Solution Concepts in Coevolutionary Algorithms

A Dissertation

Presented to

The Faculty of the Graduate School of Arts and Sciences

Brandeis University

Department of Computer Science

Jordan B. Pollack, Advisor

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

Sevan Gregory Ficici

May, 2004

This dissertation, directed and approved by Sevan Gregory Ficici's committee, has been accepted and approved by the Faculty of Brandeis University in partial fulfillment of the requirements for the degree of:

DOCTOR OF PHILOSOPHY

Adam B. Jaffe, Dean of Arts and Sciences

Dissertation Committee:

Jordan B. Pollack, Department of Computer Science, Chair.

James Pustejovsky, Department of Computer Science

Michael J. Kearns, Department of Computer and Information Science, University of Pennsylvania

Peter M. Todd, Center for Adaptive Behavior and Cognition, Max Planck Institute for Human Development

Dedication

To my dearest wife Hasmik and mother Silva

In loving memory of my father Krikor and grandmother Hermine

and my grandparents Siranoush, Himayak, and Sahag

In memory of the 1.5 million Armenian martyrs of the 1915 Genocide, perpetrated by the Ottoman Turks and denied to this day by the Turkish government.

Acknowledgments

I have the pleasure to acknowledge some of the many people who have inspired, supported, and educated me over the years. Thanks to:

Early influences: Bruce M. Boghosian, Frank Blondale, Gordon Booth, Aleck Brinkman, J. Chapman Flack, Gregory Gargarian, Rick Inatome, Jacqueline Karaaslanian, Tom Roeper, Pete Vanden Bosch. *Colleagues and associates*: Noubar Afeyan, Seth Bullock, Damon Centola, Sune Danø, Ezequiel A. Di Paolo, David B. Fogel, Joshua Knowles, Sandeep Krishna, Kamal Malek, Jason Noble, Ludo Pagie, R. Paul Wiegand, Darrell Whitley. *Current and former members of the Brandeis community*: Richard Alterman, Edward Balkovich, Eric Berkson, R. Scott Buchanan, Paul Buitelaar, Martin Cohn, Alan Danziger, Jeanne DeBaie, Myrna Fox, Andy Garland, Ira M. Gessel, Dan Griffin, Tamar Hajian, Timothy J. Hickey, Harry Mairson, Maja Mataric, Z. George Mou, James A. Storer, Patrick Tufts, David Waltz, Scott Waterman, David Wittenberg. *Good friends*: Clint Bidlack, David Davidian, Rachel E. Feldman, Rose Gulati, A.J. LoCicero, R. Steven Longmuir, Ohannes Salibian, Zarouhi Sarkisian, Taline Voskeritchian.

I am most fortunate to have had Jordan Pollack as my advisor. His enthusiasm, vision, encyclopedic knowledge, and intellectual rigor were all invaluable (especially considering my transition into a new field). I thank him for his kindness, patience, and guidance. I also thank him for assembling a first-rate group of researchers in the DEMO Lab; Jordan and the members (and alumni) of the DEMO Lab are the single most important source of intellectual inspiration for this dissertation. Grateful thanks to my friends:

Ari Bader-Natal, Alan D. Blair, Anthony Bucci, Keki Burjorjee, Paul Darwen, Edwin D. de Jong, Hugues Juillé, Pablo Funes, Gregory Hornby, Kristian Kime, Simon Levy, Hod Lipson, Ofer Melnik, Prem Melville, John Rieffel, Miguel Schneider-Fontán, Elizabeth Sklar, Ayal Spitz, Christian Swinehart, Shivakumar Viswanathan, and Richard A. Watson.

I want to thank especially Shivakumar Viswanathan and Anthony Bucci, for the many discussions and feedback on my work. I owe particular thanks to Ofer Melnik, who was my collaborator on the research presented in Chapter 4. Foremost, I must thank Richard Watson for the many, many hours of challenging, imaginative, and insightful conversation that continually energized my efforts.

Special thanks go to my dissertation committee, Jordan B. Pollack, James Pustejovsky, Michael J. Kearns, and Peter M. Todd, for their insights and helpful feedback on this work.

Finally, my deepest thanks, love, and affection go to my parents, Krikor and Silva Ficici, my grandmother, Hermine Panosyan, and my wife, Hasmik Vardanyan. For their love, patience, and support in all my endeavors, I am forever grateful.

«Ճանաչել զիմաստութիւն եւ զխրատ, իմանալ զբանս հանճարոյ:»

Մեսրոպ Մաշտոց

ABSTRACT

Solution Concepts in Coevolutionary Algorithms

A dissertation presented to the Faculty of
the Graduate School of Arts and Sciences of
Brandeis University, Waltham, Massachusetts

by Sevan Gregory Ficici

Inspired by the principle of natural selection, *coevolutionary algorithms* are search methods in which processes of mutual adaptation occur amongst agents that interact strategically. The outcomes of interaction reveal a reward structure that guides evolution towards the discovery of increasingly adaptive behaviors. Thus, coevolutionary algorithms are often used to search for optimal agent behaviors in domains of strategic interaction.

Coevolutionary algorithms require little *a priori* knowledge about the domain. We assume the learning task necessitates the algorithm to 1) discover agent behaviors, 2) learn the domain's reward structure, and 3) approximate an optimal solution. Despite the many successes of coevolutionary optimization, the practitioner frequently observes a gap between the properties that actually confer agent adaptivity and those expected (or desired) to yield adaptivity, or optimality. This gap is manifested by a variety of well-known pathologies, such as cyclic dynamics, loss of fitness gradient, and evolutionary forgetting.

This dissertation examines the divergence between expectation and actuality in coevolutionary algorithms—why selection pressures fail to conform to our beliefs about adaptiveness, or why our beliefs are evidently erroneous. When we confront the pathologies of coevolutionary algorithms *as a collection*, we find that they are essen-

tially epiphenomena of a single fundamental problem, namely a lack of rigor in our *solution concepts*.

A solution concept is a formalism with which to describe and understand the incentive structures of agents that interact strategically. All coevolutionary algorithms implement *some* solution concept, whether by design or by accident, and optimize according to it. Failures to obtain the desiderata of "complexity" or "optimality" often indicate a dissonance between the implemented solution concept and that required by our envisaged goal.

We make the following contributions: 1) We show that solution concepts are the critical link between our expectations of coevolution and the outcomes actually delivered by algorithm operation, and are therefore crucial to explicating the divergence between the two, 2) We provide analytic results that show how solution concepts bring our expectations in line with algorithmic reality, and 3) We show how solution concepts empower us to construct algorithms that operate more in line with our goals.

Preface

The research presented in Chapter 4 (and the authorship of Section 4.9) was done in collaboration with Ofer Melnik. The observation that coevolution can be regarded as a form of multi-objective optimization, which we use in Chapter 7, was made in collaboration with Richard A. Watson.

The research and portions of text presented in this dissertation appear in the following publications:

- **Chapter 3** S.G. Ficici and J.B. Pollack. A Game-Theoretic Approach to the Simple Coevolutionary Algorithm. In M. Schoenauer, et al, editors, *Parallel Problem Solving from Nature VI*, 467–476. Springer, 2000.

- **Chapter 4** S.G. Ficici, O. Melnik, and J.B. Pollack. A Game-Theoretic Investigation of Selection Methods Used in Evolutionary Algorithms. In Zalzala, et al, editors, *Proc. of 2000 Congress on Evolutionary Computation*, 880–887. IEEE Press, 2000.

- **Chapter 5** S.G. Ficici and J.B. Pollack. Game Theory and the Simple Coevolutionary Algorithm: Some Results on Fitness Sharing. In R. Heckendorn, editor, *2001 Genetic and Evolutionary Computation Conference Workshop Program*, 2–7. ISGEC, 2001.

- **Chapter 6** S.G. Ficici and J.B. Pollack. Effects of Finite Populations on Evolutionary Stable Strategies. In L.D. Whitley, et al, editors, *Proceedings of* *the 2000 Genetic and Evolutionary Computation Conference*, 927–934. Morgan-Kaufmann, 2000.

- **Chapter 7** S.G. Ficici and J.B. Pollack. Pareto Optimality in Coevolutionary Learning. In J. Kelemen and P. Sosík, editors, *Sixth European Conference on Artificial Life (ECAL 2001)*, 316–325. Springer, 2001.

- **Chapter 8** S.G. Ficici and J.B. Pollack. A Game-Theoretic Memory Mechanism for Coevolution. In Cantú-Paz, et al, editors, *2003 Genetic and Evolutionary Computation Conference*, 286–297. Springer, 2003.

In addition, there exists a collection of relevant publications that are nevertheless either omitted from this dissertation or discussed only briefly:

- S.G. Ficici and J.B. Pollack. Coevolving Communicative Behavior in a Linear Pursuer-Evader Game. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S.W. Wilson, editors, *Proceedings of the Fifth International Conference of the Society for Adaptive Behavior*, 505–514. MIT Press,1998.

- S.G. Ficici and J.B. Pollack. Challenges in Coevolutionary Learning: Arms-Race Dynamics, Open-Endedness, and Mediocre Stable States. In C. Adami, et al, editors, *Proc. of the Sixth Conf. on Artificial Life*, 238–247. MIT Press, 1998.

- S.G. Ficici and J.B. Pollack. Statistical Reasoning Strategies in the Pursuit and Evasion Domain. In D. Floreano, J.-D. Nicoud, F. Mondada, editors, *Fifth European Conference on Artificial Life*, 79–88. Springer, 1999.

This document is Brandeis University Computer Science Department Technical Report CS-03-243.

Contents

List of Tables

List of Figures

Chapter 1

Solution Concepts in Coevolution

Name the greatest of all the inventors: Accident. Mark Twain [1]

1.1 Introduction

Evolutionary computation (EC) has by now amply established its ability to discover novel and effective solutions to many difficult static optimization problems. Given this success, we may expect Darwin's principle of natural selection—the basis of all EC—to hold particular promise for *co*-evolutionary computation. *Coevolution* is a process of mutual adaptation that occurs amongst a set of agents that interact strategically in some domain. The outcomes of agent interaction reveal a reward structure that guides evolution towards the discovery of increasingly adaptive agent behaviors. This process of adaptation, however, is self-referential: When agents evolve to exploit reward opportunities provided by other agents, they create yet newer opportunities for others to exploit. In this way coevolution provides a dynamic ecology, continually generating new possibilities for evolutionary learning. In particular, the ferment of

[1] *Mark Twain's Notebook*, Chapter 33 "Back in America."

coevolution is believed to induce a (potentially open-ended) *arms race* towards greater and greater agent complexity.

Indeed, the seminal works of Hillis [91] and Sims [185] give credence to the expectation that coevolutionary systems are naturally poised to yield complexity from simple initial conditions. Nevertheless, the gap between the hypothesized potential of coevolutionary algorithms and realized practice remains substantial—the successes of coevolution (of which there are now many) are balanced by frequently encountered and irksome pathologies (of which there are also many). These pathologies do not merely deprive us of a satisfying result (as might a local optimum), but more importantly they appear to violate our intuitions of how selection pressure in coevolution is supposed to operate. That is, the problem is not that the coevolving agents are failing to adapt; indeed, we often find that the coevolving agents adapt to selection pressures very effectively—local optima are not the problem. The problem is, rather, that the behaviors that we expect (or wish) to be adaptive are not; further, the behaviors that are adaptive for the agents are, for us, undesirable—they are either uninteresting or do not conform to some *a priori* goal we seek. Thus, the arms-race narrative is left somewhere between fact and fiction.

This dissertation examines the divergence between expectation and actuality in coevolutionary algorithms—why selection pressures fail to conform to our beliefs about adaptiveness, or (more properly) why our beliefs are evidently erroneous. To grapple with this incongruity between expectation and actuality, coevolution researchers have appropriated or invented a myriad of terms, such as *cyclic dynamic*, *mediocre stable-state*, *collusion*, *forgetting*, *disengagement*, and *focusing*. These terms and the pathologies they denote have been discussed at length in the research literature; many notable contributions have been made to better understand them. Nevertheless, this dissertation shows that, when we confront the pathologies of coevolutionary algorithms *as a collection*, we find that they are essentially epiphenomena of a single fundamental problem, namely a general lack of rigor in our *solution concepts*.

A solution concept is a formalism with which to describe and understand the incentive structures of agents that interact strategically; we import the notion of solution concept from *game theory* [83]—the mathematical field that concerns strategic interaction. As *evolutionary* game theory [134] points out, evolving agents need not be cognizant of their incentives to act on them—Darwinian selection will steer the agents according to the reward structures extant in the ecology. All coevolutionary algorithms implement *some* solution concept, whether by design or by accident, and optimize according to it; failures to obtain the desiderata of "complexity" or "optimality" often indicate a dissonance between the implemented solution concept and that required by our envisaged goal.

This dissertation makes three distinct types of contribution. The first is thematic: We emphasize the use of solution concepts to discipline our thinking about coevolution. We show that solution concepts are the critical link between our expectations of coevolution and the outcomes actually delivered by algorithm operation, and are therefore crucial to explicating the divergence between the two. The second type of contribution is analytic: We show how solution concepts bring our expectations in line with algorithmic reality—we cannot expect to obtain complexity when agents are not incited to become complex. Specifically, we use game theory to show that certain commonly used algorithmic constructs are incapable of delivering particular types of results that we may expect or desire. Further, we mathematically prove that certain solution concepts do not engender monotonic coevolutionary progress, regardless of implementation details; other solution concepts are shown to be conducive to monotonic progress, some with specific implementation requirements. Finally, the third type of contribution concerns algorithmic invention: We show how solution concepts empower us to construct algorithms that operate more in line with our goals. Specifically, we introduce and test algorithms designed around solution concepts such as Pareto optimality and Nash equilibrium that take a principled and innovative approach to the pathologies associated with coevolution.

1.2 Solution Concepts in Search

Fundamental to all search problems is the notion of a *solution concept*. Whatever properties our problem domain may possess, and however we embed that domain into a search space, we require a solution concept to indicate which locations in the search space—if any—constitute *solutions* to our problem. A solution concept thus partitions a search space into two classes: Solutions and non-solutions. Typically, the two classes are distinguished in a systematic way—by some number of measurable properties that are present or absent in class members; in general, however, any arbitrary binary partition constitutes a valid solution concept. (Real-world search problems are usually difficult enough that we must be content with *satisficing* rather than optimization; nevertheless, any satisficing problem can be formally stated as an optimization problem, where a solution is any result that we deem "good enough.") Thus, a search space can have an exponential number of solution concepts applied to it. When we apply a particular solution concept to a search space, we obtain a particular *search problem*.

While solution concepts are intrinsic to search problems, they must nevertheless be implemented by *search algorithms*. We can choose from any number of algorithms to solve a given search problem. Some algorithms may be more or less efficient than others with respect to our search problem, yet all of the algorithms must implement the same solution concept to be *consistent* with our search problem; algorithms that implement different solution concepts solve different search problems, by definition (even when used on the same search space). Solution concepts thus form the nexus of problems and processes to solve them, and are therefore of very great importance. Many search problems, like the optimization of the function $f : \mathbb{R} \to \mathbb{R}$, require such a plain expression of optimality that the notion of "solution concept" appears unworthy of special attention. Once we broaden our scope, however, we encounter other kinds of search problems, like those found in multi-objective optimization or optimization in dynamic environments, that require much more sophisticated notions of optimality to convincingly solve. The class of search problem that arguably requires the most sophisticated notion of optimality, however, contains those problems with *interactive domains*. Perhaps the most familiar examples of interactive domains are board games such as chess, backgammon, and the like. Yet, interactive domains abound in the study of biology, sociology, economics, and other distributed systems of agents; indeed, interactive domains motivate an entire branch of mathematics, known as *game theory*.

1.3 Recognizing Solutions: The Secondary Search Problem

To solve a search problem in practice, we need a way to recognize solutions; thus, we require a solution concept built upon detectable attributes that differentiate solutions from non-solutions. And, to improve the efficiency of our search, we hope that the solution concept affords a way (often a heuristic) to assess locations in the search space with an even finer grain—a metric of goodness that can provide a reliable gradient for search to follow.

In most search problems, the properties that identify a solution (or suggest that one non-solution is preferable to another) are immediately observable. For example, a sequence of moves is a solution to Rubik's cube if it takes the cube from some initial configuration to a configuration where each of the cube's six sides are of a single color; here, the elements of the search space are behaviors (sequences of moves) that act upon (or interact with) an external object, namely the cube. There exists a contrasting class of search problem, however, where the elements of the search space do not act upon an external object or environment, but rather *interact with each other*; this is the class of search problems with interactive domains. In such a domain, the properties we need to observe to identify solutions are revealed only through such interaction. More important, we may not know *a priori* which interactions are necessary or sufficient to determine that an element of the search space is or is not a solution, or whether one element is preferable to another—our solution concept is silent on this issue. For this reason, the primary (intended) search problem entails a secondary *co-problem*; the objective of this collateral search effort is to discover which interactions promote effective search in the primary problem. Of course, this secondary search problem must have a solution concept, as well.

Many difficulties encountered in conventional coevolutionary systems can be attributed to a lack of appreciation for the secondary search effort; in particular, conventional systems use the same solution concept for both search problems. That is, the implicit assumption is that if search space element A is better than B at solving our primary problem, then A is better at solving our secondary problem, as well. To be more concrete, a conventional coevolutionary algorithm believes that a perfect strategy for a game (say chess) is also the perfect metric for an opponent's ability; yet, we know that Kasparov will beat both a beginner as well as an intermediate player, thus providing no indication that the intermediate player is better than the beginner.

(We may choose to examine the entire unfolding of the game, rather than just the outcome, to detect some distinguishing characteristics, but then we are effectively changing the game. Even if we do examine the entirety of game play, performance against an expert may still not reveal differences in skill between two sufficiently poor players.) That experts can make poor sources of gradient is demonstrated by Angeline and Pollack [9] in Tic-Tac-Toe.

1.4 Foundations

1.4.1 The Conventional Coevolutionary Algorithm

All evolutionary algorithms, whether *co*-evolutionary or not, are population-based search methods inspired by the concept of natural selection. Before we can use an evolutionary algorithm (EA), we must first choose a representation, or *evolutionary substrate*, that is appropriate for the search problem at hand; one canonical substrate is the bit-string. Next, we require appropriate "genetic" *variation operators*, for example *crossover* and *mutation*. Each possible configuration of our evolutionary substrate maps to some location in our search space. The variation operators connect locations in the search space to form a neighborhood structure. Our search effort is made more efficient when this neighborhood structure combines with our solution concept to yield a smooth fitness gradient, or landscape.

A generic evolutionary algorithm operates roughly as follows:

1. Initialize population
2. Evaluate each member of the population and assign a rating
3. If halting criterion is met, then stop; otherwise. . .

