence on the optimization performance of two "classic" EA parameters, namely population size and elitism. To better study this property, a **test suite** featuring simple best responses with tunable proximity was constructed. At this point, the new, refined hypothesis launched was that the nature of the best responses would influence the performance effects of some CoEA-specific parameters as well.

Indeed, chapter 4 showed this hypothesis to be true for three parameters: collaboration method, communication frequency and communication flow. Analysis was performed on the test suite and applicability of the knowledge gained was successfully tested on additional problems from the literature. All problems consisted of functions available in closed form. Thus, the optima were known, the best responses could be analytically computed and were uniquely defined and continuous.

Chapter 5 extended the analysis to problems with more complex best responses (e.g. discontinuous or local). Two new **test suites** (still functions in closed form) were introduced and the effects of collaboration methods and population size reinvestigated. The knowledge gained from previous chapters helped understand the interaction between the proximity of the best responses and other problem properties (e.g. sizes of basins of attraction).

Thus, the best-response problem property and the dynamics analysis of best-individual trajectories proved useful for understanding the behavior and optimization capabilities of a wide range of variations of a basic coevolutionary algorithm on synthetic problems.

Chapter 6 took one important step further and showed that the insights obtained from the analysis of synthetic problems were transferable to a more realistic problem, featuring a simulation domain (not available in closed form) where the optima were not known and the best responses could only be estimated.

Last but not least, while all aforementioned studies were performed in the context of cooperative compositional setups, chapter 7 proved the strength of the best-response driven analysis extends to competitive test-based setups as well.

8.2 Future Work

This dissertation is a stepping stone. The work presented can be extended to add stepping stones in some directions more than in others.

One major point is that the dissertation has focused on analyzing and understanding CoEAs and not on designing them. Therefore, only little immediate advice for practitioners has been given throughout the thesis. The reason I withheld from that is that the best design choices for a particular problem may not be within the set of algorithm properties that I analyzed. Also, additional relationships between best responses are likely to exist. While this work is the first to connect all four key pieces together, within the span of a PhD dissertation I could only analyze a subset of the properties of those pieces. Additional study is needed to have the wider coverage needed to be successful in practice.

I have limited my analysis to studying one algorithm property at a time and its interaction with up to three problem properties simultaneously. In general, different algorithm properties and different problem properties may interact in non-linear ways, yet it is impossible to vary all of them at the same time, because of combinatorial explosion. This is a general issue without a general solution. All we can rely on is sensitivity analysis and our intuition on which properties would have a higher impact.

I have also limited my analysis to the class of two-population CoEAs. I believe the ideas and tools presented for this class are more easily portable to three or more populations than they are to single-population CoEAs. The main reason why the best responses are so influential on performance is because they are key to the interaction between populations. Extension to three or more populations will certainly put a strain on the analysis, yet the concepts and some heuristics are transferable.

For all the problems I analyzed, the search spaces were subsets of $\mathbb{R}^2$. While the definition of best responses does not depend on the nature of the space, tracking movement through other types of spaces can certainly be more difficult. My suggestion for extension in this respect is to leverage existing research on metrics for these spaces.

Finally, I have only looked at "traditional" CoEAs. In recent years a number of highly-complex CoEAs have been introduced, featuring mechanisms like memory and speciation. While the current research is not directly applicable to understanding those algorithms, it is applicable to deciding whether a complex algorithm is necessary to begin with. This is especially important since this work has shown that what makes a problem difficult for a CoEA is different from what makes a problem difficult for an EA (e.g. modality) and also different from what our intuition may tell us (e.g. nonlinearity).

BIBLIOGRAPHY

BIBLIOGRAPHY

Alligood, K. T., T. D. Sauer, and J. A. Yorke (1996). *Chaos: An Introduction to Dynamical Systems*. Springer.

Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computations. In M. Palaniswami and Y. Attikiouzel (Eds.), *Computational Intelligence: A Dynamic Systems Perspective*, pp. 152–163. IEEE Press.