4. Select population members for breeding according to their ratings

5. Generate "offspring" from selected "parents" with variation operators

6. Insert offspring into population, and go to Step 2

Typically, each member of our initial population is some random substrate configuration, for example a random bit-string. The evaluation process examines each individual in the population and rates it according to some objective function that subsumes our solution concept; these ratings are often called "fitness values," but this is an abuse of the term, which we will define precisely. Next, our *selection method* picks members of the population for the reproduction of "offspring." Individuals with better ratings are more likely to become "parents" than individuals with worse ratings. The *fitness* of an individual is the number of offspring it procreates. If the generation of offspring is sexual, then recombinative operators randomly blend facets of a parent pair to produce an offspring; recombination is possibly followed by additional random mutations to the offspring. If offspring are generated asexually, then a single parent is involved in the production of an offspring and mutation is the only variation operator used. Evolutionary algorithms typically use fixed population sizes. Thus, once offspring are generated, space must be made for them in the population. On the one extreme, the entire population is replaced *en masse* by offspring; this is a *generational* algorithm. On the other extreme, only a few offspring are generated at a time, leaving most of the population intact; this approach is a *steady-state* algorithm. We then iterate the algorithm another cycle, with the hope that some of the offspring are improvements over their parents. Each step of this process is the subject of intense research and many variations of the basic formulation exist; for general references to evolutionary algorithms see Goldberg [85], Mitchell [141], Banzhaf, et al, [21], and Bäck, et al, [18, 19].

Co-evolutionary algorithms are primarily distinguished from "ordinary" evolution by the evaluation process. In coevolution, as we point out above, an individual can only be evaluated by having it interact with other evolving individuals; these interaction partners are members of the same population or different populations, depending upon the search problem. (That coevolution may involve more than one evolving population is often claimed to be a secondary distinction from ordinary evolution. Nevertheless, multiple populations can be used in ordinary evolution, as well; these are so-called *island models*. Occasionally the claim is made that coevolution *intrinsically requires* more than one population; one need look no further than the seminal works of Axelrod [14] and Sims [185] for examples single-population coevolution.)

Regardless of the number of populations, the most conventional pattern of interaction is to have each member of each population interact with all other individuals that can serve as partners. For example, if we have a symmetric two-player game and a single population of n individuals, then we will have $n(n-1)/2$ distinct interactions; if we have an asymmetric two-player game and two populations of size m and n respectively, then we will have mn distinct interactions. This pattern of interaction is termed *complete mixing* by the evolutionary game theory literature [134]. Also similar to evolutionary game theory, an individual's rating (or score) is the average outcome over all its interactions (sometimes plus a constant c to keep all scores non-negative—fitness cannot be less than zero, after all). Other patterns of interaction are discussed by Angeline and Pollack [9], Sims [185], Panait and Luke [157], Hillis [91], and Pagie and Hogeweg [156], with the latter two using *spatial* patterns of interaction.

As it stands, the conventional coevolutionary algorithm lacks consideration of the secondary search effort. Those individuals that are most successful in the primary search effort will obtain the most offspring; thus, these successful individuals also have the most influence on testing, since the very same offspring are used as interaction partners.

1.4.2 Competitive vs. Cooperative Coevolution

The work of Potter [165] (see also, Potter and De Jong [167]) introduces a coevolutionary methodology called *cooperative coevolution* (not to be confused with *cooperative game theory*). Many real-world problems are too difficult to optimize when treated monolithically; often, however, such problems can be decomposed into a collection of easier sub-problems, the solutions to which work together to solve the original, larger problem. Cooperative coevolution aims to solve a difficult problem X by coevolving an effective set of solutions to a decomposition of X; if X is decomposed into n sub-problems, then n reproductively isolated populations are coevolved to "cooperatively" solve the problem X. The less the sub-problems interact with each other, the more effective cooperative coevolution will be. Thus, results pertaining to Kauffman's *NK landscapes* [108] and Watson and Pollack's *hierarchical if-and-only-if* [200] function apply. From a game-theoretic perspective, if the sub-problems interact (such that the solution to one sub-problem is globally effective only in *combination* with certain solutions to another sub-problem) then our original problem X has features of a *coordination game* [82] (we discuss *symmetric* coordination games in Section 3.7.1); more generally, the domains with which cooperative coevolution is concerned can be cast as variable-sum games for n players.

The algorithmic facet of cooperative coevolution that most distinguishes it from more "conventional" coevolution is its ability to dynamically adjust the problem's decomposition. An example given by Potter [165] concerns the number and connectivity of hidden nodes in a neural network; the number of populations corresponds to the number of hidden nodes, and population members determine connectivity to fixed input and output layers. If coevolutionary progress becomes unsatisfactory, then new populations can be added; populations found to contribute little towards the over-all solution can be eliminated. In this way, the system does not require an *a priori* decomposition of the problem. Note, however, that this method of dynamic decomposition does require all populations to adhere to a pre-specified *interface* (or meta-interface) that governs interaction between components (even if the components play different functional roles).

Cooperative coevolution is almost always contrasted with so-called *competitive coevolution*, which is typically defined as coevolution applied to a zero-sum game. Nevertheless, the algorithmic framework of "competitive coevolution" is not specific to zero-sum games; indeed, the framework is often applied to variable-sum games, most famously the Iterated Prisoner's Dilemma (e.g., Axelrod [14] and Lindgren [128]). Further, both "styles" of coevolution (i.e., competitive and cooperative) can use multiple, reproductively isolated populations; both can use similar patterns of inter-population interaction, similar diversity maintenance schemes, and so on. Aside from the novel problem-decomposition scheme of cooperative coevolution, the most salient difference between cooperative and competitive coevolution resides primarily in the game-theoretic properties of the domains to which these algorithms are applied. Consequently, game theory is profitably related to coevolutionary algorithms of either description (see, for example, Ficici, Melnik, and Pollack [58], and Ficici and Pollack [62] and the subsequent work—on cooperative coevolution—of Wiegand, Liles, and De Jong [206, 207]). Thus, the oft-made cooperative-competitive dichotomy is less sharp than conventionally portrayed.

1.4.3 Assumptions

When we perform coevolutionary optimization, we make very weak assumptions about the information we begin with and expect to obtain. Specifically, we assume the following: 1) We are initially ignorant of the gamut of behaviors available to an evolving agent—indeed, discovery of the behavior space is a central task of coevolutionary optimization; 2) We are initially ignorant of the outcomes obtained by the possible behaviors—we incrementally discover the outcomes as we discover behaviors and have them interact; 3) We can treat the evolving individual as a "black box"—we need not assume the ability to observe its behavior; we merely require knowledge of the *outcome* of its behavior; 4) We cannot definitively establish the identity of a behavior exhibited by an evolving agent without exhaustive testing—thus, we cannot easily establish that two agents will behave identically in all situations unless they have identical genotypes and the genotype-phenotype mapping is deterministic; 5) There is a many-to-one mapping from genotype space to phenotype space; thus, we cannot assume that individuals with different genotypes must behave differently. These assumptions are radically different from those made by the related fields of research that we discuss below in Section 1.5.

1.4.4 Dynamics vs. Goals

To comprehend the operation of a coevolutionary algorithm, we can either attempt to understand it purely as a dynamical system, or we can attempt to understand its dynamics with respect to some desired result or behavior. The approach we choose depends upon the purpose for which the algorithm is constructed. The former approach is appropriate if, for example, the algorithm is meant to model specific mechanisms or processes that we hypothesize to operate in nature. In this case, the behavior of the algorithm is viewed as a consequence of algorithm design. In contrast, the latter approach is appropriate if we begin with a particular outcome in mind and seek to build a mechanism capable of achieving or approximating that outcome. In this case, the design of the algorithm is viewed as a consequence of the desired behavior. In this dissertation, we use this second approach to consider coevolutionary algorithm dynamics; we presume that the coevolutionary algorithm performs optimization, and the notion of optimality is specified by a solution concept.

Our emphasis on equilibria is not to say that we are uninterested in coevolutionary dynamics. Indeed, the notion of equilibrium can scarcely be discussed without reference to dynamics. Nevertheless, since the result we seek from a coevolutionary algorithm is an equilibrium state, our consideration of dynamics must be calibrated accordingly. Certainly, in a real coevolutionary algorithm, factors such as mutation and finite population size perpetually keep the evolving population from true equilibrium; further, if we care to use a coevolutionary algorithm in an open-ended domain, then by definition we cannot achieve a permanent equilibrium. Thus, one can argue that we are left only with dynamics, and that is where our interest must lie, after all. Nevertheless, even under these circumstances, we care that our dynamics are informed by our intended solution concept and seek to satisfy it; if our dynamics do not deliver the desired outcome in the absence of confounding factors (such as open-endedness, mutational noise, and finite populations), then their justification for our purposes appears dubious.

1.4.5 Adaptation vs. Optimization

De Jong [47, 48] points out that the design intent behind the *canonical* genetic algorithm (GA), as defined by Holland [94], is to investigate natural evolution—not perform fitness-function optimization. Indeed, the canonical GA does not optimize the fitness function *per se*, but rather optimizes expected cumulative reward. (This associated, but different, task is similar to the *k-armed bandit problem*: We have k slot machines and N coins; each of these slot machines has a different expected rate of reward that is unknown to us. Our task is to apportion our N coins amongst the k machines to optimize our expected cumulative return. Thus, we have finite resources with which to *explore* the rates of return of the k machines and *exploit* the machine with the highest observed return. Further details about the bandit problem and its relevance to GA operation are found in Holland [94], De Jong [46], and Goldberg [85].) De Jong further points out that, when we refashion the GA to emphasize fitness-function optimization (e.g., by adding elitism), analyses of the canonical GA may no longer apply; instead, new analyses are required to understand the behavior of the altered GA. While De Jong does not explicitly invoke the notion of solution concept in his remarks, his concerns about GA praxis are echoed in spirit by our approach to coevolutionary algorithms; most particularly, in Chapters 4 and 5 we show that certain alternative selection methods and fitness sharing change the behavior of an otherwise standard coevolutionary algorithm such that conventional expectations of algorithm behavior are no longer met.

De Jong also remarks that natural evolution is a process of adaptation within a *dynamic environment*. Harvey [89] expresses a similar point regarding evolutionary learning algorithms for adaptive autonomous agents. In particular, Harvey observes that an agent's environment may be dynamic, and may contain other adaptive agents. Broadly speaking, both De Jong and Harvey are reflecting upon what is essentially a coevolutionary framework. Further, they agree that such a framework does not offer a global optimization objective.

For example, Harvey states that if an agent's environment changes only incrementally over a period of time, then an *adaptation* of the agent's current behavior is likely a more appropriate response than beginning an entirely new optimization effort *tabula rasa*; thus, evolutionary learning algorithms appear well-suited for such an agent. But, Harvey further comments that a series of such local adaptations may be the only sense in which the agent "optimizes," and therefore we cannot easily imagine a notion of global or long-term optimization for our agent. Nevertheless, as we will see below and later in this dissertation, there do exist on-line learning algorithms that can be understood to optimize in the limit; these algorithms do not necessarily require the environment to be stationary (see Section 1.5), and we will show in Chapter 9 that there exists a sense in which a coevolutionary system can monotonically improve, according to a global metric, even in open-ended domains where future innovations cannot be anticipated (which implies a non-stationary environment). Thus, while we are not interested to debate whether natural evolution performs adaptation or optimization, we do argue that an appropriately constructed coevolutionary algorithm can optimize. There exists a common belief that objective metrics of goodness are an ill-conceived proposition in coevolution (see, for example, Freund and Wolter [78] who equate frequency-dependent fitness—which obtains in coevolution—with the lack of an "absolute goal"); we show that this belief does not transfer to algorithms designed around appropriate solution concepts.

1.4.6 Expression of a Solution

When we apply evolution to a static single-objective problem, the most fit individual of the final population is considered the "solution"; if more than one individual in the final population achieves this maximal fitness, then we are indifferent as to which is selected (even if these individuals are genetically or phenotypically distinct). When we apply evolution to a static *multi*-objective problem, then the solution that is delivered is typically the *Pareto front*, which is a *set* of non-dominated feasible members of a trade-off surface [76]; these individuals are either in the evolving population or in an archive of some sort (we discuss multi-objective optimization more in Chapters 2 and 7). The practice of coevolution follows, for the most part, the convention that the solution to a coevolutionary problem will be a single "best" individual (or possibly a set of individuals where multiple populations that coevolve with each other each contribute their best individual, e.g., Potter and De Jong [167]). Indeed, the efficacy of some coevolutionary algorithms and the correctness of some analyses are predicated on the existence of a single, globally perfect individual behavior (e.g., Rosin and Belew [174] and Schmitt [183, 182]).

In their experiments with robotic pursuit-and-evasion, Nolfi and Floreano [148] observe that the notion of a single global winner may be inadequate. Other researchers explicitly embrace the belief that a solution may need to be expressed as a *collection* of evolved agents (all from the same population), where no single agent in the collection can be considered a proper solution on its own. (In the case of multi-population coevolution, each population may need to contribute such a collection.) For example, Darwen [41] (also Darwen and Yao [42]) uses a speciating evolutionary algorithm to evolve a collection of strategies for the Iterated Prisoner's Dilemma (IPD). A novel gating mechanism is used (once evolution is complete) to determine which strategy in the collection likely offers the most effective response against a particular opponent at every point during a 100-iteration game; this procedure is shown to improve robustness. We note that the "output" of this framework is not entirely a black-box: The gating mechanism is hand-built and agent behaviors are observable. Nevertheless, the solution obtained is clearly not in the form of a single best individual.

Finally, evolutionary game theory [134] regards the state of the entire population as a solution (provided the state is at a stable fixed-point). For example, in the Hawk-Dove game, the Nash equilibrium solution is obtained when the population reaches a particular ratio of Hawks and Doves; this is known as a *polymorphism.* Thus, the solution here is not a single individual (e.g., a Hawk), nor is it a collection of unique individuals (e.g., a Hawk and a Dove); rather, the solution is a specific *distribution* over the two strategy types. We will discuss evolutionary game theory and the Hawk-Dove game in detail in Chapters 4, 5, and 6. In the formalism we introduce in Chapter 2, we use a construct called a *behavior complex* to represent various types of strategy collections (including singletons).

1.5 Related Fields

There exist a number of fields of research that are highly relevant to our work in this dissertation; we review them here and discuss where the body of research in coevolutionary optimization fits in this constellation.

1.5.1 Game Theory

Game theory is the mathematics of strategic interaction [83]. Given a group of interacting entities, we use game theory to predict the outcome of their interaction; if we are one of the interacting entities, then we additionally use game theory to decide how we should behave.

In this dissertation, we take from game theory its emphasis on the notion of *solution concept* (particularly Nash equilibrium [145]) and use it as our organizing theme. Further, we use game theory to obtain several of our results (e.g., Chapters 4 and 5) and inform algorithm design (Chapter 8). Nevertheless, the active research questions in game theory remain quite distinct from those we consider in this dissertation.

We make no attempt to summarize all of the research directions pursued in game theory; rather, we highlight some important distinctions between game theory and coevolutionary optimization.

Despite the primacy and power of Nash's equilibrium concept, it often leaves the game theory practitioner wanting. Games frequently have more than one Nash equilibrium—sometimes a great many; given that we are interested to obtain a Nash equilibrium, the question arises of how one is to rationalize the choice of one equilibrium over another. The choice of equilibrium is especially important in games where the players must coordinate to attain equilibrium.

As Samuelson [180] points out, much of game theory is concerned with the problem of *equilibrium selection*. Research on this problem examines, among other things, various *refinements* of the Nash concept, such as Pareto-optimal and risk-dominant equilibria. Other refinements originate from dynamical-systems considerations, such as local stability, which we discuss below. Finally, conventional game theory makes several strong assumptions, such as agent rationality, complete information, and common knowledge, that simply cannot apply to coevolving agents; hence, the importance of *evolutionary* game theory, discussed below.

1.5.2 Numerical Methods for Computing Nash in Normal-Form Games

If we are interested to locate a Nash equilibrium in a *normal-form* game (i.e., a game expressed as a payoff matrix), then we can apply numerical methods to do so, provided we already have the complete payoff matrix for the game. If the game is zero-sum, then we convert it into a *linear program* (LP) and solve it using the *simplex method* (simple to code, works well in practice, though is known to have a worst-case cost that is exponential in the size of the game) or more recent *interior-point methods* (more difficult to code, but are generally faster in practice and have a complexity polynomial in the size of the game). Readers interested in these methods and LP are referred to Thie [192] and Vanderbei [198].

In the most general case, where we wish to solve a general-sum asymmetric (or *bi-matrix*) game, the conventional method is to convert the game into a *linear complementarity problem* (LCP). Current algorithms to solve LCPs are based upon the Lemke-Howson method [124]; recent treatments of the topic are found in Pang, Stone, and Cottle [158] and Facchinei and Pang [57]. LCP algorithms work well in practice, but are known to be exponential in the size of the game.

The above methods locate only a single Nash equilibrium; in games with many equilibria, these methods do not guarantee convergence to any particular type of Nash. For an overview of methods to find Nash equilibria in games, including methods for locating multiple Nash, see McKelvey and McLennan [136]. In summary, all of the methods discussed here require *a priori* knowledge of the complete payoff matrix; thus, they do not address the problem of locating Nash equilibria when the space of strategies must first be discovered. (Nevertheless, these methods can be incorporated into larger algorithms, as we show in Chapter 8.)

1.5.3 Computing Nash from Extensive-Form and Graphical Models

More recently, attention has turned to the computation of Nash equilibria in games expressed in *extensive form* (i.e., game trees). While any extensive-form game can be converted into normal form, the size of the normal form game is exponential in the size of the game tree; the work of Koller [115] takes advantage of this fact to obtain exponential speed-up over more conventional normal-form methods. Still newer innovations by Kearns, Littman, and Singh [109], Koller and Milch [116], and Ortiz and Kearns [154] use graphical models of n-player games to obtain even more efficient representations and algorithms for solving games. These graphical models gain efficiency over conventional representations when the structure of players' interaction is constrained. While these methods represent the game differently, they encode no less knowledge of the game than the normal form; thus, they require knowledge that we initially lack when using coevolution.

1.5.4 Evolutionary Game Theory

In conventional game theory, three key assumptions buttress the Nash equilibrium solution concept and make it compelling: 1) An agent is rational, 2) an agent has complete knowledge of the game (i.e., the strategies and payoffs for all players), and 3) there exists *common knowledge* amongst the agents of the first two assumptions. The historical criticisms against Nash equilibrium concern the strength of these assumptions, for without their support the substance of the Nash concept appears to collapse. Indeed, we can easily imagine situations where any or all of these three assumptions cannot realistically hold.

In particular, one cannot attribute properties such as rationality to most (if not all) biological agents. In their seminal paper, Maynard-Smith and Price [135] provide an entirely new motivation for the Nash equilibrium concept that enables its application to biology: Our strong assumptions—requirements of agent rationality, knowledge of the game, and common knowledge—are replaced by the principle of Darwinian selection. Their framework is known as *evolutionary game theory* (EGT) [134] and has profoundly changed the tenor of game-theoretic research.

Because of their shared interest in evolutionary principles, we can expect evolutionary game theory to have some kinship to, and relevance for, coevolutionary algorithms; indeed, we will use EGT to obtain some of our key results (see Chapters 4 and 5). Nevertheless, there exist important distinctions between the goals of evolutionary game theory and coevolutionary optimization. The focus of evolutionary game theory remains the problem of equilibrium selection—EGT is not concerned with search or discovery. More particularly, EGT studies equilibrium selection under *replicator dynamics* that nominally model processes in biologic [93] or economic [81] systems. In contrast, coevolutionary algorithm research is very concerned with search and discovery, in addition to equilibrium selection. Also, coevolutionary optimization cares to obtain *particular* outcomes (as specified by solution concepts), and so we build methods to achieve these outcomes rather than model natural processes.

While evolutionary game theory removes the strong assumptions of conventional game theory, it makes several new ones: 1) We assume an infinite population, 2) We assume *complete mixing*, that is, every agent interacts with every other agent, 3) The outcomes of each interaction are expected payoff values, 4) An agent's *fitness* is proportional to its cumulative payoff over all interactions with the members of the current population (this requires cumulative payoffs to be non-negative—see Chapter 4 for details).

As in conventional game-theoretic analyses of normal-form games, evolutionary game theory assumes knowledge of the entire payoff matrix. This knowledge, however, does not reside in any individual agent—individuals are presumed to lack cognizance of their behaviors and the gamut of possible strategies. Knowledge of the strategy space is, instead, a burden placed on the population as a whole: The initial state of the ecology is required to encompass the universe of strategies that we wish to consider; strategies not extant in the initial state will not appear later in evolutionary time.

The EGT framework considers a population of agents that interact with each other. Agents that obtain higher fitness reproduce more offspring than agents who receive lower fitness; further, because agents' fitness values are contingent upon the distribution of agent strategies (known as *frequency-dependent fitness*), the distribution of strategies in the population can change over time—this is the sense in which the population "evolves." The precise nature of the system's dynamics is governed by the game's payoffs and the replicator equations that are used. In-depth discussions of replicator properties are found in Weibull [203], Hofbauer and Sigmund [93], Samuelson [180], Fudenberg and Levine [82], and Gintis [84].

With evolutionary game theory, Maynard-Smith [134] introduces a new solution concept called an *evolutionary stable strategy* (ESS). The ESS is a refinement of Nash equilibrium that adds a criterion of non-invadability. Despite its name, the ESS is actually a static solution concept (as is Nash equilibrium), and has been criticized for this reason (for example, by Nowak [150]). More recently, many other solution concepts have been proposed for the EGT framework (e.g., Lessard [126]), including those that are more properly rooted in dynamical systems theory (e.g., Rowe, et al [176]).

1.5.5 On-Line Learning of Best-Responses

In many learning frameworks, the environment in which learning takes place is distinct from the environment in which the learner is intended to operate. Frequently, the activity of learning stops once the learner is deployed—the period of learning is circumscribed. A very active and diverse area of research concerns *on-line* learning algorithms, which do not distinguish between learning and deployment. With on-line learning, the learner concurrently learns from and operates in the intended deployment environment. Here we discuss on-line algorithms for domains of strategic interaction, where the learner's environment consists of (other) players that are external to the learning system; through interaction with these players, the learner induces *best-response* strategies over a period of time. All of the algorithms we discuss here have the additional property of converging to Nash equilibrium when used in self-play (some under more general circumstances than others). The main appeal of on-line methods is their ability to continually adapt to the behaviors they observe in others.

The earliest example of such algorithms dates from 1951 in the work of Brown [28] and Robinson [171] and is known as *fictitious play*. This algorithm continually updates a history of strategies played by others and plays the pure strategy that is the best response to the current state of this historical distribution. Importantly, when all players use the fictitious play algorithm (i.e., "real" players are replaced with this algorithm, hence "fictitious" play), the historical distributions converge to Nash equilibrium for certain classes of games; one such class is the zero-sum game. Fictitious play does not converge for all games, however. Clearly, fictitious play requires the ability to identify the behaviors of the other players and knowledge of the entire payoff matrix. Further discussions on fictitious play, its variations, as well as a variety of other best-response dynamics, are found in Fudenberg and Levine [82].

A much more recent on-line algorithm, due to Freund and Schapire [79, 80], learns a best response for any two-player game. Their *multiplicative weights* (MW) algorithm makes more modest knowledge demands than fictitious play; given the (potentially mixed) strategy Q used by the other player for a particular round, MW needs to know what payoff each of its available pure strategies would have obtained against Q. Thus, MW is given column-wise "snapshots" of the payoff matrix. These payoffs are used to update weights, one for each pure strategy, that determine the mixed strategy that MW will use in the next round. This algorithm does not maintain an explicit history of prior strategies played by the other player, and therefore cannot explicitly calculate a best-response action. Rather, the less direct feedback of payoffs is used to manipulate the weights. The authors relate the operation of MW to *boosting algorithms*, which are methods that combine the opinions of many poor classifiers through weighting to obtain a very good classifier. The MW algorithm is shown to converge to a best response, which will be a Nash strategy if the other player is optimal, or will be a different strategy that takes advantage of sub-optimal opponent play. In particular, the MW algorithm achieves Nash equilibrium in self-play. The MW algorithm (and a further refinement by Auer, et al [12]) nominally requires knowledge of its available strategies. We may argue that MW learns the payoff structure of the game as it goes, and so it bears some resemblance to the reinforcement learning models we discuss below.

The very recent work of Conitzer and Sandholm [38] is shown to learn a best response in any n-player game. This work assumes complete knowledge of the game's strategies and their payoffs, however. Further, the algorithm begins with an *off-line* computation of equilibrium strategies for each player; when other players' behaviors appear non-stationary, then this equilibrium strategy is used by the algorithm, otherwise the algorithm computes a best-response to the stationary distribution. This algorithm also achieves equilibrium in self-play.

1.5.6 Multi-Agent Learning of the Game and Equilibria

Next, we wish to discuss methods that seek to learn the payoff structure of the game concurrently with their effort to learn an equilibrium or best-response strategy. The methods we include here use reinforcement learning (RL) and are designed for deployment in multi-agent *Markov games*. An agent must learn the reward structure of the game (i.e., the payoffs obtained by different actions in different states) in addition to learning the optimal behavior. Nevertheless, the agent has complete knowledge of its action set. This is in contrast to all of the above methods (with the possible exception of MW), which assume *a priori* knowledge of the game (i.e., the payoffs in addition to the strategies). The RL methods also have the property that they converge to equilibrium when used by all agents. Littman [130] examines two-player zero-sum games; Wang and Sandholm [199] consider n-player coordination games (or team games); Finally, Bowling and Veloso [24] address 2x2 bi-matrix general-sum games. All of these methods include some form of best-response behavior.

As a last example of multi-agent learning using RL, we mention Wolpert and Tumer's Collective Intelligence (COIN) framework [208, 195]. Not unlike some of the graphical models we discuss above, COIN is interested in arbitrarily constrained agent interaction structures, primarily with high localization. These agents interact and seek to maximize their rewards. In addition to agents' local reward functions, there exists a world utility function over the possible histories of agent behavior. COIN considers how one is to engineer agents' local utility functions such that their selfish behavior optimizes not only their private utility functions, but also the world utility. Thus, COIN has some kinship with the field of *mechanism design* (see [133, C. 23] and [146]).

1.5.7 Computational Learning Theory

Rosin and Belew [174] (also Rosin [172]) relate coevolutionary optimization to Valiant's *computational learning theory* (COLT) [196] (see also Kearns and Vazirani [111]). The subject of COLT is efficient inductive concept learning. Broadly speaking, a *concept* is a binary partition of a set $\mathcal{X}$ of elements; all members of one partition are considered instances of the concept, and all members of the other partition are not. The learning task is to induce approximately the structure of a partition (the *target concept*) through exposure to positive and negative examples drawn randomly from the set $\mathcal{X}$ with an arbitrary (though fixed) distribution; the learning algorithm is required to approximate the target concept to within some error ϵ with some probability $1-\delta$, while meeting efficiency constraints.

The analogy that Rosin and Belew draw between coevolution and COLT begins with an asymmetric two-player zero-sum game $\mathbf{G}$. They assume that Player 1 has some perfect pure strategy that beats all Player-2 strategies. This perfect Player-1 strategy embodies the target concept, which we may portray as "the set of Player-2 strategies that can be defeated"; this set contains all Player-2 strategies, as we assert above. An inferior Player-1 strategy embodies a *learner hypothesis* (i.e., an approximation to the target concept) that requires refinement; this process of refinement is achieved through exposure to Player-2 strategies that the inferior Player-1 strategy cannot beat. In terms of the COLT framework, when an imperfect Player-1 strategy fails to beat some Player-2 strategy, then we say that the learner hypothesis has misclassified an instance of the target concept; that is, we have found a Player-2 strategy that the learner hypothesis claims not to belong to the set of defeatable Player-2 strategies.

Rosin and Belew assume an asymmetric situation between the roles of Player 1 and Player 2: There exists a perfect strategy for Player 1 that defeats all Player-2 strategies; for any imperfect Player-1 strategy, there exists some Player-2 strategy that defeats it. Rosin and Belew define a *teaching set* $\mathcal{X}$ to be a set of Player-2 strategies such that any imperfect Player-1 strategy can be defeated by some Player-2 strategy in $\mathcal{X}$. The *specification number* is the size of the minimal teaching set (for the domain under consideration). Rosin and Belew relate the notion of specification number to the work of Goldman and Kearns [177], which defines a *minimal teaching sequence* (MTS). The MTS is the minimal set of examples required to unambiguously identify a particular concept (out of a space of possible concepts).

Rosin and Belew frame the coevolution between Player 1 and Player 2 as a sequence of concept learning problems. The main loop of this process is termed a *competitive algorithm*, which determines what target concepts to use for an embedded *strategy learning algorithm* (SLA); the SLA is assumed to always succeed in learning the given target concept. One particular competitive algorithm that Rosin and Belew discuss is the *covering competitive algorithm* (CCA), which is shown to provide polynomial-time learnability of a perfect Player-1 strategy (with a key assumption that we discuss below). In each iteration of the CCA, the Player-1 strategy learning algorithm is to discover a strategy that can defeat all previously discovered Player-2 strategies; the Player-2 strategy learning algorithm is to discover a teaching set with respect to all previously discovered Player-1 strategies. This process guarantees that the concepts being learned over time incrementally approach the target concept manifested by the perfect Player-1 strategy.

Rosin and Belew assume that the strategy learning algorithms for both players will always succeed and run in polynomial time; they also recognize that this assumption is a strong one. For example, even if the addition of some example clarifies the identity of the target concept (i.e., eliminates certain hypotheses), we may still be unable to efficiently access any improved hypotheses due to the constraints imposed on genetic search by neighborhood structures. Thus, any real-world application of the covering competitive algorithm will certainly handle intransitive structures (which are limited in scope and extent by the assumption of a perfect Player-1 strategy) and eliminate evolutionary forgetting, but it will remain vulnerable to loss of gradient and disengagement (see Chapter 3 for a detailed look at these pathologies). More fundamentally, Kearns and Valiant [110] show that not all concepts can be learned efficiently.

Nevertheless, the relevance of computational learning theory to coevolutionary optimization is clear. Indeed, the use of examples to unambiguously identify a target concept is echoed by our notion of a secondary search effort which seeks interactions needed to identify solutions to our primary search problem. The issue of teaching will be revisited in Chapter 3, where we review methods to create gradient for search, and is the topic of Chapter 7, where we introduce a new method for gradient creation; this approach, called *Pareto coevolution* [64], is essentially a heuristic for picking concept examples that suggest *reachable* improvements to learner hypotheses.

In the conventional coevolutionary algorithm, the inductive processes of the various player roles operate concurrently; this is in contrast to the discrete, synchronized nature of Rosin and Belew's competitive algorithm (though some coevolutionary algorithms do occasionally stop such concurrent operation—see Chapter 3). Thus, if we want to draw a parallel between COLT and more conventional coevolutionary approaches, we must consider the learners' inductive procedure to be the entire "main loop," rather than some subroutine as in the competitive algorithm. In this case, we find that the distribution of test cases to which learners are exposed is generally not stationary, but rather changes as learning proceeds (successfully or not).

1.5.8 Coevolution as Simulator

In Section 1.4.4 we state that a coevolutionary algorithm can be used to perform optimization or to model processes in nature. While this dissertation concerns coevolutionary optimization, we wish to point out some ways in which coevolutionary methods, in their broadest sense, are used to simulate interactive systems and model phenomena. From the field of Artificial Life we have several complex simulators that are used to investigate a number of evolutionary phenomena; chief amongst the goals of such simulators is to investigate the prospect of attaining true *open-ended* (co)evolution with resultant complex structures (we discuss open-endedness more in Chapter 9). Examples of these systems include Holland's *ECHO* [95] (see also Hraber, et al, [97]), Ray's *Tierra* [168, 169], Adami and Brown's *Avida* [3, 1], and Taylor's *Cosmos* [190, 189].

Coevolutionary systems are also used to examine aspects of adaptive behavior. For example, Miller and Todd [138, 139] (also Todd and Miller [194]) use coevolution to investigate the role of sexual selection in evolution. Of particular interest, their simulations illustrate Fisher's [66] process of "runaway sexual selection" in which male traits (and female preferences for those traits) become highly exaggerated. Indeed, some traits are accentuated to the extent that they become liabilities with respect to other aspects of the male's environment, such as predators. If we equate fitness with reproductive success, as opposed to simple survival, then the requirements of female partners are easily understood as important. That female requirements can exact some tradeoff with respect to other components of fitness may appear unintuitive; nevertheless, as Miller and Todd discuss, a male's survival *in spite of* an otherwise deleterious trait serves as a reliable signal of his viability. Indeed, Miller and Todd also discuss how sexual selection may help in the process of optimization; here, they point out that sexual selection can be an effective mechanism for the creation of diversity, since female preferences are not anchored to static aspects of a broader learning environment and are free to vary. In connection with diversity creation, earlier work of Todd and Miller [193] shows how sexual selection can enable *sympatric speciation*, that is a process of speciation that does not rely on geographic separation to implement reproductive isolation. Other simulations of sympatric speciation are found in Kondrashov and Shpak [117] and Dieckmann and Doebeli [52], for example.

The evolution of communication is also frequently studied in the framework of co-adaptation. An early example is that of Werner and Dyer [205] in which "female" and "male" agents learn to coordinate behaviors to improve reproductive success. The agents are spatially distributed on a lattice; the females are stationary and sighted while the males are mobile and blind. When a male encounters a female, they procreate offspring which replace randomly selected agents in the lattice. The females generate signals that the males receive; in the most adaptive pairs, these signals serve to direct the movement of the male such that it more quickly finds the female. Since coordination leads to more efficient reproduction of offspring, a communicative convention naturally emerges from the system. Other investigations on the evolution of communication include Saunders and Pollack [181], Oliphant and Batali [151], and Di Paolo [50, 51]. Our own early work concerns the use of coevolution and communication [60] and is described further in Chapter 3.

Coevolutionary simulations are also used to study economic and social systems. For example, LeBaron [121, 122] and LeBaron, Arthur, and Palmer [123] investigate properties of financial markets with agent-based simulations. Arthur [10, 11] and Axtell [15, 16] consider tasks of social coordination where agents have bounded rationality.

The examples of coevolution that we discuss above present a variety of applications in which the objective is to develop a better understanding of some system or process, rather than obtain an optimal solution. This is not to say that the notion of optimization is entirely foreign to these studies, however. On the contrary, several of our results (primarily from Chapters 4 through 6) are relevant to coevolutionary models where polymorphic Nash equilibria may be natural outcomes of the system.

1.6 Contributions and Dissertation Structure

This dissertation explores the pathologies characteristic of coevolution and frames them in a novel way. We illustrate that these pathologies can be understood to result when solution concepts are improperly implemented or poorly chosen. We review the coevolutionary algorithm literature from this perspective and use solution concepts to guide the development of novel methods to address these pathologies. The following overviews the dissertation and its contributions chapter by chapter.

Chapter 2: Framework This chapter develops a formal framework to show how single-player games against Nature can be reformulated to become multi-player games; it further shows how multi-player games can be reformulated to become multi-objective optimization problems. The utility of this framework is that it invites us to think more sharply about the decisions we make when we build a coevolutionary system. As important, this framework will be used in Chapter 9 to present our results on *monotonicity*.

Chapter 3: Taxonomy and Review This chapter begins by asking When is a coevolutionary algorithm a useful tool? We review the successes of coevolution and then taxonomize the pathologies that are commonly encountered; the taxonomy operates on the novel view that these pathologies result when solution concepts are improperly implemented or poorly chosen. Finally, we review relevant coevolutionary algorithm literature that addresses various pathologies and consider it in the light of our taxonomy and our emphasis on solution concepts.

Chapter 4: A Game-Theoretic Analysis of Selection Methods Here we present our first central result, which shows that several commonly used selection methods are inappropriate for use in coevolutionary algorithms when the domain in question is a variable-sum game. Specifically, the examined selection methods do not properly implement the Nash equilibrium solution concept and prevent the algorithm from converging onto polymorphic Nash equilibria; instead, these selection methods induce cycling, distorted point attractors, or chaos. A simple evolutionary game-theoretic framework is used to establish these results. This work is first published in Ficici, Melnik, and Pollack [58].

Chapter 5: A Game-Theoretic Analysis of Fitness Sharing Methods Using the methods of Chapter 4, we turn our attention to fitness-sharing methods. We show that the fitness sharing methods we investigate distort polymorphic Nash equilibria. The results of Chapters 4 and 5 challenge the tacit assumption that an evolving population can simultaneously perform search and properly represent the solution of coevolutionary search. The work in this chapter is also found in Ficici and Pollack [63].

Chapter 6: Effects of Finite Populations A strong assumption made in evolutionary game theory [134] is that the evolving population is infinitely large. Recent simulations by Fogel, et al, [73, 74, 75] show that finite populations produce behavior that, at best, deviates with statistical significance from the *evolutionary stable strategy* (ESS) predicted by EGT. They conclude that evolutionary game theory loses its predictive power with finite populations.

In this chapter, we revisit the question of how finite populations affect EGT dynamics. Some of the experiments of Fogel, et al, use truncation selection, which we show in Chapter 4 to distort polymorphic equilibria even with infinite populations. Other of their experiments use fitness-proportional selection, implemented with the standard "roulette wheel" mechanism. By paying close attention to how the multinomial distribution produced by roulette wheel interacts with the dynamical map of an infinite-population system, we are able to account for the divergence between ESS predictions (based on infinite populations) and results observed in our own finite-population simulations. We build a predictive Markov-chain model of a finite-population system; we then show that Baker's SUS [20] selection method largely corrects the observed divergence. This work is first published in Ficici and Pollack [61].

Chapter 7: Pareto Coevolution In this chapter, we present our first core algorithmic innovation. The algorithm presented here is based upon the observation that the challenge faced by a coevolving individual can be described as a multi-objective optimization problem. Specifically, each individual with whom an agent interacts can be considered a dimension of optimization for that agent. This suggests that notions used in multi-objective optimization (MOO), in particular the solution concept of Pareto optimality, can be applied to coevolution. Pareto optimality takes us away from the conventional notion that the solution we seek is an *individual*; instead, we are now open to the possibility that the solution to a coevolution problem may be a *set* of individuals, known as the *Pareto front*.

In particular, the Pareto solution concept provides a principled way of handling *tradeoff surfaces*, where improvement with respect to one objective entails degradation with respect to another—a fresh perspective on coevolution and the problem of intransitivity. In addition to recognizing the relevance of Pareto optimality to coevolution, this chapter introduces the orthogonal concept of *distinctions*, which is used to address directly the pathology of disengagement. In so doing, we seek a matched pair of solution concepts for the primary and secondary search problems we discuss above. We apply our methodology to the density classification task for cellular automata and obtain a rule with an 84% classification rate; this rule is better than all previously published results except for those of Juillé and Pollack [104, 105, 106]. The work in this chapter is first published in Ficici and Pollack [64].