Angeline, P. J. and J. B. Pollack (1993). Competitive environments evolve better solutions for complex tasks. In *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93)*, pp. 264–270.

Angeline, P. J. and J. B. Pollack (1994). Coevolving high-level representations. In C. G. Langton (Ed.), *Artificial Life III*, Volume XVII, Santa Fe, New Mexico, pp. 55–71. Addison-Wesley.

Axelrod, R. (1989). Evolving new strategies in the iterated prisoner's dilemma. In L. Davis (Ed.), *Genetic Algorithms and Simulated Annealing*, London, pp. 32–41.

Bader-Natal, A. and J. B. Pollack (2004). A population-differential method of monitoring success and failure in coevolution. In *Genetic and Evolutionary Computation Conference*, Volume 3102 of *Lecture Notes in Computer Science*, pp. 585–586. Springer Verlag.

Bergstrom, C. and M. Lachmann (2003). The red king effect: When the slowest runner wins the coevolutionary race. *Proceedings of the National Academy of Science 100*, 593–598.

Blumenthal, H. J. and G. B. Parker (2004). Punctuated anytime learning for evolving multi-agent capture strategies. In *Proceedings of the Congress on Evolutionary Computation – CEC-2004*. IEEE Press.

Bongard, J. and H. Lipson (2005). Nonlinear system identification using coevolution of models and tests. *IEEE Transactions on Evolutionary Computation 9*(4), 361–384.

Bucci, A. and J. B. Pollack (2003). Focusing versus intransitivity: geometrical aspects of coevolution. In E. Cantu-Paz et al. (Eds.), *Genetic and Evolutionary Computation Conference (GECCO 2003)*. Springer.

Bucci, A. and J. B. Pollack (2005). On identifying global optima in cooperative coevolution. In H.-G. Beyer et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005*, New York. ACM Press.

Bull, L. (1997). Evolutionary computing in multi-agent environments: Partners. In T. Baeck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms*, pp. 370–377. Morgan Kaufmann.

Bull, L. (1998). Evolutionary computing in multi-agent environments: Operators. In D. Wagen and A. E. Eiben (Eds.), *Proceedings of the Seventh International Conference on Evolutionary Programming*, pp. 43–52. Springer Verlag.

Bull, L. (2001). On coevolutionary genetic algorithms. *Soft Computing 5*(3), 201–207.

Bull, L. (2005a). Coevolutionary species adaptation genetic algorithms: A continuing saga on coupled fitness landscapes. In C. M. et al. (Eds.), *Proceedings of the 8th European Conference on Advances in Artificial Life – ECAL 2005*, pp. 845–853. Springer.

Bull, L. (2005b). Coevolutionary species adaptation genetic algorithms: growth and mutation on coupled fitness landscapes. In *Proceedings of the Congress on Evolutionary Computation – CEC-2005*. IEEE.

Cartlidge, J. and S. Bullock (2002). Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. In *Proceedings of the Congress on Evolutionary Computation – CEC-2002*, pp. 1420–1425. IEEE.

Cartlidge, J. and S. Bullock (2003). Caring versus sharing: How to maintain engagement and diversity in coevolving populations. In W. Banzhaf et al. (Eds.), *Proceedings of the 7th European Conference on Advances in Artificial Life – ECAL 2003*, Volume 2801 of *Lecture Notes in Computer Science*, pp. 299–308. Springer.

Cartlidge, J. and S. Bullock (2004a). Combating coevolutionary disengagement by reducing parasite virulence. *Evolutionary Computation 12*(2), 193–222.

Cartlidge, J. and S. Bullock (2004b). Unpicking tartan ciao plots: Understanding irregular co-evolutionary cycling. *Adaptive Behavior 12*(2), 69–92.

Chambers, J. M. et al. (1983). *Graphical Methods for Data Analysis.* Duxbury Press.