Chapter 8: A Game-Theoretic Memory Mechanism Another pathology that can arise if the wrong solution concept is used for the secondary search problem is that of *forgetting*, where previously learnt traits are lost, only to be needed later, and so must be re-learnt. As our second algorithmic innovation, we introduce a new memory mechanism based upon Nash equilibrium, demonstrate its ability to accumulate desirable traits, and contrast its behavior and performance against other memory methods found in the literature. This work is first published in Ficici and Pollack [65].

Chapter 9: Solution Concepts and Monotonicity Our final analytic results show that certain solution concepts have a special property we call *monotonicity*, which determines whether a search algorithm can return answers of non-decreasing quality when queried over time during its operation. We show that Nash equilibrium is a monotonic solution concept; Pareto optimality is monotonic only if implemented a certain way. The solution concept implemented in the conventional coevolutionary algorithm uses frequency-dependent fitness and selects the single individual that obtains the highest average score from interaction with others; both this solution concept and a modified version that uses frequency-independent fitness are *not* monotonic.

More importantly and contrary to common belief, monotonicity implies that objective (i.e., global) metrics of goodness can be applied to coevolutionary domains. The non-monotonicity of conventional coevolutionary algorithms may help explain the common sentiments that 1) objectivity is intrinsically foreign to coevolutionary systems and 2) that coevolutionary algorithms do not optimize. Further, our monotonicity results obtain in "open-ended" domains—where there exists an infinity of behavioral strategies—even when the properties of strategies cannot be anticipated in advance of their actual discovery.

Chapter 2

Foundational Concepts and Framework

2.1 Introduction

The purpose of this chapter is manifold. First, we wish to build a framework within which we can precisely articulate different kinds of search problems; we want to express games against nature and multi-player games, single-objective optimization and multi-objective optimization, static domains and dynamic domains. Second, we want the framework to be expressive enough to allow us to frame a search problem in many ways; we will show, for example, how some games against nature can be re-cast as multi-player games and how multi-player games can be re-cast as multi-objective optimization problems. Most importantly, we present this framework to help discipline our thinking about coevolutionary algorithms, and expose the various decisions that we make when we construct and use coevolutionary algorithms. Finally, we will use portions of this framework in Chapter 9 to discuss the concept of *monotonicity*. To help the reader, Table 2.1 provides a quick reference to the notation used.

Category	Notation	Meaning
Domains	$\mathfrak{D}$	Problem domain
	$\mathcal{R}$	Set of domain roles; there are two role types
	P	*Player* role-type
	$\mathcal{N}$	*Nature* role-type
Behaviors	$\mathcal{B}_i$	Set of behaviors available to role i
	$\mathcal{E} = \langle b_1, \ldots, b_n \rangle$	Behavioral event; the interaction of an n-tuple of behaviors
	v	Metric or dimension of behavior; also known as a *view* of behavior
	$\mathcal{V}_i$	Set of metrics used to assess behavior of player-role i
	$M(i, \mathcal{E}, v) \rightarrow \mathbb{R}$	Measurement function; returns goodness of player-role i's behavior in event $\mathcal{E}$ according to given metric v
	$\mathbf{T}$	Table of measurements
Configurations	$\mathcal{X}$	Strategy complex
	$\mathcal{K}$	Configuration of complexes
Solutions	$\mathcal{K}^*$	Solution
	S^*	Set of solutions
	$\mathcal{O}$	Solution Concept
	$\mathrm{Pref}(\mathcal{K}_\alpha, \mathcal{K}_\beta)$	Preference relation between two configurations
Search	$\mathcal{P}$	Set of phenotypic behaviors
	$\mathcal{W}$	State of knowledge

Table 2.1: Guide to notation used in our framework.

2.2 Domains and Roles

A *problem domain* $\mathfrak{D}$ specifies a universe of relevant behaviors we wish to study; this universe may be finite or infinite, countable or uncountable. Further, every problem domain $\mathfrak{D}$ defines at least two *roles* that interact; at least one of these roles is a *player* and no more than one is *nature*. The two role *types*—player and nature—are denoted P and $\mathcal{N}$, respectively. Thus, the set of roles $\mathcal{R}$ defined by a domain must follow one of three templates: $\mathcal{R} = \langle P, \mathcal{N} \rangle \mid \langle P_1, \ldots, P_n \rangle \mid \langle P_1, \ldots, P_n, \mathcal{N} \rangle$. The universe of domain behaviors is divided into sets, such that domain role i has an associated set of available behaviors $\mathcal{B}_i$. A search problem, then, concerns finding optimal behaviors for those roles of a domain that are players. The vast majority of search problems can be described as either *games against nature* or *multi-player games*.

2.2.1 Games Against Nature

In the Travelling Salesperson Problem (TSP), we seek a sequence of visits to different cities that minimizes the over-all round-trip distance travelled. (Formally, we treat each city as a vertex of an undirected, complete graph; a legal tour of the cities is a graph cycle that includes all of the cities.) Given n different cities, the salesperson has $n!$ possible sequences from which to choose. The TSP domain thus has two roles: One player (the salesperson), who has $n!$ available behaviors, and nature, which determines the geographic distribution of cities the salesperson must visit. Another example is Rubik's cube, where we seek a sequence of moves that takes the cube from some initial configuration to a configuration where each of the cube's six sides are of a single color. Again, we have two roles: One player (the manipulator of the cube) and nature (which determines the properties of the cube and its initial configuration). As a final example, consider the domain where we seek a real value x to maximize the function $f : \mathbb{R} \to \mathbb{R}$. The player role chooses the value x and nature determines the properties of the function f. All of these domains are examples of *games against nature*.

2.2.2 Multi-Player Games

Tic-Tac-Toe is a domain that specifies two player roles, 'X' and 'O', and each role has a finite set of available (deterministic) behaviors. The game of chess is similar, in this sense. The game of backgammon can be said to include three roles: Two players (black and white) and nature (as manifested by the dice). All of these domains are examples of *multi-player games*. The domains we examine in this dissertation—interactive domains—are also multi-player games. Indeed, we will see that some games against nature can also be cast as multi-player games; examples include the *majority function* in cellular automata research [142, 119, 104, 152] and *sorting networks* [92, 100].

2.2.3 Domain Symmetry

A domain exhibits *symmetry* if any two or more player roles are indistinguishable, otherwise it is *asymmetric*. That is, in a symmetric domain, there exist at least two player roles i and j for whom the the sets of available behaviors are identical ($\exists i, j : \mathcal{B}_i = \mathcal{B}j$); further, the measured outcomes obtained by a behavior $b \in \mathcal{B}_{i,j}$ is independent of which player role (i or j) is using that behavior, though it is still generally dependent upon the behaviors of the other players $k \neq i, j$ and nature. An asymmetric domain, in contrast, gives no two player roles the same set of behaviors, or, if it does, then the measured outcomes obtained by a "common" behavior b depends upon which player role uses it. We can reasonably argue, therefore, that in this latter case the behaviors sets are not truly identical.

2.3 Contexts for Behavior: Events

Behaviors obtain meaning only when they operate within some context. For example, in the TSP domain we have $n!$ distinct behaviors that the salesperson may exhibit, yet none of these behaviors are meaningful unless we apply them to a specific context—in this case, a particular geographic distribution of the n cities. As another example, a chess strategy cannot unfold without the interaction of an opposing strategy.

The contexts within which we consider behaviors are *events*. An event occurs when each domain role is instantiated by a particular behavior and these behaviors interact. The event $\mathcal{E}$ is the n-tuple $\mathcal{E} = \langle b_1, \ldots, b_n \rangle$, where $b_i \in \mathcal{B}_i$ is the behavior exhibited by role i of the domain.

2.4 Dimensions of Behavior

A behavior $b \in \mathcal{B}_i$ participating in an event $\mathcal{E}$ can be assessed according to any number of objective metrics, or *dimensions*, of behavior. Each "view" on behavior $v \in \mathcal{V}_i$ can be applied to any behavior used by player role i. We do not apply metrics to the behavior of nature.

Typically, a problem domain comes with associated metrics of goodness to be applied to behaviors; such is the case for any conventional board game, for example, where the game's rules not only circumscribe the space of allowable behaviors, but also provide criteria to judge the game's outcome. Nevertheless, as we will discuss below, not all domains have clearly associated metrics of goodness. Therefore, in this formalism, we do not insist upon a tight linkage between a problem domain $\mathfrak{D}$ and the sets of applied objective metrics $\mathcal{V}_{1\ldots n}$.

2.4.1 Single and Multi-Objective Optimization

According to the cardinality of $\mathcal{V}_i$, we categorize the problem for Player i as either *single-objective optimization* or *multi-objective optimization* (MOO). For example, the TSP domain has a single metric that measures round-trip distance; Rubik's cube also has a single metric that indicates whether the cube is in the target state or not (or, the metric may instead indicate a "distance"—e.g., in move space—from the target state). Maximization of the function $f : \mathbb{R} \to \mathbb{R}$ requires a single metric that measures the magnitude of f given some argument (behavior) x.

In contrast, we may have a domain where we seek a real value x that maximizes a function $g : \mathbb{R} \to \mathbb{R}$ and simultaneously minimizes another function $h : \mathbb{R} \to \mathbb{R}$. Behaviors (i.e., values of x) in this domain have two distinct dimensions to be measured, one with respect to g and the other with respect to h. Another multi-objective domain may concern searching the design space for a bridge, where we wish to minimize mass yet maximize strength. This latter example shows that one objective can conflict with another, which implies the existence of a *trade-off surface*; further, different objectives may be measured with fundamentally different units.

2.4.2 Selection of Metrics

The relevant metrics of behavior are self-evident for many domains. Yet, there exist domains for which one cannot easily surmise what the correct set of metrics is. For example, a domain like robot locomotion presents a challenge to the measurement of behavior because one can imagine many different beliefs about what precisely constitutes effective locomotion. Indeed, one may be entirely unable to formulate a precise description and rely instead on heuristic metrics. The seminal work of Sims [186], for example, uses a heuristic approach that excludes those robot behaviors that locomote by falling over.

Thus, many problem domains we care about do not have such precise and transparent ways to characterize behavior. When we apply heuristic metrics, the *intended* problem domain and the *measured* problem domain diverge. The extent of divergence is generally unknowable. (Thus, we are not infrequently surprised to find an obviously poor candidate that the objective function opines "good.")

To continue with our example robot-locomotion domain, we wish to discover an effective combination of robot morphology and control. We may choose to measure the average rate of movement, distance travelled from the starting point, and the surface area of the robot that comes in contact with the ground; we prefer robots that travel fast, but not in circles, and do not drag themselves on the ground (to encourage the discovery of limbs, perhaps). Thus, we have three different units of measurement in this domain: Rate, distance, and area. While we have three metrics of behavior, they are all applied to the same observed event: The robot's interaction with its environment (i.e., player and nature).

2.5 Measurement of Events

We are interested to learn the success with which various behaviors operate in the domain. We define the function $M(i, \mathcal{E}, v) \to \mathbb{R}$ to measure the success of the behavior used by role i in event $\mathcal{E}$ according to a particular objective metric $v \in \mathcal{V}_i$. The scalar returned by M is a *measurement of success*; higher values indicate greater success according to the applied metric (we do not assume any particular type of scale—indeed, a *nominal scale* may conceivably be used). Given a problem domain $\mathfrak{D}$, the function M is defined over all possible events (i.e., all n-tuples of behavior choices by the n domain roles), over all defined metrics $v \in \mathcal{V}_i$, and for all values of i that correspond to player roles (i.e., we do not measure the "success" of nature's behavior.) Figure 2.1 illustrates the measurement of a behavioral event in our robot locomotion domain.

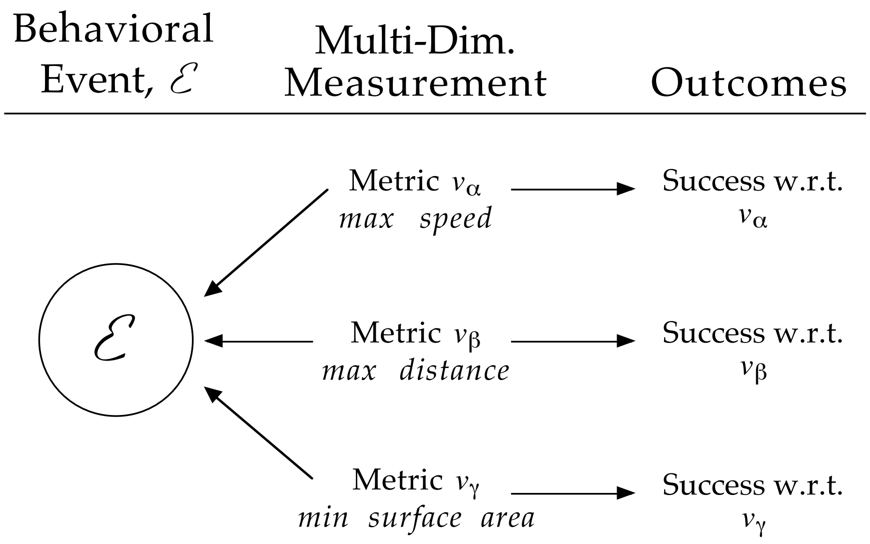


Figure 2.1: Multiple metrics applied to the same behavioral event $\mathcal{E}$.

2.5.1 Fidelity of Measurement

Once we know what metrics of behavior we wish to apply, we may face the additional issue of measurement fidelity. For example, our measurement device will have finite precision; if we cannot measure with sufficient precision, then we will not be able to distinguish between all domain behaviors. In effect, poor measurement reduces the gamut of our domain. Another challenge to the accurate measurement of behavior arises when a domain role (a player or nature) can exhibit stochastic behaviors. For example, in backgammon, our measurement of a player's behavior (against some opponent) will be poor unless the period of observation spans a great many games.

2.5.2 Measurement Tables and Sub-Tables

Our measurement function M is defined over all events $\mathcal{E} \in \mathcal{B}_1 \times \ldots \times \mathcal{B}_n$, over all metric sets $\mathcal{V}_i$, and for all player roles. We can imagine placing all the measurements returned by M into a table, which we denote $\mathbf{T}$. We will say that one table $\mathbf{T}^\alpha$ is a *sub-table* of another table $\mathbf{T}^\beta$, denoted $\mathbf{T}^\alpha \subseteq \mathbf{T}^\beta$, if and only if $\forall i : \mathcal{B}_i^\alpha \subseteq \mathcal{B}_i^\beta$ and $\mathcal{V}_i^\alpha \subseteq \mathcal{V}_i^\beta$. Figure 2.2 illustrates two examples of sub-tables for a two-player domain. Our measurement table is similar to a normal-form game matrix in that it records the outcomes of interactions between different behaviors; nevertheless, $\mathbf{T}$ is distinct from a game matrix because the measurements it contains are not payoffs, as we discuss below.

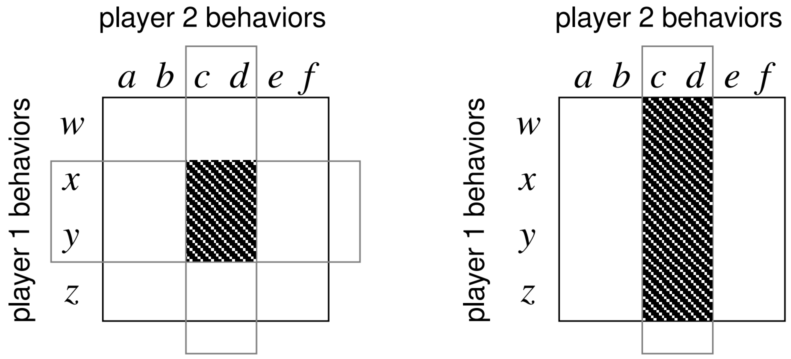


Figure 2.2: Two examples of sub-tables. Each sub-table, indicated by the shaded region, contains a subset of the available behaviors; the sets of metrics (not shown) remain the same in this example.

2.5.3 Measurements and Utilities

The measurements we obtain with different metrics of behavior, while all scalars, result from conceptually distinct perspectives of an event $\mathcal{E}$; therefore, measurements across different metrics are not comparable (even if they use the same unit of measurement) and must not be interpreted as *payoffs* or *utilities* as conventionally understood in game theory. In our framework, to interpret measurements as utilities is essentially to impose a common currency of goodness over all metrics of behavior. To return to our robot-locomotion domain, a robot's behavior yields some triple of payoffs that represents the goodness of the behavior with respect to speed, distance, and surface area. By treating these payoffs as utilities, we are free to exchange units of goodness in one dimension of behavior for units of goodness in another; a robot that receives the triple $\langle 25, 25, 25 \rangle$ is equally good as another robot that receives $\langle 75, 0, 0 \rangle$. While we may be satisfied with such a tradeoff, we do not wish to impose a common currency of goodness in our measurement function M; rather, we wish to maintain the independence of the different dimensions of behavior, such that each measurement reflects the goodness of a behavior with respect to a single dimension of behavior. To provide a way to combine these dimensions into a holistic assessment of behavioral goodness, we use a higher-level construct called a *solution concept*.

2.6 Solution Concepts

As we discuss above, each measurement $M(i, \mathcal{E}, v)$ represents the success of a particular behavior, acting in a particular context, with respect to a single metric (or dimension) of behavior. Here we introduce the *solution concept*, which provides a way to integrate across multiple events and dimensions of success to obtain a single holistic assessment of quality, or behavioral success. The precise way in which the integration occurs depends upon the solution concept being applied. Thus, the solution concept does not operate directly upon the problem domain, but rather upon the intermediary measurement function M, which additionally specifies metrics of behavior. Therefore, we will say that solution concepts "solve" measurement functions, rather than domains.

2.6.1 Behavior Complexes

A domain role's contribution to a solution need not be a single behavior, but may instead be a collection of some sort. To allow for this possibility, we introduce the *behavior complex.* A *behavior complex* $\mathcal{X}_i$ is a subset of the corresponding set of behaviors $\mathcal{B}_i$ that are made available to domain role i. Additionally, a behavior complex may possess other attributes, according to the solution concept $\mathcal{O}$ under consideration. For example, if our solution concept is Nash equilibrium, then a behavior complex specifies a mixed strategy. The complex $\mathcal{X}$ corresponding to the mixed strategy m contains the set of pure strategies played by m with non-zero probability, that is, $\mathcal{X} = \{b \in \mathcal{B} : \Pr(b|m) > 0\}$; as an additional attribute, the complex must specify the probability distribution used by the mixture. Alternatively, if our solution concept is Pareto optimality, then a behavior complex specifies the non-dominated front. We will discuss these possibilities in detail over the course of this dissertation; our point here is merely to illustrate that the behavior complex may take different forms.

2.6.2 Configurations and Solutions

A *configuration* $\mathcal{K}$ is an n-tuple of behavior complexes, one for each role of the domain. (Note that a configuration is not the same thing as an event. An event is an n-tuple of behaviors; a configuration is an n-tuple of behavior complexes. See Section 2.3.) A *solution* (with respect to the measurement table $\mathbf{T}$) is a configuration $\mathcal{K}^*$ that exhibits some set of properties defined by the solution concept $\mathcal{O}$. A *solution set* is the set of all possible solutions with respect to the given measurement table $\mathbf{T}$ and solution concept $\mathcal{O}$, which we denote $\mathrm{S}^*(\mathbf{T}, \mathcal{O})$; the solution set may be empty.