Chang, M., K. Ohkura, K. Ueda3, and M. Sugiyama (2004). Modeling coevolutionary genetic algorithms on two-bit landscapes: Random partnering. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2004*, pp. 513–524.

Cliff, D. and G. F. Miller (1994). Co-evolution of pursuit and evasion i: Biological and game-theoretic foundations. technical report. Technical report, School of Cognitive and Computing Sciences, University of Sussex.

Cliff, D. and G. F. Miller (1995). Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *European Conference on Artificial Life*, pp. 200–218.

Cliff, D. and G. F. Miller (1996). Co-evolution of pursuit and evasion II: Simulation methods and results. In P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack, and S. W. Wilson (Eds.), *From animals to animats 4*, Cambridge, MA, pp. 506–515. MIT Press.

Coevolution Wiki. Coevolution wiki. http://www2.demo.cs.brandeis.edu/cgi-bin/coec-wiki.

de Jong, E. (2004a). The incremental pareto-coevolution archive. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2004*, pp. 525–536.

de Jong, E. (2004b). Intransitivity in coevolution. In X. Yao et al. (Eds.), *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, Volume 3242 of *Lecture Notes in Computer Science*, Birmingham, UK, pp. 843–851.

de Jong, E. (2004c). Towards a bounded pareto-coevolution archive. In *Proceedings of the Congress on Evolutionary Computation – CEC-2004*, pp. 2341–2348.

de Jong, E. (2005). The MaxSolve algorithm for coevolution. In H.-G. Beyer et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005*, New York. ACM Press.

de Jong, E. and A. Bucci (2006). DECA: Dimension extracting coevolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2006*. ACM Press.

de Jong, E. and J. B. Pollack (2004). Ideal evaluation from coevolution. *Evolutionary Computation Journal 12*(2), 159–192.

De Jong, K. (2006). *Evolutionary Computation: A Unified Approach*. MIT Press.

De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph. D. thesis, University of Michigan.

Eiben, A. E., R. Hinterding, and Z. Michalewicz (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation 3*(2), 124–141.

Ficici, S., O. Melnik, and J. Pollack (2000). A game-theoretic investigation of selection methods used in evolutionary algorithms. In e. a. A. Zalzala (Ed.), *Proc. of CEC2000*. IEEE Press.

Ficici, S., O. Melnik, and J. Pollack (2005). A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE Transactions on Evolutionary Computation 9*(6), 580–602.

Ficici, S. G. (2004). *Solutions concepts in Coevolutionary Algorithms*. Ph. D. thesis, Brandeis University Department of Computer Science.

Ficici, S. G. (2006). A game-theoretic investigation of selection methods in two-population coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2006*, New York, NY, USA. ACM Press.

Ficici, S. G. and J. B. Pollack (1998). Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In C. Adami, R. K. Belew, H. Kitano, and C. Taylor (Eds.), *Artificial Life VI: Proc. of the Sixth Int. Conf. on Artificial Life*, Cambridge, MA, pp. 238–247. The MIT Press.

Ficici, S. G. and J. B. Pollack (2000). Effects of finite populations on evolutionary stable strategies. In D. L. Whitley et al. (Eds.), *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*. Morgan-Kauffman.

Ficici, S. G. and J. B. Pollack (2001). Pareto optimality in coevolutionary learning. In *ECAL '01: Proceedings of the 6th European Conference on Advances in Artificial Life*, London, UK, pp. 316–325. Springer-Verlag.

Ficici, S. G. and J. B. Pollack (2003). A game-theoretic memory mechanism for coevolution. In E. Cantu-Paz et al. (Eds.), *Genetic and Evolutionary Computation Conference (GECCO 2003)*, pp. 286–297. Springer.

Floreano, D. and S. Nolfi (1997). God save the red queen! competition in co-evolutionary robotics. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, USA, pp. 398–406. Morgan Kaufmann.

Funes, P. and J. B. Pollack (2000). Measuring progress in coevolutionary competition. In *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pp. 450–459. MIT Press.

Funes, P. and E. Pujals (2005). Intransitivity revisited coevolutionary dynamics of numbers games. In *Proceedings of the Conference on Genetic and Evolutionary Computation – GECCO-2005*, pp. 515–521. ACM Press.

Goldberg, D. and K. Deb (1990). A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.), *Proceedings of the First Workshop on Foundations of Genetic Algorithms*. Morgan Kaufmann.

Goldberg, D. E. and J. Richardson (1987). Genetic algorithms with sharing for multimodal-function optimization. In J. J. Grefenstette (Ed.), *Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA 1987)*, pp. 41–49. Lawrence Erlbaum Associates.

Hamilton, W. D. (1982). *Population Biology of Infectious Diseases*, Chapter Pathogens as Causes of Genetic Diversity in their Host Populations, pp. 269–296. Berlin: Springer-Verlag.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. In *CNLS '89: Proceedings of the ninth annual international conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent computation*, pp. 228–234. North-Holland Publishing Co.

Hochberg, Y. (1988). A sharper bonferroni procedure for multiple tests of significance. *Biometrika 75*, 800–803.

Hofbauer, J. and K. Sigmund (1998). *Evolutionary Games and Population Dynamics*. Cambridge University Press.

Holland, J. H. (1985). Properties of the bucket brigade. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 1–7. Lawrence Erlbaum Associates, Inc.

Jansen, T. and R. P. Wiegand (2003a). Exploring the explorative advantage of the CC (1+1) EA. In *Proceedings of the 2003 Genetic and Evolutionary Computation Conference (GECCO 2003)*. Springer Verlag.

Jansen, T. and R. P. Wiegand (2003b). Sequential versus parallel cooperative coevolutionary (1+1) EAs. In *Proceedings of the Congress on Evolutionary Computation – CEC-2003*. IEEE Press.

Juillé, H. (1995). Incremental co-evolution of organisms: A new approach for optimization and discovery of strategies. In *Proceedings of the Third European Conference on Advances in Artificial Life*, pp. 246–260. Springer-Verlag.

Juillé, H. and J. B. Pollack (1998). Coevolving the ideal trainer: Application to the discovery of cellular automata rules. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, pp. 519–527. Morgan Kaufmann.

Kauffman, S. and S. Johnson (1991). Co-evolution to the edge of chaos: coupled fitness landscapes, poised states and co-evolutionary avalanches. In C. Langton et al. (Eds.), *Artificial Life II: Studies in the Sciences of Complexity*, Volume X, pp. 325–369. Addison-Wesley.

Koza, J. R. (1991). Evolution and co-evolution of computer programs to control independent-acting agents. In J.-A. Meyer and S. W. Wilson (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior, 24-28, September 1990*, Paris, France, pp. 366–375. MIT Press.

Liekens, A., H. Eikelder, and P. Hilbers (2004). Predicting genetic drift in 2 x 2 games. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2004*, pp. 549–560.

Liekens, A. M. L. (2005). *Evolution of Finite Populations in Dynamic Environments.* Ph. D. thesis, Technische Universitat Eindhoven.

Luke, S. and R. P. Wiegand (2003). When coevolutionary algorithms exhibit evolutionary dynamics. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*. Springer.

McGill, R., J. W. Tukey, and W. Larsen (1978). Variations of boxplots. *The American Statistician 32*(1), 12–16.

Miller, J. H. (1996). The coevolution of automata in the repeated prisoner's dilemma. *Journal of Economic Behavior and Organization 29*(1), 87–112.

Oliehoek, F. A., E. de Jong, and N. Vlassis (2006). The parallel nash memory for asymmetric games. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2006*, New York, NY, USA, pp. 337–344. ACM Press.

Olsson, B. (2001). Co-evolutionary search in asymmetric spaces. *Information Sciences 133*(3-4), 103–125.

Osborne, M. and A. Rubinstein (1994). *A Course in Game Theory.* MIT Press.