2.6.3 Defining Solution Concepts

A *solution concept* (a.k.a. optimality concept) $\mathcal{O}$ is defined either *extensionally* or *intensionally*. A solution concept can simply specify which configurations belong to the solution set and which do not, without stating any underlying properties by virtue of which a configuration is considered a solution; a solution concept so defined is extensional. Alternatively, a solution concept can state a number of properties that a configuration must possess to be a member of the solution set; such a solution concept is intensional.

2.6.4 Preference Relation

Given the solution set, we can define the following preference relation (or predicate). We *strongly prefer* a solution $\mathcal{K}^*$ to a non-solution $\mathcal{K}$; that is, $\mathrm{Pref}(\mathcal{K}^*, \mathcal{K}) = 1$ and $\mathrm{Pref}(\mathcal{K}, \mathcal{K}^*) = 0$. Further, we *weakly prefer* $\mathcal{K}_\alpha$ to $\mathcal{K}_\beta$ if and only if for every measurement sub-table $\mathbf{T}_\beta$ in which $\mathcal{K}_\beta$ is a solution there exists a sub-table $\mathbf{T}_\alpha$ in which $\mathcal{K}_\alpha$ is a solution and where $\mathbf{T}_\alpha \supset \mathbf{T}_\beta$. Finally, if we neither strongly nor weakly prefer one configuration over another, then we have no preference at all between the two. Particularly, given two solutions, we do not prefer one over the other; that is, $\mathrm{Pref}(\mathcal{K}^*_\alpha, \mathcal{K}^*_\beta) = 0$ and $\mathrm{Pref}(\mathcal{K}^*_\beta, \mathcal{K}^*_\alpha) = 0$.

Definition 1 (Preference Relation) $\mathrm{Pref}(\mathcal{K}_\alpha, \mathcal{K}_\beta) = 1$ *iff for all* $\mathbf{T}_\beta$*, where* $\mathbf{T}_\beta \subseteq \mathbf{T}$ *and* $\mathcal{K}_\beta \in \mathrm{S}^*(\mathbf{T}_\beta, \mathcal{O})$*, there exists* $\mathbf{T}_\alpha$*, where* $\mathbf{T}_\alpha \subseteq \mathbf{T}$ *and* $\mathcal{K}_\alpha \in \mathrm{S}^*(\mathbf{T}_\alpha, \mathcal{O})$*, such that* $\mathbf{T}_\alpha \supset \mathbf{T}_\beta$.

2.6.5 Solution Concept Refinements

When our solution set contains more than one solution, we may have an informal preference for one solution over another. To formally incorporate our preference into the solution concept, we need a solution concept *refinement*. For example, let us consider the central solution concept of game theory, Nash equilibrium. Many games have multiple Nash equilibria, and much research effort has been invested to rationalize the choice of one Nash over another [180]; some Nash equilibria have additional properties that others do not, and so may be preferred. Examples of these properties include Pareto dominance and risk dominance [83], and choices based upon these properties are examples of solution concept refinements.

We define the concept of refinement for our framework as follows.

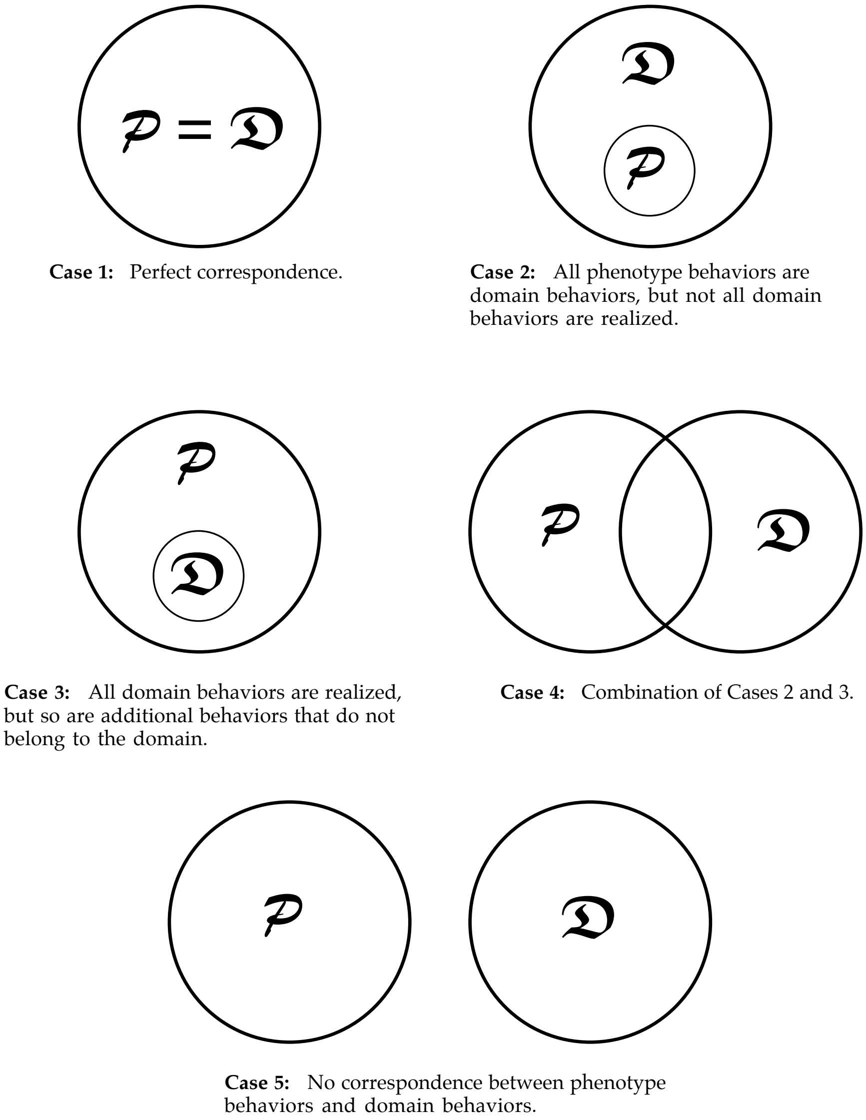
Definition 2 (Refinement) *For any two solution concepts $\mathcal{O}_1$ and $\mathcal{O}_2$, concept $\mathcal{O}_1$ is a* refinement *of concept $\mathcal{O}_2$ iff* $\forall \mathbf{T} : \mathrm{S}^*(\mathbf{T}, \mathcal{O}_1) \subseteq \mathrm{S}^*(\mathbf{T}, \mathcal{O}_2)$ *and* $\exists \mathbf{T} : \mathrm{S}^*(\mathbf{T}, \mathcal{O}_1) \subset \mathrm{S}^*(\mathbf{T}, \mathcal{O}_2)$.

2.7 Genotypes and Phenotypes

A *substrate* is an arbitrary representation that one may subject to variation for search. Each instantiation of a substrate is interpreted in some way to yield a behavior. Examples of substrates include grammars (e.g., [127]), neural network architectures (e.g., [7]), genetic programs (e.g., [39, 118]), and bit-strings (e.g., [85]). In evolutionary algorithm parlance, a substrate is a *genome* and its instantiation is a *genotype*; the interpretation of a genotype results in a *phenotype*, which in-turn yields a behavior. Note that the genotype-to-phenotype mapping is often many-to-one; that is, two or more distinct genotypes may result in phenotypic behavior that is indistinguishable, at least as far as our measurements are concerned. (If the translation from genotype to phenotype is stochastic, then many-to-many mappings are possible, as well.)

The universe of phenotypic behaviors $\mathcal{P}$ enabled by a substrate (via its interpretation) may overlap in any number of ways with the universe of behaviors specified by a domain $\mathfrak{D}$. We can imagine five general cases, given a single substrate, as illustrated in Figure 2.3. First, the two sets $\mathcal{P}$ and $\mathfrak{D}$ may be identical; thus, every behavior possible in the domain is generated by the substrate and no behavior generated by the substrate is external to the domain. Second, the set $\mathcal{P}$ may be a proper subset of $\mathfrak{D}$. In this case, all behaviors generated by the substrate belong to the domain, but not all domain behaviors are covered; in particular, the domain behaviors we care about most—those that belong to solutions—may be excluded. Third, the set $\mathcal{P}$ may be a proper superset of $\mathfrak{D}$; in this case, all the behaviors of the domain are covered, but the substrate also generates behaviors that fall outside of the domain. (An example of this situation can be found in [9] where genetic programs are evolved to play Tic-Tac-Toe; in addition to generating all the possible (deterministic) strategies for the domain, their substrate also generates behaviors that make illegal moves. Such behaviors are not disallowed, but rather cause the player making an illegal move to forfeit a turn.) Fourth, neither set may contain the other, but a non-empty intersection may exist; this case combines the features of Cases 2 and 3. Fifth, the two sets may have an empty intersection; such a situation is akin to expecting locomoting robots to play Tic-Tac-Toe strategies.

More generally, we may use a different substrate for each role defined by the domain. Examples of this include the *majority function* in cellular automata research [142, 119, 104, 152] and *sorting networks* [92, 99]. In the majority function, rules are bit-strings of length 128 and initial conditions are bit-strings of length 149; sorting networks are represented as sequences of comparison-exchange operations, while inputs are bit-strings. Thus, the overlap of substrate-generated and domain behaviors may change from role to role.



Case 1: Perfect correspondence.

Case 2: All phenotype behaviors are domain behaviors, but not all domain behaviors are realized.

Case 3: All domain behaviors are realized, but so are additional behaviors that do not belong to the domain.

Case 4: Combination of Cases 2 and 3.

Case 5: No correspondence between phenotype behaviors and domain behaviors.

Figure 2.3: Five cases of overlap between phenotypic and domain behaviors.

2.8 Distributions over Behaviors

Let us denote the behaviors made available to role i by the domain and substrate $\mathcal{B}_i^{\mathfrak{D}}$ and $\mathcal{B}_i^{\mathcal{P}}$, respectively. Depending upon the nature of the substrate, we may encounter a phenotypic behavior $b \in \mathcal{B}_i^{\mathcal{P}}$ that does not map to any single domain behavior but rather corresponds to a probability distribution over some subset of domain behaviors; such a phenotypic behavior is akin to a *mixed strategy* in game theory.

To illustrate this point, let us consider the children's game Rock-Paper-Scissors. The conventional presentation of this domain defines three *pure strategies*: Rock, Paper, and Scissors. Our substrate may code for this domain with an integer ranging from one to three. In this case, each of the domain's pure strategies corresponds to exactly one pure strategy (phenotypic behavior) generated by the substrate. Alternatively, our substrate may use a triple of floating-point numbers, $\langle r, p, s\rangle$, where $0 \leq r, p, s \leq 1.0$ and $r + p + s = 1.0$. This substrate allows us to code mixed strategies for Rock-Paper-Scissors. In this case, exactly three phenotypic behaviors map exactly to individual domain behaviors: $\langle 1.0, 0, 0\rangle \rightarrow$ Rock, $\langle 0, 1.0, 0\rangle \rightarrow$ Paper, and $\langle 0, 0, 1.0\rangle \rightarrow$ Scissors. Every other phenotypic behavior $b \in \mathcal{B}^{\mathcal{P}}$ lacks a corresponding behavior in $\mathcal{B}^{\mathfrak{D}}$.

Thus, the universe of substrate-generated behaviors is properly viewed as distinct from the universe of domain-defined behaviors. In game-theoretic parlance, we view each phenotypic behavior $b \in \mathcal{B}^{\mathcal{P}}$ as a pure strategy, regardless of how it may correspond to the domain; mixed strategies (i.e., distributions over phenotypes) are introduced by behavior complexes and appropriate solution concepts (See Section 2.6, above). Alternatively, in cases where $\mathcal{P}$ and $\mathfrak{D}$ are not identical, we may choose to view $\mathcal{P}$ as perfectly representing the *de facto* domain, since we are *measuring* $\mathcal{P}$ and not $\mathfrak{D}$.

2.9 From Domain to Measurement

With the introduction of the substrate, we can now trace the transformation of a domain into a table of measurements, as Figure 2.4 illustrates. The problem domain $\mathfrak{D}$ specifies a universe of behaviors that we are interested to explore. The substrate (along with an interpretive process) provides a way to express behaviors (phenotypes $\mathcal{P}$) and thereby realize our exploration of the domain—the substrate provides the behaviors we can actually observe. The sets of metrics $\mathcal{V}_i$ that we apply to a behavioral event yield our measurements of success. These measurements are then integrated by the solution concept $\mathcal{O}$ to indicate which configurations $\mathcal{K}$ of behavior complexes constitute solutions to our search problem. We are interested to investigate how solutions obtained through these levels of indirection compare to the idealized solutions that exist in the domain. Further, we are particularly interested to investigate the complications that arise when we implement solution concepts in search algorithms.

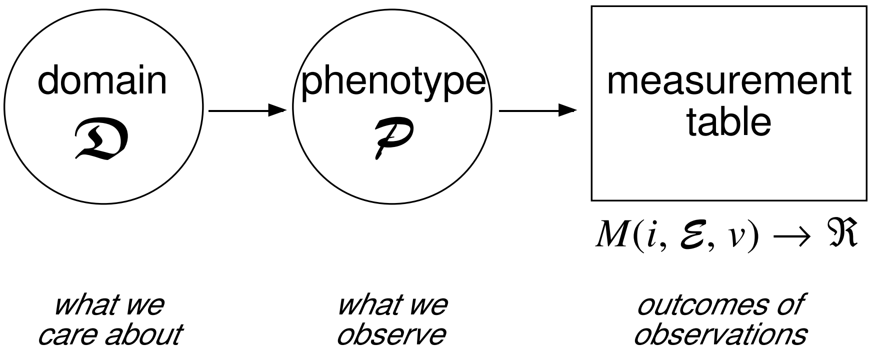


Figure 2.4: Transformation of domain of interest into a measurement table via substrate.

2.10 Examples: Measurement Tables

Now that we have reviewed our formalism, we will illustrate how a problem domain can be viewed in a number of ways. Using the density classification task for cellular automata as our example, we show how a number of different measurement tables can be obtained. The objective of the density classification task (also known as the *majority function*) is to find an automaton rule that will cause the automaton to converge to a state of all-ones if the density (i.e., proportion of one-bits) of the initial condition is beyond some pre-specified threshold; otherwise, the automaton rule is to cause the automaton to converge to a state of all-zeros. The typical statement of this problem specifies the threshold to be 0.5. Further, the automaton rule typically has a radius of three, which gives a space of 2^{128} rules; the automaton typically has a lattice of 149 bits, which gives a space of 2^{149} possible initial conditions (ICs) to classify. Finally, the rule is given 300-320 time-steps to operate. Chapter 7 discusses the majority function in detail.

2.10.1 Single-Player Game Against Nature

A conventional, albeit impractical, approach to the density classification task is to treat it as a game against nature. Our domain $\mathfrak{D}$ has one player role, the behaviors of which are the 2^{128} automaton rules; the role of nature represents a monolithic evaluator that tests a rule on each of the 2^{149} possible initial conditions. An event consists of a rule interacting with nature: $\mathcal{E} = \langle rule_i, \mathcal{N} \rangle$. We apply a single metric of goodness v that returns the percentage of initial conditions that the tested rule correctly classifies. Thus, our measurement table appears as illustrated in Figure 2.5.

2.10.2 Two-Player Constant-Sum Game

We can view the majority function as a constant-sum game for two players; Player 1 represents automaton rules, while Player 2 represents automaton initial conditions. An event consists of a rule interacting with an initial condition: $\mathcal{E} = \langle rule_i, ic_j \rangle$. The

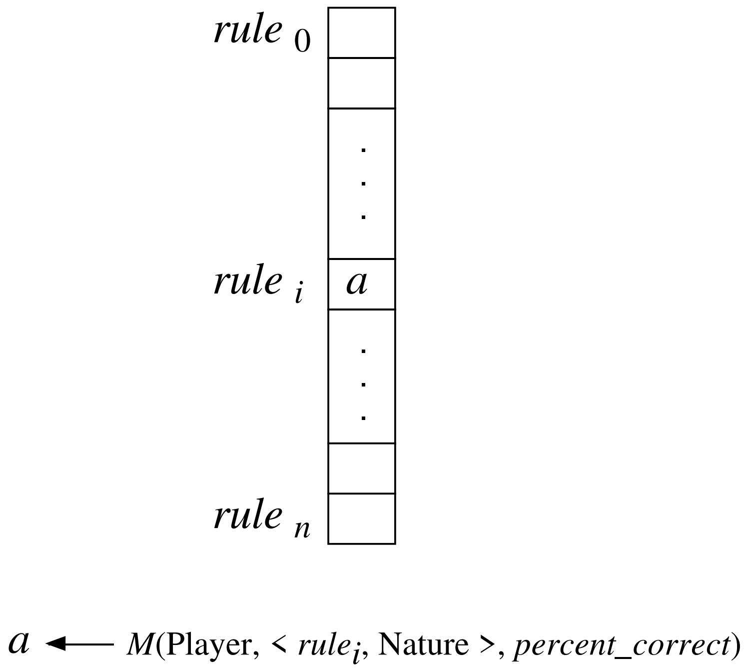


Figure 2.5: Measurement table obtained when viewing majority function as a game against nature.

metric of goodness for Player 1 returns a value of one if the rule correctly classifies the IC, otherwise it returns zero; the metric for Player 2 is the logical complement—it returns a value of one if the rule incorrectly classifies the IC, otherwise it returns zero. Now our measurement table appears as illustrated in Figure 2.6.

2.10.3 Single-Player Multi-Objective Game Against Nature

We can view the majority function as a multi-objective single-player game against nature. Like our first example, each event consists of an interaction between a rule and nature: $\mathcal{E} = \langle rule_i, \mathcal{N} \rangle$. Unlike our first example, however, we do not have a single metric of goodness; rather, we have a multiplicity of metrics. Each metric concerns the performance of the tested rule with respect to a particular aspect of its interaction with nature. Indeed, we may have 2^{149} metrics—one for each possible

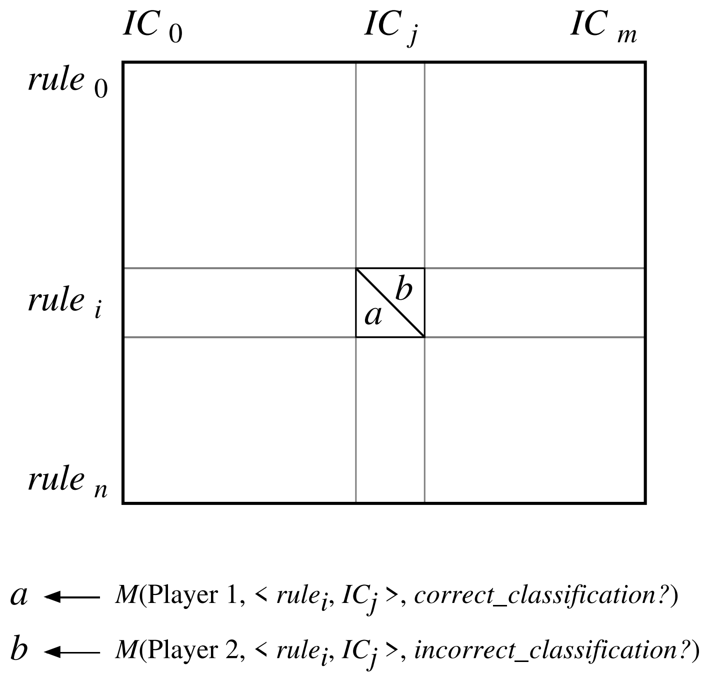


Figure 2.6: Measurement table obtained when viewing majority function as a two-player constant-sum game between rules and initial conditions.

initial condition. (Alternatively, we may have 150 metrics—one for each possible density class (0–149). Another possibility is to have just two metrics, one to measure the rule's performance on ICs of density ≤ 0.5 and the other for ICs of density > 0.5.) Our measurement table now appears as illustrated in Figure 2.7.