Pagie, L. and P. Hogeweg (2000). Information integration and red queen dynamics in coevolutionary optimization. In *Proc. of the 2000 Congress on Evolutionary Computation*, Piscataway, NJ, pp. 1260–1267. IEEE Service Center.

Pagie, L. and M. Mitchell (2002). A comparison of evolutionary and coevolutionary search. *International Journal of Computational Intelligence and Applications 2*(1), 53–69.

Panait, L. (2006). *The Analysis and Design of Concurrent Learning Algorithms for Cooperative Multiagent Systems.* Ph. D. thesis, George Mason University, Fairfax, VA.

Panait, L. and S. Luke (2002). A comparison of two competitive fitness functions. In W. B. Langdon et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pp. 503–511. Morgan Kaufmann.

Panait, L. and S. Luke (2005). Time-dependent collaboration schemes for cooperative coevolutionary algorithms. In *AAAI Fall Symposium on Coevolutionary and Coadaptive Systems.* AAAI Press.

Panait, L. and S. Luke (2006). Selecting informative actions improves cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006.* ACM.

Panait, L., S. Luke, and J. F. Harrison (2006). Archive-based cooperative coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2006.* ACM Press.

Panait, L., K. Sullivan, and S. Luke (2006). Lenience towards teammates helps in cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006.* ACM.

Panait, L., R. P. Wiegand, and S. Luke (2003). Improving coevolutionary search for optimal multiagent behaviors. In G. Gottlob and T. Walsh (Eds.), *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 653–658.

Panait, L., R. P. Wiegand, and S. Luke (2004a). A sensitivity analisys of a cooperative coevolutionary algorithm biased for optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2004.*

Panait, L., R. P. Wiegand, and S. Luke (2004b). A visual demonstration of convergence properties of cooperative coevolution. In *Parallel Problem Solving from Nature – PPSN-2004*, pp. 892–901. Springer.

Paredis, J. (1997). Coevolving cellular automata: Be aware of the red queen. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA. Morgan Kaufmann.

Parker, G. B. and H. J. Blumenthal (2003). Comparison of sample sizes for the co-evolution of cooperative agents. In *Proceedings of the Congress on Evolutionary Computation – CEC-2003*. IEEE Press.

Popovici, E. and K. De Jong (2003). Understanding EA dynamics via population fitness distributions. In E. Cantu-Paz et al. (Eds.), *Genetic and Evolutionary Computation Conference (GECCO 2003)*. Springer.

Popovici, E. and K. De Jong (2005a). A dynamical systems analysis of collaboration methods in cooperative co-evolution. In *AAAI Fall Symposium Series Co-evolution Workshop*.

Popovici, E. and K. De Jong (2005b). Relationships between internal and external metrics in co-evolution. In *Proceedings of the Congress on Evolutionary Computation – CEC-2005*. IEEE.

Popovici, E. and K. De Jong (2005c). Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In H.-G. Beyer et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005*, New York. ACM Press.

Popovici, E. and K. De Jong (2006a). The dynamics of the best individuals in co-evolution. *Journal of Natural Computing 5*(3), 229–255.

Popovici, E. and K. De Jong (2006b). The effects of interaction frequency on the optimization performance of cooperative coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2006*. ACM Press.

Popovici, E. and K. De Jong (2006c). Sequential versus parallel cooperative coevolutionary algorithms for optimization. In *Proceedings of the Congress on Evolutionary Computation – CEC-2006*. IEEE Press.

Potter, M. (1997). *The Design and Analysis of a Computational Model of Cooperative Coevolution*. Ph. D. thesis, George Mason University, Computer Science Department.

Potter, M. and K. De Jong (1994). A cooperative coevolutionary approach to function optimization. In *Proceedings of the Third Conference on Parallel Problem Solving from Nature (PPSN III)*, Jerusalem, Israel, pp. 249–257. Springer.

Price, P. W. (1998). *Biological Evolution*. Saunders College Publishing.