2.10.4 Two-Player Variable-Sum Game

We can view the majority function as a variable-sum game for two players. This time, *both* player roles represent automaton rules. An event consists of one rule interacting with another: $\mathcal{E} = \langle rule_i, rule_j \rangle$. The metric of goodness for Player 1 v_1 indicates the number of initial conditions that $rule_i$ correctly classifies that $rule_j$ does not; similarly, the metric for Player 2 v_2 indicates the number of ICs that $rule_j$ correctly classifies that $rule_i$ does not. (Thus, table entries on the main diagonal are zeros.)

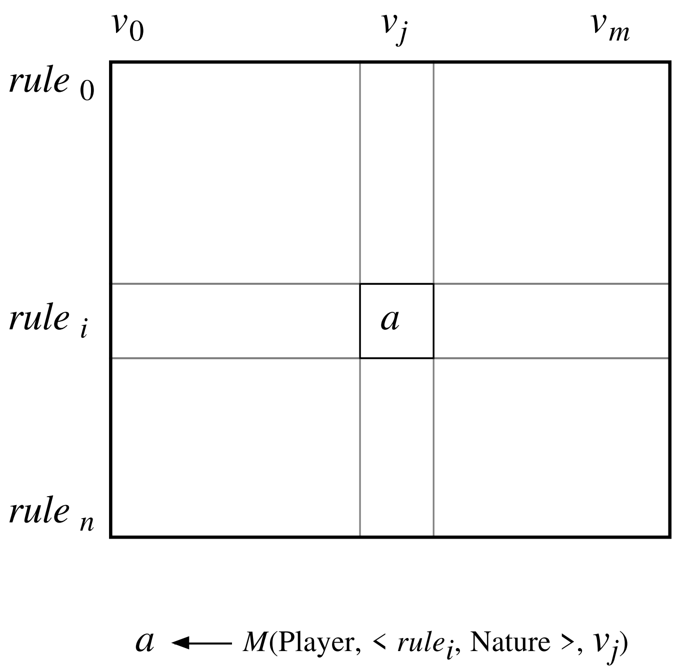


Figure 2.7: Measurement table obtained when viewing majority function as a multi-objective single-player game against nature.

This approach is essentially that used by Juillé and Pollack [102] for the *intertwined spirals problem*. Our measurement table now appears as illustrated in Figure 2.8.

2.10.5 Another Single-Player Multi-Objective Game Against Nature

Just as we transform the constant-sum game in Section 2.10.2 into a multi-objective optimization problem in Section 2.10.3, so too can we transform the variable-sum game in Section 2.10.4. An event consists of a rule interacting with nature: $\mathcal{E} = \langle rule_i, \mathcal{N} \rangle$. In contrast to Sections 2.10.1 and 2.10.3, the role of nature is now to *compare* the performance of the tested rule against those of all other rules with respect to all possible initial conditions. As in Section 2.10.3, a rule's interaction with nature is observed from multiple perspectives. For example, we may have 2^{128} metrics

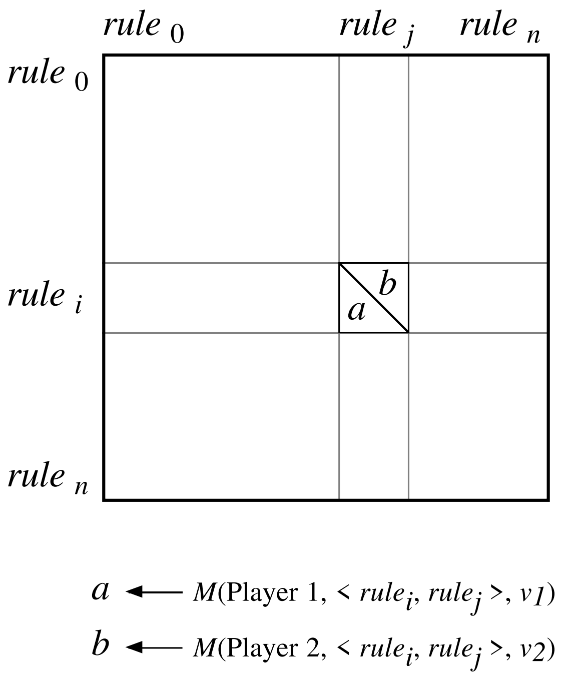


Figure 2.8: Measurement table obtained when viewing majority function as a two-player variable-sum game between two rules.

of goodness, where metric v_j indicates how many initial conditions $rule_i$ correctly classifies that $rule_j$ does not. Rule comparison is achieved not through direct "rule-to-rule interaction," but rather through the indirection of nature acting as a proxy for all rules. Our measurement table now appears as illustrated in Figure 2.9 (note that the main diagonal again contains zeros). While this example may appear contrived, it does display the critical conceptual shift that invites the application of multi-objective optimization methods.

2.10.6 Robustness over Multiple States of Nature

Rather than seek a value x that maximizes $f(x)$, we may instead care to find a value of x that is most robust when a noise term μ is added to x; that is, we wish to find a value of x that maximizes the expected value of $f(x + \mu)$, given the distribution

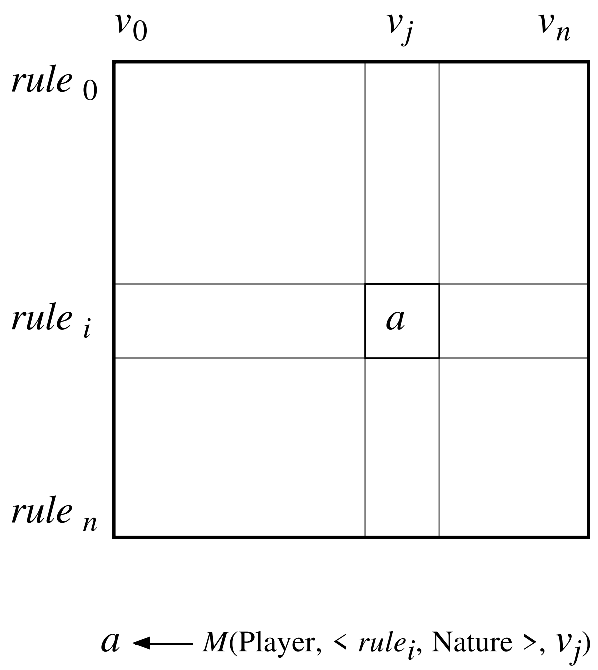


Figure 2.9: Measurement table obtained when viewing majority function as a different multi-objective optimization problem.

of the noise term μ. The addition of our noise term transforms what is otherwise a single-objective game against nature into a *multi*-objective game against nature where nature has multiple states. Each possible value of μ maps to a distinct state of nature, and we measure the performance of our behavior x in each of them. Note that the solution to this problem may be a probability distribution over our behaviors. Research in *dynamic landscapes* (e.g., Branke [26]) can be understood as a single-player game against multiple states of nature. As another example, we may consider the robot-locomotion domain, where the surface on which the robot moves can change.

2.10.7 Measurement Tables and the Secondary Search Effort

Our measurement table represents complete knowledge of our *de facto* domain: We know what behaviors are available, we know the outcomes of all interactions, and given a solution concept for our primary search problem we know the set of solutions (to our primary problem). But, as we explain in Chapter 1, coevolutionary optimization requires two distinct search efforts: To recognize solutions in our primary search effort, we need to discover appropriate interactions. Thus, given a measurement table **T**, we require the solution concept for our secondary search effort to yield a set of interaction partners that is sufficient for our primary search effort to succeed. Thus, solutions to our secondary search effort are not unlike Rosin and Belew's teaching sets [174] and Goldman and Kearns' teaching sequences [177]. Many of the pathologies encountered in coevolutionary optimization occur because the solution concept for the secondary search effort does not yield such a set of interaction partners. We discuss these pathologies in detail in Chapter 3 and provide heuristics to yield "helpful" interaction partners in Chapter 7.

Chapter 3

A Taxonomy of Issues and Research

3.1 Introduction

This chapter presents a taxonomy of the pathologies associated with coevolutionary algorithms. In so doing, a critical review of the coevolution literature is made, as well. (Two accounts of coevolution research through 1997 are found in Paredis [160] and Angeline [8].) To begin, let us briefly review our approach outlined in Chapters 1 and 2. We have a problem domain of interest and a solution concept that defines certain locations of our search space to be solutions. Because the elements of our search space must interact with each other to reveal their properties, we are required to pursue a secondary search effort to obtain relevant interactions; without the appropriate set of interactions, we cannot identify the solution to our primary search problem. The pathologies encountered with coevolutionary algorithms arise when we select a poor solution concept for either the primary or secondary search effort, or when we improperly implement solution concepts. We will thus taxonomize the pathologies according to which search effort is flawed.

This chapter is organized as follows. Section 3.2 reviews the potential benefits of coevolutionary optimization; Section 3.3 reviews methods that have been used to monitor progress in coevolution; Sections 3.4, 3.5, and 3.6 review the pathologies of gradient loss, cycling, and forgetting, respectively; Section 3.7 discusses a little-known effect in fitness measurement that can obscure the identity of solutions in certain situations; Section 3.8 concludes the chapter with a discussion of diversity maintenance and its connection to the secondary search effort.

3.2 Why Use Coevolution?

Before we review the pathologies associated with coevolution, let us first discuss the potential advantages of coevolutionary optimization. There are four principal reasons to use a coevolutionary algorithm for machine learning:

1. There exist domains that do not require coevolution, but coevolutionary algorithms make more efficient use of finite computational power by focusing evaluation effort on the most relevant tests—for example, those that best distinguish the quality of potential solutions.

2. There exist domains that intrinsically require coevolution; these are domains that are interactive in nature, such as games.

3. There exist domains that require less (human-supplied) inductive bias when using coevolution than when using other search methods.

4. There exist domains that are "open-ended"; that is, there exists an infinity of possible behaviors.

We now elaborate on these points and discuss examples.

3.2.1 Efficiency

Our discussion of how coevolution can improve search efficiency focuses on two well-studied problem domains: Minimal sorting networks and cellular automaton rules for density classification. In both domains, the universe of possible test cases is expressible with n-digit binary strings, and therefore is clearly circumscribed. Nevertheless, while tests are easily enumerated, the number of possible tests grows exponentially in n. For small enough values of n, we can afford exhaustive testing; otherwise, an alternative approach is required.

As we note above, the recognition of a solution requires appropriate testing. Exhaustive testing is certainly sufficient to recognize a solution, but is also unnecessary for the sorting network and density classification domains, as we will discuss. More generally, if traits are heritable, then we can assume—to varying degrees—that tests passed by parents will likely be passed by offspring, and so these tests can be omitted. Much of the difficulty in constructing a coevolutionary algorithm concerns understanding when this assumption is safe to make. A properly constructed algorithm extracts efficiency gains when assumptions of heritability are founded.

Minimal Sorting Networks

The seminal work of Hillis [91, 92] is our first illustration of how coevolution can improve search efficiency. Hillis seeks to discover minimal 16-input sorting networks. One certainly does not require a coevolutionary algorithm to find such networks, since the quality of a sorting network is straight-forwardly (albeit tediously) found by evaluating its performance on all possible binary inputs; for a 16-input network, we have 2^{16} such tests. Of course, as the size of the sorting network increases, the number of tests required to establish correctness grows exponentially. Thus, if computational resources prohibit exhaustive testing, some form of test-case sampling must be used instead.

Hillis first tries to evolve sorting networks using random samples of inputs for evaluation. He reports, however, that much of the testing effort is wasted; most of the tests obtained through random sampling are soon solved by all of the evolving networks, thereby providing no gradient for selection to operate. (Indeed, as Juillé reports [99], all but 151 tests are solved by the initial 32 comparators that Hillis uses to seed the initial population.) Thus, Hillis next tries to competitively coevolve the test-case samples such that they remain appropriate to the abilities of the evolving networks as they improve. Not only does Hillis obtain better results through the use of coevolution, but he also finds a sorting network with 61 comparison-exchange operations—only one more than the currently known minimal network by Green [113].

Though Juillé [99] later obtains a 60-comparator network with a non-coevolutionary algorithm, he does so by building on the first 32 comparators of Green's solution and taking advantage of the small number of remaining test cases (151) needed to establish a network's correctness; thus, Juillé uses exhaustive testing. Hillis, in contrast uses sample sizes of 10–20 test cases in evaluation. The improvement by Juillé notwithstanding, Hillis compellingly demonstrates the potential of coevolution to dynamically adjust the focus of evaluation effort and successfully optimize.

Automata Rules for Density Classification

One of the more impressive testaments to coevolution's ability to optimize is found in Juillé and Pollack's [104, 105, 106] work on the *majority function*. This problem comes from the study of binary 1-D cellular automata and concerns the ability of an automaton to determine a global property of an initial condition (IC) from local information. In particular, we wish to discover an automaton rule that will cause the automaton to converge to a state of all ones if the IC has more ones than zeros, and converge to a state of all zeros otherwise. The canonical form of this problem uses an IC of 149 bits, a rule of radius 3, and allows 320 time steps for the automaton to converge. Land and Belew [119] have shown that no perfect rule for this classification task exists, but the maximal possible classification rate is yet unknown. (We discuss the majority function in detail in Chapter 7.)

Very much like the sorting network problem, the set of test cases is clearly delimited; we can easily generate a test, but the number of possible tests (2^{149}) makes exhaustive evaluation impossible. The accuracy of rule classification is consistently seen to diminish, on average, as the initial condition's density approaches 0.5 [143, 106]; rules that correctly classify difficult densities, on average, also correctly classify easier densities. Thus, the space of rules approximately forms a total ordering when rule performance is regarded with respect to IC densities; similarly, the space of IC densities also approximates a total ordering. Note that these orderings lack the anti-symmetric property, since two rules (or IC densities) may have identical performance, yet be different rules (or IC densities). In contrast, when rule performance is regarded with respect to individual initial conditions (i.e., not densities), Land and Belew's result suggests that the space of rules (and the space of IC instances) is follows a partial ordering. For this reason, most learning methods applied to this problem use IC density classes as the basis for testing, rather than specific initial conditions; doing so produces smoother orderings that help better organize test results.

Most of the evolutionary approaches to solve the majority problem use a fixed distribution of IC densities for evaluating rules. For example, Mitchell, et al [143, 142] and Das, et al [43] uniformly sample IC densities between 0 and 1; this produces an even distribution of IC difficulty. These studies use modest population sizes: 100 rules and 300 [143] or 100 [142, 43] test cases (ICs). In contrast, Andre, et al [6, 5], uniformly sample the space of initial conditions themselves (not IC densities); the resulting distribution of IC densities is binomial, meaning that the most difficult densities are strongly emphasized. Das, et al, point out that sampling densities binomially tends to prevent progress from the very beginning of an evolutionary run—since ICs of appropriate difficulty (i.e., very low or very high density) are so sparsely sampled, little gradient exists to distinguish the rules and allow evolution to progress (much like random sampling in the sorting-network domain reveals little gradient, except there random tests are too easy); at the same time, they acknowledge that uniform-density sampling becomes less effective as rules improve, since the more difficult tests become increasingly under-sampled, again leading to a lack of gradient. Andre, et al, increase the amount of gradient they obtain from their binomial sampling simply by taking more (specifically, 1000) samples of the IC space; they also use a larger (by two orders of magnitude) rule population size of 51200. Their approach yields a rule that is significantly better than that of Das, et al: 82.3% vs. 76.9% when tested against a uniform sampling of ICs.

The first attempt to use coevolution with the majority problem is by Paredis [161]. His approach coevolves rules with actual initial conditions rather than density classes. The two populations are placed in a competitive framework. Despite the use of *lifetime fitness evaluation* (LTFE) to integrate individuals' scores over multiple fitness evaluations, an oscillatory dynamic is observed (we discuss LTFE more below, in connection with evolutionary forgetting). This behavior is caused by an intransitive cycle similar to that seen in the *matching pennies game* [82]: If the IC population contains more initial conditions with density < 0.5 than > 0.5, then a simple "always converge to zeros" rule will likely out-perform any slightly more sophisticated rule, for example one that accurately differentiates between very low and very high densities but not otherwise. Once such converge-to-zero rules gain a majority, then a pressure arises for ICs to increase their density beyond 0.5. In turn, this creates an environment for a simple "always converge to ones" rule, an so on. To dampen the oscillations, Paredis essentially reverts to a non-coevolutionary approach whereby the IC population retains a uniform distribution of densities, as in Mitchell, et al [143, 142] and Das, et al [43]; ICs are continually replaced, but not according to fitness and without heritability. Nevertheless, the LTFE mechanism still modulates which ICs are chosen as tests, though no indication is given regarding how LTFE affects over time the distribution of densities used for testing. Paredis reports results similar to those of Mitchell, et al [142].

Thus, according to the framework we present in this dissertation, the purely competitive coevolutionary framework implements the wrong solution concept for what we call the secondary search effort (which is responsible for discovering the relevant tests for the primary search effort). Juillé and Pollack [106] also obtain cyclic behavior in a purely competitive framework, even when rules are coevolved against IC density classes, rather than the initial conditions themselves. They also show that a cooperative structure, where IC densities are rewarded for being solvable, quickly leads to a fixed point of simple rules. They next revert to a competitive structure but add Rosin's [172] *competitive fitness sharing* method for diversity maintenance of both rules and densities; this arrangement causes the population of IC densities to quickly converge onto 0.5—the hardest class—well before the rules are capable of solving them. Thus, Juillé and Pollack demonstrate many of the pathologies one can encounter using coevolution. In all cases, the IC densities fail to evolve such that they provide gradient for substantial improvement of the rules; the solution concept used for the IC densities is wrong in each case. Finally, Juillé and Pollack propose a modification based on entropy to arrive at a novel calculation of selective pressure for IC densities; their new solution concept—built on top of competitive fitness sharing—rewards IC densities for revealing rule performance that is distinct from random guessing. With this approach, Juillé and Pollack discover a rule that correctly classifies 86.3% of randomly sampled automaton initial-conditions; their rule is the best currently known.

The recent paper of Werfel, et al [204], asserts that the success achieved by Juillé and Pollack is attributable more to their use of competitive fitness sharing than to their additional modification, which computes entropies of rule performance. To support this assertion, Werfel, et al, construct a competitive framework where IC densities are selected according to the entropy calculation alone—without competitive fitness sharing; this arrangement produces very poor results: Only four percent of the trials yield rules with classification accuracy ≥ 0.7 and none ≥ 0.8. Further, they construct a non-coevolutionary algorithm where rule scores are nevertheless discounted according to competitive fitness sharing; this algorithm produces noticeably better results: 43 percent of the trials yield rules with accuracy ≥ 0.7 and 10 percent ≥ 0.8. The two methods combined provide the best results: 47 percent of the trials give rules with accuracy ≥ 0.7 and 27 percent ≥ 0.8. Given these results, Werfel, et al, claim that fitness sharing used with ordinary evolution provides more genetic diversity than coevolution using the entropy calculation; they conclude that coevolution merely heightens the benefit of fitness sharing when the two are used together.

Nevertheless, Werfel, et al, neglect the fact that coevolution with fitness sharing—but without the entropy calculation—also performs very poorly (as demonstrated by Juillé and Pollack [106]); in this configuration, competitive fitness sharing does not provide diversity at all, but rather causes IC densities to converge to 0.5. Further, by categorizing rule performance over four broad intervals of accuracy (the best category requires ≥ 0.8 accuracy), Werfel, et al, de-emphasize the magnitude of improvement over other methods that Juillé and Pollack's rule represents. Werfel, et al, frame their investigation as a comparison between resource sharing and coevolution; yet, doing so largely operates on arbitrary beliefs of which techniques fall under the purview of "coevolution" and which do not. Indeed, the issue of gradient creation transcends such boundaries: All of the methods examined (purely competitive coevolution, fitness sharing, and the entropy calculation) are implicated in both success and failure. In Section 3.4, we explain why competitive fitness sharing alone might not provide sufficient gradient for coevolutionary search.

To be sure, one can produce dismal or excellent results with coevolution—what we care to better understand in this dissertation are the factors that make the difference. Therefore, we cannot treat coevolution as a monolithic procedure; we must instead view it as a plurality of techniques, each of which implement different solution concepts.

3.2.2 Intrinsically Interactive Domains

Above, we consider domains that can be searched without coevolution. More specifically, these are domains in which test problems are easily enumerated and constructed, making the process of rating and recognizing solutions simple, in principle; for such domains, coevolution offers real benefits when exhaustive testing is prohibitively costly (and sometimes even when exhaustive testing is feasible, as we discuss in Section 3.2.2, below). In contrast, the domains we consider here are those in which test problems are neither easily enumerated nor constructed—this in addition to the fact that the number of test cases may be astronomical. Such domains are typically *intrinsically interactive*. By this we mean that the source of the tests used to recognize solutions and direct search is the domain itself; interactive domains characteristically lack concomitant, self-evident metrics of goodness for use in search.

Board games are canonical examples of intrinsically interactive domains. Let us say that we wish to discover the perfect strategy for the game of checkers. How do we construct a metric of quality for a checkers player? We can imagine a number of static (or at least stationary) testing environments: 1) Use all possible strategies for testing, 2) Use only perfect strategies for testing, 3) Use a sample (random or otherwise) of strategies for testing, 4) Use a hand-built metric of goodness that measures aspects of a strategy's play for testing. Alternatively, we can use adaptive testing, such as in coevolution. Of course, regardless of our approach, we must have a solution concept in mind—most particularly to integrate outcomes over multiple tests. We now examine each of these options in turn; Table 3.1 summarizes their advantages and disadvantages.

Exhaustive Testing

We can rate a strategy by having it play against all other possible strategies, but of course the potential difficulties here are several. First, as we have seen above, the number of strategies to play against may be prohibitive; indeed, if our interactive domain is open-ended, then the number of strategies is infinite. Second, we may be unable to systematically enumerate the space of possible strategies. Third, even if we can enumerate the strategies, the enumeration might require prohibitively expensive representations of a strategy (e.g., an astronomically large game tree).

Perfect Strategies as Tests

Much cheaper than using all possible tests, we can use instead only the most difficult tests, or "perfect" strategies (according to our solution concept). There exists, of course, a logical flaw in this approach: If we require a perfect strategy at the outset, then there is no need to perform search to discover it in the first place. Nevertheless,

	All test cases
Pro	Provides all available gradient.
Con	Expensive (likely intractable); we may be unable to enumerate test-case set.
	Hardest test case
Pro	Minimizes number of evaluations.
Con	May be unknown or costly to obtain; does not generally provide sufficient gradient for evolutionary learning; unresponsive to learners' performance.
	Random samples from test-case set
Pro	Allows tractable number of evaluations.
Con	Unresponsive to learners' performance; does not generally provide sufficient gradient for evolutionary learning.
	Pre-determined fixed sample from test-case set
Pro	Allows tractable number of evaluations.
Con	May allow over-fitting; unresponsive to learners' performance; test-case set that is both small and provides sufficient gradient likely difficult to obtain or non-existent; thus, does not generally provide sufficient gradient for evolutionary learning.
	Hand-built evaluator of performance
Pro	Highly targeted evaluation.
Con	Requires intimate domain knowledge and precise articulation of that knowledge; requires considerable effort to construct.
	Coadapted test cases
Pro	Responsive to learners' performance; improved provision of gradient for evolutionary learning; tractable number of evaluations; likely to require less effort than hand-building an evaluator or hardest test case; has yielded superior results in several domains.
Con	Many diverse pathologies that are known to stunt learning.

Table 3.1: Types of evaluation functions.

Angeline and Pollack [9] show that standard evolution against a perfect Tic-Tac-Toe strategy produces weaker players than does coevolution. These results agree with those of Juillé and Pollack [106] discussed above, where testing against only the hardest cases provides too little gradient for search to progress. (Using only weak tests gives essentially the same problem.)

Note that the categorical nature of Tic-Tac-Toe outcomes (i.e., win, lose, draw) means that testing against the perfect player gives a very coarse-grained gradient; as a result, the fitness landscape has vast fitness-neutral plateaus. In contrast, testing against the most difficult test case in the majority function (density class 0.5) yields a much finer granularity—rules are known to solve between 50 and approximately 86 percent of these test cases (of which there are astronomically many). Nevertheless, the peaks of the rules' fitness landscape remain surrounded by sufficiently large plateaus that evolution cannot locate a peak simply by testing against the 0.5 density class.

Random Sampling of Tests

As we describe above, Hillis [91, 92] experiments with random samples of test cases. In the sorting network domain, this approach fails to provide sufficient challenge to the evolving networks; the result is a fitness landscape that fails to distinguish between networks once they achieve nominal performance. In the domain to Tic-Tac-Toe, Angeline and Pollack [9] conduct an experiment (in addition to the one described above) where they evolve against a player that behaves randomly; each test against the "random" player consists of four games. Thus, their approach is similar to testing against a random sample of deterministic opponents. The best strategy they obtain in this way beats the random player 56.25% of the time, yet it consistently loses to more competent hand-built opponents. In contrast, when players are coevolved, Angeline and Pollack report far more robust strategies.

Fixed Subset of Tests

While not an intrinsically interactive domain, Pagie and Hogeweg's [155] function approximation problem nicely illustrates how coevolved tests can provide more effective and efficient learning than a fixed set of test cases. With genetic programs as their evolutionary substrate, they seek to induce a continuous, two-dimensional function. Because the function is continuous, there exists an infinity of test cases; thus, one cannot attempt exhaustive testing.

In one experiment, Pagie and Hogeweg evolve their genetic programs using a fixed set of 676 test cases that densely and uniformly covers the domain of the function (which is limited with a bounding box). A spatial evolutionary algorithm is used, with a square lattice of 50 units per side. In a contrasting experiment, small subsets of the 676 test cases are coevolved for evaluating the genetic programs. Like the work of Hillis [91], the test cases are also spatially distributed on the lattice; each evolving genetic program is evaluated by the nine test cases in its immediate neighborhood. Thus, each genetic program can have a different set of nine coevolved tests.

Over twenty trials of each experiment, none of the trials that use the fixed test-set result in a genetic program that accurately fits all 676 data points. In contrast, 45 percent of the coevolution trials are reported to yield genetic programs that fit all 676 data points. Further, the genetic programs obtained through coevolution are shown to generalize better when tested on data points outside of the original set of 676 tests. If we consider the set of 676 test points to be the entirety of the function's domain, then Pagie and Hogeweg's results are evidence against the use of exhaustive testing, as well.

Thus, we have the surprising result that the larger, fixed test-set produces genetic programs that over-fit the test cases whereas the smaller, coevolved test-set produces better generalization *even though the coevolved tests are selected from the fixed test-set*. The sequence in which test-cases are presented and mastered therefore impacts the course of search. Pagie and Hogeweg remark that the highly focused testing done by coevolution allows the genetic programs to more easily cross fitness "valleys" and subsequently discover higher peaks. (This increased latitude to make errors can, however, lead to a phenomenon known as *evolutionary forgetting*, which we discuss in detail in Section 3.6 and Chapter 8.)

Hand-Built Evaluator of Play

Angeline and Pollack [9] hand-build three different Tic-Tac-Toe strategies for use as evaluators of play: A perfect player, a near-perfect player (that loses only when "forked"), and a player that behaves randomly. These fixed strategies are used as touch-stones of opponent quality. Another approach to hand-building a metric of goodness is to characterize aspects of expert play and rate players according to the number of such characteristics they display when interacting with others. Such a metric is difficult to construct because it requires both intimate knowledge of the domain and its precise articulation. Further, one might be uncertain that all relevant characteristics are captured; indeed, in open-ended domains, one cannot in principle use such an approach at all. Nevertheless, the early work of Samuel [178] closely follows this idea of a hand-built evaluator; Samuel makes extensive use of domain-specific knowledge to search for a good checkers player. Hand-built evaluators represent considerable investments of human effort, and we discuss them in more detail when we consider the supply of inductive bias.

Adaptive Testing: Coevolution

There exists an evaluation method that is more convenient than hand-crafting metrics of goodness, more tractable than exhaustive testing, and more responsive to the learners' performance than sampling test cases: We simply play strategies against each other (as they are discovered) and use the outcomes as indicators of over-all quality. Broadly speaking, this is the approach used in coevolutionary optimization. Thus, we can regard coevolution as a heuristic to obtain a metric of merit for interactive domains.

There exist some very impressive results from the use of co-adaptive and coevolutionary methods for intrinsically interactive domains. To begin, we can consider the work of Tesauro [191], who uses temporal difference (TD) learning and self-play (i.e., co-adaptation) to obtain a world-class backgammon strategy. While Tesauro attributes the success largely to TD-learning, the follow-up work of Pollack and Blair [164] shows that self-play can achieve surprisingly good results when coupled with even a very weak learning algorithm, essentially a variation of hill climbing. We may expect such a combination to be susceptible to a number of problems, such as over-specialization, forgetting, and cycling. But, these common pathologies appear remarkably absent. Instead, Pollack and Blair discover monotonic improvement against a fixed reference strategy (which is not used for directing search). The lack of brittleness appears to result from properties inherent in the game of backgammon; skills learned at any level of proficiency remain relevant against more advanced players and so are constantly cultivated; this causes skill accumulation, rather than over-specialization and forgetting.

More recently, Chellapilla and Fogel [34, 35] revisit the checkers domain. Rather than use hand-built metrics such as Samuel does, they coevolve a neural network to evaluate board positions for game-tree search. Using a rather conventional coevolutionary algorithm, they are able to obtain an expert-level checkers player. We contrast this work with that of Samuel in more detail when we come to the topic of inductive bias.

3.2.3 Open-Ended Domains

Perhaps the most dramatic illustration of coevolution's potential is Sims' [185] coevolution of virtual creatures that compete to obtain control of a cube. The domain in this work is essentially open-ended—though a contest between two agents is finite in length (eight simulated seconds), no constraints on agent behaviors are stipulated. The substrate used to express agent morphology is generative, and therefore potentially capable of an infinity of behaviors; nevertheless, limits on the depth of the generativity are placed, making the *de facto* strategy space finite. We discuss open-endedness further in Section 9.7.

3.2.4 Inductive Bias

In the domain of checkers, Samuel [178] constructs a polynomial equation where individual terms correspond to particular aspects of a strategy's play; term coefficients weight the relative importance of these aspects. This polynomial is used to evaluate board positions generated in game-tree search. Using a type of coadaptive hill-climbing, which entails self-play by the machine, Samuel adjusts which terms are used by the polynomial and their coefficients. The polynomial is restricted to use 16 out of a possible 38 terms that are defined; Samuel's hope is to supply the system with more terms than necessary and have the adjustment procedure locate those relatively few that are most salient. In later work, Samuel [179] introduces a number of improvements; among them, he more deeply considers interaction effects between polynomial terms. His methods ultimately produce a player that can beat him, but is otherwise far from expert level.

The recent work of Chellapilla and Fogel [34, 35] revisits the checkers domain. In this work, the use of game tree search is retained (though not to the same extent), but the board evaluation function is now implemented by a neural network that receives as input multiple overlapping subsections of the board; in particular, unlike the 38 terms available to Samuel's polynomial evaluation function, each of which represented a different hand-crafted metric of play quality, Chellapilla and Fogel's neural network is not given any such domain knowledge as a foundation—it is treated as a "black box." Coevolution is used to adapt the weights of the neural network. In contrast to Samuel's results, Chellapilla and Fogel discover a strategy that plays expert-level checkers; through coevolution, the neural network is able to learn a far more effective board evaluation function than Samuel's hand-designed metrics achieve. Thus, when given an appropriately pliable substrate, coevolution can discover the salient characteristics of a good strategy that are otherwise difficult to articulate by hand.

The domain of robot locomotion provides another example of how coevolution can reduce the need for inductive bias. Lipson and Pollack [129] coevolve robot morphology and control (so-called "body-brain" coevolution). Complex morphologies are much easier to build than to control; indeed, there exists a ubiquity of complex morphologies that can only be controlled remotely by humans (and often with difficulty), for the learning tasks posed by these morphologies is beyond the capabilities of available learning systems. To learn to control a complex morphology, one cannot start *tabula rasa*; instead, the learning task will be greatly simplified if the controller is already pre-adapted for a similar, thought somehow less complex morphology. In this manner, body-brain coevolution provides a way to gradually achieve both complex morphology and competent control of that morphology, without the considerable inductive bias that is otherwise required.

Finally, we return once again to the work of Angeline and Pollack [9], who use coevolution to obtain Tic-Tac-Toe strategies. A particularly interesting aspect of this work is that the evolving players are not informed of the game's rules—learning to play *legal* Tic-Tac-Toe is part of their task. If a player makes an illegal move, then its turn is forfeited. Thus, the learning system must be able to recognize an illegal move, but building such a capability represents much less inductive bias than building a legal move generator.

3.3 Monitoring Progress

An individual's observed quality is a function of the interactions used to test that individual; that is, observed quality (and ultimately fitness) is contextual. For example, in a zero-sum game, a good individual interacting with superior individuals will appear poor; on the other hand, a mediocre individual interacting with poor individuals will appear superior. Thus, if we simply monitor population fitness values (whether mean or maximum), we cannot reliably detect coevolutionary progress—the evolution of increasingly optimal behaviors. This difficulty is a manifestation of van Valen's Red-Queen Effect [197].

To deal with the problem of detecting progress, a number of approaches can be used. Interestingly, several methods to detect and monitor progress relate strongly to various *memory mechanisms*, which we review below; the connection is not coincidental, as the prevention of evolutionary "forgetting" and the task of monitoring progress can both be addressed by maintaining a history (or "memory") of where search has explored during its operation. Examples of these history-based monitoring methods include Miller and Cliff's [140] (also Cliff and Miller [36, 37]) *current individual ancestral opponents* (CIAO) and Floreano and Nolfi's [67, 68] *master tournament* (MT). A method similar to CIAO and MT is used by Rosin [172, p. 99]. All of these methods operate by collecting the most fit individuals over evolutionary time (typically the most fit in each generation) and playing them against each other; if coevolution is progressing, then individuals collected later in time will out-perform those collected earlier in time. These methods are introduced in the context of asymmetric contests (pursuit and evasion in the case of [140, 36, 37] and board games in [172]), but are easily applied to symmetric games, as well.

Another method that collects individuals is Stanley and Miikkulainen's [187] *dominance tournament* (DT); this method is introduced in the context of a zero-sum symmetric "robot duel" game, though it can be used in asymmetric games, as well. Unlike the CIAO and MT methods, DT adds an individual to the collection if and only if it defeats all other individuals already in the collection. This method ensures that no intransitive cycle (which we discuss in detail below) can exist amongst the individuals in the collection. Stanley and Miikkulainen argue that this strict criterion for inclusion better reveals important evolutionary innovations. Further, the computational cost of the dominance tournament is far less than than of CIAO and MT, since we can stop evaluating a potential collection member as soon as it loses to some current member; this savings is used by Stanley and Miikkulainen to evaluate a given pair of individuals over many initial conditions and obtain a more accurate assessment of which individual of the pair wins.

The above methods are relatively generic. Other methods to monitor progress use domain-specific knowledge. For example, for the games of Tic-Tac-Toe and Nim, Rosin [172, p. 96] (also Rosin and Belew [175]) exhaustively counts the number of reachable game tree paths that lead an individual to a loss. Another example is Reynold's [170] simple pursuit and evasion domain, where agent strategies are represented by vector fields; we can easily compare strategies and contrast them to known optimal solutions. Finally, our own early work in time series generation and prediction [59, 60] uses entropy calculations to monitor the complexity of agents' behaviors.

3.4 Loss of Gradient and Disengagement

As we state earlier, coevolution entails two search problems; the *primary* search problem concerns the domain of interest, while the *secondary* search problem concerns the discovery of interactions that will allow us to search the primary domain effectively and recognize solutions. In many cases, these two search problems are easily identified. For example, in cellular-automaton research, the primary search problem typically is to find an optimal automaton rule; the secondary search problem is to find appropriate automaton initial conditions. In other situations, the two search problems are more intertwined. For example, in the case of checkers, strategies interact with other strategies; here, test and tested are not categorically distinct. Even if we decide to split strategies into two separate populations, according to whether they play black or red, our goal most likely is to discover good strategies for both roles, not just one. Thus, in such domains the distinction between primary and secondary search problems is relative—red players act as tests to help find good black players and vice versa.

Let $\mathcal{L}$ denote our current set of evolving individuals (i.e., search space locations) in our primary search problem; we need to evaluate the members of $\mathcal{L}$. Let $\mathcal{T}$ denote our current set of interactions, obtained through the secondary search effort, that we have available for testing the members of $\mathcal{L}$. If no member of $\mathcal{T}$ can distinguish any two members of $\mathcal{L}$, then we have a *loss of gradient* in the primary search effort. When the primary and secondary search problems involve separate populations, then a loss of gradient means that the populations have become *disengaged*. (Of course, loss of gradient and disengagement are more realistically viewed as a matter of degree rather than a categorical phenomenon.) Let us assume a state of gradient loss between $\mathcal{L}$ and $\mathcal{T}$; further, let us assume that there exist members of $\mathcal{L}$ that are sub-optimal. Given the state of $\mathcal{L}$, if the solution concept for the secondary search problem $\mathcal{O}_\mathcal{T}$ judges the members of $\mathcal{T}$ to be superior to (secondary) search space locations not in $\mathcal{T}$, then $\mathcal{O}_\mathcal{T}$ is either improper or has been incorrectly implemented.

3.4.1 Examples

We have already considered several examples of gradient loss. Juillé and Pollack [104, 105, 106] coevolve cellular automaton rules with automaton initial conditions for a density classification task. In one experiment, they use a competitive framework between the two populations along with Rosin's competitive fitness sharing [172] to maintain test-case diversity. Competitive fitness sharing does maintain phenotypic diversity, but only in the absence of a clearly superior alternative; in the case of Juillé and Pollack's illustration, once automaton rules achieve a nominal level of competence, the IC density that minimizes rule performance is 0.5, and so other densities are quickly lost. Tests with density 0.5 are also the most difficult to accurately classify, and they arrive well before the automaton rules are ready for such a challenge; disengagement is the result. Juillé and Pollack also illustrate that a strictly cooperative relationship leads to gradient loss, as the IC densities are not incited to challenge the rules.