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Reynolds, C. W. (1994). Competition, coevolution and the game of tag. In P. M. R. Brooks (Ed.), *Artificial Life IV*, Cambridge MA, pp. 59–69. MIT Press.

Ridley, M. (1993). *The Red Queen*. Macmillan Pub Co.

Rosin, C. D. (1997). *Coevolutionary search among adversaries*. Ph. D. thesis, University of California, San Diego.

Rosin, C. D. and R. K. Belew (1995). Methods for competitive co-evolution: Finding opponents worth beating. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 373–381. Morgan Kaufmann Publishers Inc.

Rosin, C. D. and R. K. Belew (1997). New methods for competitive coevolution. *Evolutionary Computation 5*(1), 1–29.

Sarma, J. (1998). *An Analysis of Decentralized and Spatially Distributed Genetic Algorithms*. Ph. D. thesis, George Mason University, Fairfax, VA.

Schmitt, L. M. (2003a). Coevolutionary convergence to global optima. In E. Cantu-Paz et al. (Eds.), *Genetic and Evolutionary Computation Conference (GECCO 2003)*, pp. 373–374. Springer.

Schmitt, L. M. (2003b). Theory of coevolutionary genetic algorithms. In M. Guo et al. (Eds.), *Parallel and Distributed Processing and Applications, International Symposium, ISPA 2003*, pp. 285–293. Springer.

Sims, K. (1994). Evolving 3D morphology and behaviour by competition. In R. Brooks and P. Maes (Eds.), *Artificial Life IV Proceedings*, MIT, Cambridge, MA, USA, pp. 28–39. MIT Press.

Skolicki, Z., T. Arciszewski, M. Houck, and K. De Jong (2005). Emerging security patterns: Co-evolution of terrorist and security scenarios. In B. H. V. Topping (Ed.), *Proceedings of 8th International Conference on the Applications of AI to Civil, Structural, and Environmental Engineering.*

Spears, W. (1994). Simple subpopulation schemes. In *Proceedings of the Evolutionary Programming Conference*, pp. 296–307. World Scientific.

Stanley, K. O. and R. Miikkulainen (2002). The dominance tournament method of monitoring progress in coevolution. In W. B. Langdon et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002).* Morgan Kaufmann.

Thompson, J. N. (1994). *The Coevolutionary Process.* The University of Chicago Press.

Wahde, M. and M. G. Nordahl (1998). Coevolving pursuit-evasion strategies in open and confined regions. In C. Adami et al. (Eds.), *Artificial Life VI: Proc. of the Sixth Int. Conf. on Artificial Life*, pp. 472–476. The MIT Press.

Watson, R. A. and J. B. Pollack (2001). Coevolutionary dynamics in a minimal substrate. In L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 702–709. Morgan Kaufmann.

Wiegand, R. P. (2004). *An Analysis of Cooperative Coevolutionary Algorithms.* Ph. D. thesis, George Mason University, Fairfax, VA.

Wiegand, R. P., W. Liles, and K. De Jong (2001). An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In L. Spector (Ed.), *Proceedings of GECCO 2001*, pp. 1235–1242. Morgan Kaufmann. Errata available at http://www.tesseract.org/paul/papers/gecco01-cca-errata.pdf.

Wiegand, R. P., W. C. Liles, and K. A. De Jong (2002). The effects of representational bias on collaboration methods in cooperative coevolution. In *Proceedings of the Seventh Conference on Parallel Problem Solving from Nature*, pp. 257–268. Springer.

Wiegand, R. P. and M. Potter (2006). Robustness in cooperative coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2006*. ACM Press.

Wiegand, R. P. and J. Sarma (2004). Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In X. Yao et al. (Eds.), *8th International Conference on Parallel Problem Solving from Nature - PPSN VIII*, Birmingham, UK, pp. 912–921. Springer.

Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Wikipedia.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin 1*, 80–83.

Williams, N. and M. Mitchell (2005). Investigating the success of spatial coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005*, New York, NY, USA, pp. 523–530. ACM Press.