Watson and Pollack's [202] "numbers" games are geometric in nature and so provide easily visualized domains in which to observe phenomena such as disengagement. We discuss one such game in detail in Chapter 8. The numbers games vividly illustrate how two populations can become disengaged and, further, what can happen afterwards: When every individual in one population beats every individual in the other population, neither population has selection pressure acting on it. As a result, both populations enter spontaneously created regions of fitness-landscape flatness (i.e., networks of *selective neutrality* [112, 22]). The populations then drift solely according to the biases of the variation methods. In the case of the numbers games, this drift eventually brings the two populations back into "contact" and gradient is re-established. (This drift causes another pathology known as *evolutionary forgetting*, which we discuss below. Of course, to the extent that drift provides a way to repair a state of disengagement, we may consider it beneficial, as does Rosin [172, p. 104]. Nevertheless, forgetting due to drift is not a satisfying remedy to disengagement; we discuss an alternative in Chapter 7.)

Our early work [59, 60] provides an excellent example not only of gradient loss, but also of how pathologies follow from improper solution concepts. In this work, we examine a domain that is essentially a repeated *matching-pennies game* [82]. We have one population of *generators* that ballistically produce binary time series; a generator is implemented as a recurrent neural network that has no inputs and one output. We have another population of *predictors* that observe generator behavior and attempt to predict future behavior; these are implemented as recurrent neural networks that have one input and one output.

Figure 3.1 illustrates the interaction between a generator and predictor. At each time step t, both the generator and predictor simultaneously output a binary digit; the goal of the predictor is to output the same bit as the generator. The output of the generator at time t is observed by the predictor such that the output can be taken into account to help predict the generator's output at time $t+1$; the generator does not observe the predictor's behavior and therefore has no indication about the performance of the predictor during their interaction (the generator's only feedback comes at the end of their interaction).

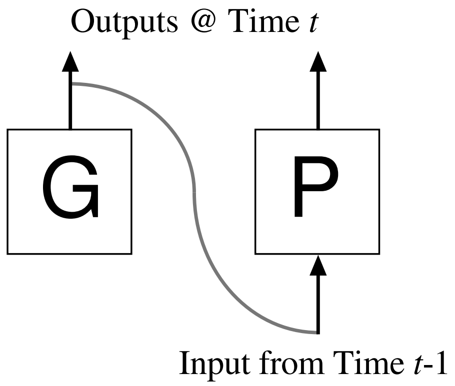


Figure 3.1: Interaction between a generator and predictor.

If our goal (i.e., primary search problem) is to evolve good time-series predictors, what solution concept should be used for evolving generators (i.e., the secondary search problem)? If we reward generators for being predictable, then the generator and predictor populations converge onto either "always output one" or "always output zero" behaviors; the generators make the predictors appear perfect, and so further progress is prevented. Alternatively, we can place the two populations in a zero-sum setting, where the predictor gets one point for every time-step it predicts correctly and the generator gets one point for every time-step the predictor is incorrect. In this case, we obtain a classic intransitive cycle that leads to oscillatory behavior; we will discuss this more below. Another alternative is to reward the predictor only to the extent that it predicts better than random guessing; prediction rates $\leq 50\%$ give the predictor minimal reward and the generator maximal reward. This reward structure causes the generators to quickly evolve high-entropy behaviors that cannot be predicted better than random guessing, regardless of how clever the predictor is.

Thus, we again obtain a lack of gradient. (This case in particular illustrates that a purely competitive framework does not necessarily keep landscapes dynamic and therefore help evolution escape local minima, as Hillis [92] suggests.)

Finally, we try a novel three-population framework, where generators evolve to be predictable to "friendly" predictors and simultaneously unpredictable to "hostile" predictors; the three-way interaction that occurs in illustrated in Figure 3.2. Predictors still receive minimal reward for prediction rates $\leq 50\%$, but now the generator obtains maximal reward when the friendly predictor is perfect and the hostile predictor is no better than random. This framework constrains the generator by making random behavior maladaptive; the predictors we evolve in this way are more robust. This three-player framework anticipates the notion of *distinctions* that we develop in Chapter 7.

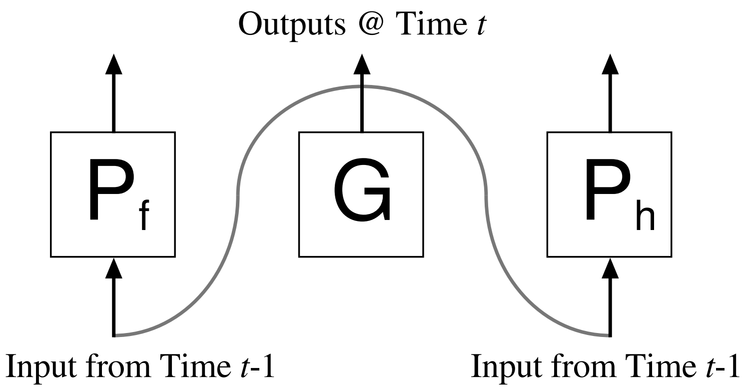


Figure 3.2: Interaction between a generator and two predictors, one friendly (P_f) and the other hostile (P_h).

3.4.2 Algorithmic Remedies

As with all of the common pathologies we discuss in this chapter, a great many researchers have addressed the issue of gradient loss. Since gradient loss is most commonly associated with two-population coevolution, research has concentrated on this case. (Gradient loss in a single population implies that all of the evolving individuals are receiving the same fitness; for example, we can imagine a symmetric zero-sum game where all of the individuals tie each other.)

Rosin and Belew's [172] *phantom parasite* method operates in conjunction with their *competitive fitness sharing* mechanism in the context of zero-sum games. As we state above, competitive fitness sharing is a diversity maintenance method. If a member α of population X defeats all members of the opposing population, then α loses by definition to the phantom parasite; if another member β in population X loses to some member of the opposing population, then β wins by definition against the phantom parasite. In this way, a niche is created in which the otherwise superior individual α is at a disadvantage; this prevents α from taking-over population X, which allows "easier" individuals to remain in population X and keep the two populations engaged. This approach clearly alters the solution concept; we will have more to say about competitive fitness sharing and the phantom parasite in Chapter 5. The experiments of Juillé and Pollack [106] that show competitive fitness sharing to lead to disengagement in the majority function problem do not use the phantom parasite method to maintain "balance" between the populations; such an experiment is certainly worth investigating.

Similar in spirit to Rosin and Belew, Cartlidge and Bullock [33] seek to discount the fitnesses of individuals who attain perfect scores against the opposing population, and thereby prevent them from taking over and causing disengagement. In particular, Cartlidge and Bullock introduce a non-monotonicity in the function that maps performance to fitness; an individual α who achieves moderate success against opponent y obtains higher a fitness contribution than another individual β who achieves very high success against y. The location of the function's peak is a parameter. Cartlidge and Bullock describe their approach as *moderating parasite virulence*. Clearly, this method changes the solution concept.

We next consider the approaches by Paredis [162] and Olsson [153]. These methods to prevent disengagement manipulate the sequence in which the two populations create offspring; when one population appears to have an advantage over the other, the evolutionary process of the stronger population is temporarily slowed or halted, allowing the weaker population more time to adapt. Note that these two methods do not alter solution concepts; that is, for any (cross-product) state of the two populations, the evaluator ratings obtained by the populations' members when we use the balancing methods are the same as when we do not use them. This is in contrast to the other methods we discuss in this section, which actually modify evaluator ratings (thereby distorting the solution concept) to affect inter-population balance.

Finally, we consider balancing mechanisms that are explicitly designed to adhere to particular solution concepts. Juillé and Pollack [104, 105, 106] evolve a population of cellular automaton rules for the majority function against another population of initial condition densities. Rather than place them in a strictly competitive framework, Juillé and Pollack additionally penalize densities that cause rules to perform similar to random guessing; the heuristic here is that such densities provide little information to guide the primary search effort.

In Chapter 7 (see also Ficici and Pollack [64]), we also use the majority function to explore the problem of gradient loss and disengagement. We introduce a matched-pair of solution concepts for the primary and secondary search problems; these solution concepts derive from the notion of *Pareto optimality*. In our approach, an IC density is rewarded for distinguishing the performance of different rules in pair-wise comparisons. The Pareto-coevolution method we introduce is subsequently pursued by Bucci and Pollack [29, 30] and de Jong and Pollack [45].

3.5 Intransitivity, Cycling, and the Red-Queen

Another pathology that has been investigated by many is *cycling population dynamics*, which is caused by intransitive superiority structures. Perhaps the best-known example of such a structure is the children's game Rock-Paper-Scissors (RPS). This is a symmetric zero-sum game for two players that has three pure strategies; these strategies are arranged in an intransitive cycle: Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. There exists a single Nash equilibrium strategy for this game, which is a mixed strategy where each of the three pure strategies is played with probability one-third.

Evolutionary game theory shows that the Nash equilibrium of the RPS game is not an attractor of the replicator dynamics [203, 93]. The standard discrete-time replicator—equivalent to conventional "fitness-proportional" selection in generational coevolutionary algorithms—causes the Nash equilibrium to be unstable; any perturbation of equilibrium leads to divergent cyclic dynamics. Figure 3.3 gives a phase plot of these dynamics. Given a finite population, such divergence will cause two of the strategies to go extinct. (Continuous-time replicators cause the Nash equilibrium to be neutrally stable; a perturbation of equilibrium produces a cycle that does not increase in magnitude.) As we discuss below, a strategy driven to extinction by intransitivity may be adaptive at some later point in time; this is an example of *evolutionary forgetting*.

Another simple example of intransitivity is found in the *matching pennies game* [82]. This is an asymmetric zero-sum game for two players. Each player has two pure strategies, *heads* (H) and *tails* (T). Player 1 (P1) scores a point (and Player 2 scores nothing) if both players choose the same pure strategy, whereas Player 2 (P2) scores a point (and Player 1 scores nothing) if the players choose different pure

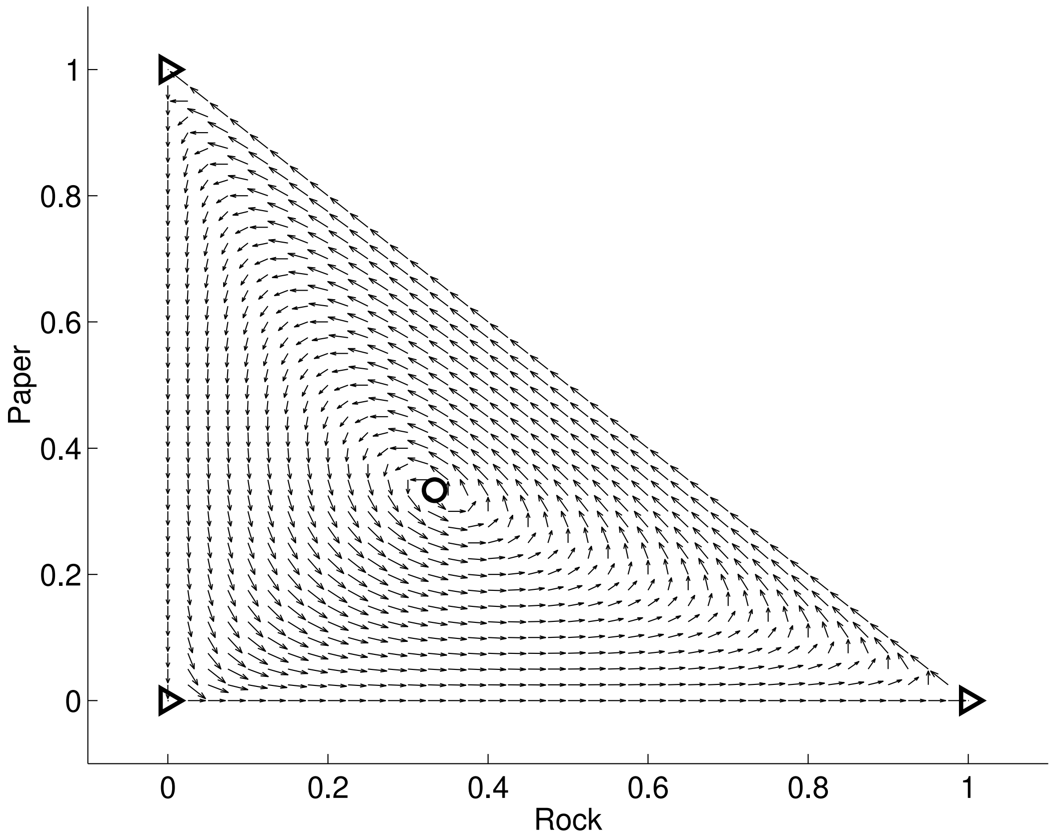


Figure 3.3: Phase diagram of Rock-Paper-Scissors game under discrete-time replicator dynamics. Circle indicates unstable Nash equilibrium; triangles indicate saddle points.

strategies. Thus, we have an intransitivity, as illustrated in Figure 3.4: If P1 plays H, then T is adaptive for P2; if P2 plays T, then T is adaptive for P1; if P1 plays T, then H is adaptive for P2; if P2 plays H, then H is adaptive for P1, and so on. The Nash equilibrium for this game is for both players to choose each pure strategy with probability one-half.

We can use the matching pennies game to illustrate cyclic dynamics in two populations. As with RPS, above, the discrete-time replicator causes the equilibrium to be unstable; any perturbation causes a divergent cycle. Figure 3.5 illustrates a sample trajectory in the cross-product state space of the two populations. The Nash equilibrium is indicated by the circle in the middle of the graph. Our initial condition is a slight perturbation of the equilibrium proportions; from this initial condition, the populations' trajectories exhibit increasingly strong oscillations between the two pure strategies.

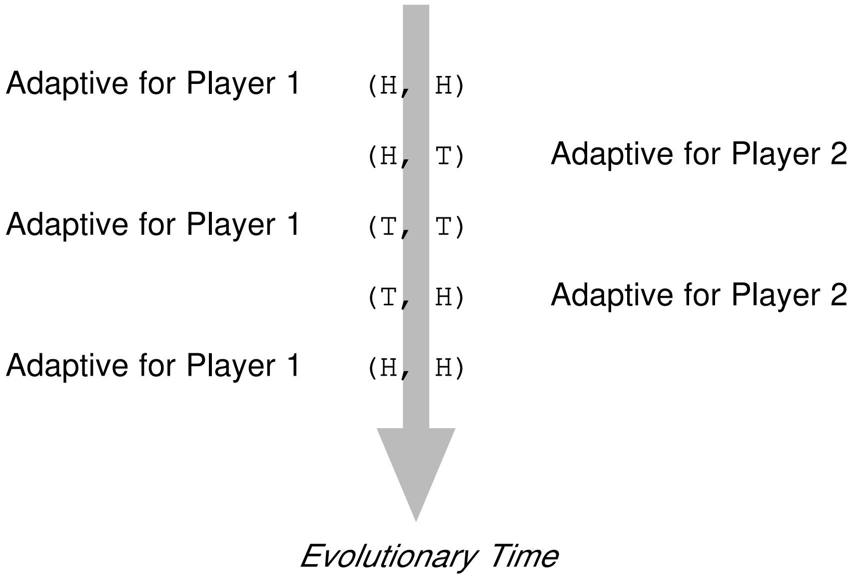


Figure 3.4: Intransitivity structure for matching pennies game.

Clearly, the standard replicator dynamic does not implement the solution concept of Nash equilibrium in the above examples. More generally, we may claim that

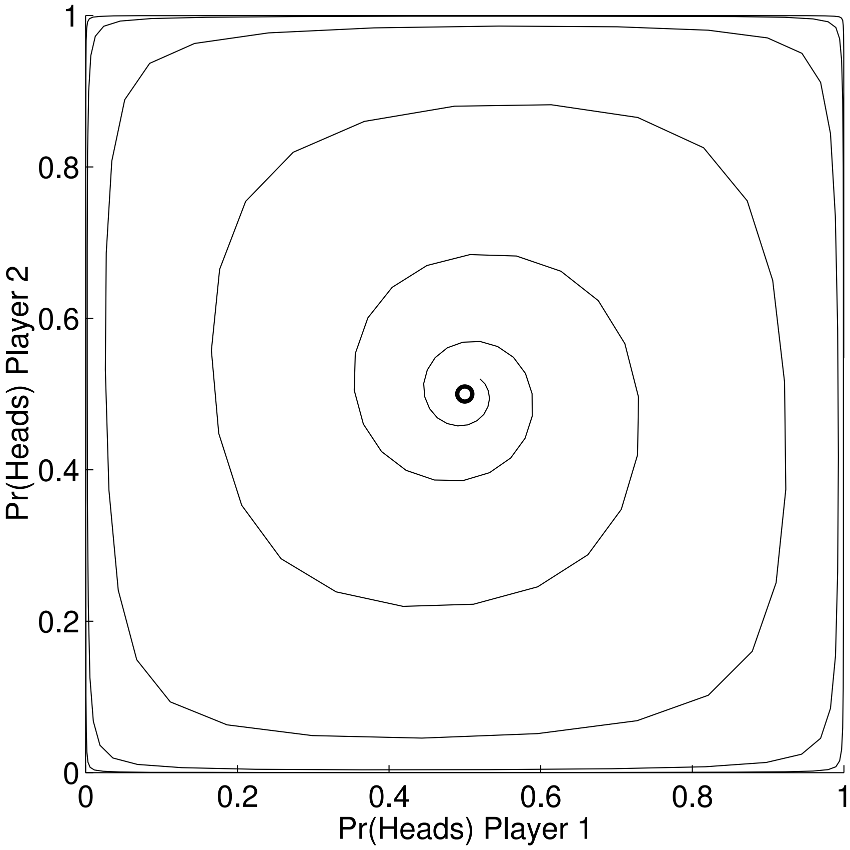


Figure 3.5: Sample trajectory of two populations in matching-pennies game. Horizontal and vertical axes indicate the proportions of Player 1 and Player 2 populations that plays Heads, respectively. Trajectory begins near unstable Nash equilibrium (circle), which is at 0.5, 0.5.

the standard replicator does not implement *any* solution concept for RPS and the matching-pennies games for the simple reason that the system does not converge. Thus, we argue that cycling manifestly indicates improper implementation of a solution concept. (Alternatively, we may interpret cycling as an indication that no solution exists.)

Discussions of cyclic dynamics almost always include a reference to van Valen's *red-queen effect* [197]. While related, these two phenomena are not synonymous. The red-queen effect refers to the fact that, to maintain a level fitness in a dynamic environment, a specie must continuously evolve. Thus, this principle certainly applies where cyclic dynamics are found—indeed, we can argue that it is this principle that animates the cyclic dynamics. In contrast, however, the red-queen effect also applies to an evolutionary "arms race" between two competing species, where each specie forces the other to become increasingly competent at certain behaviors. Because of the relativistic nature of coevolution, these two situations cannot be distinguished simply by monitoring fitness levels. For this reason, a variety of monitoring techniques have been developed, which we review above in Section 3.3. Intransitivity and the Red Queen have lead to the common perception that one cannot obtain "directionality" in coevolution, which challenges the position that coevolution can be used to optimize; we discuss this point in depth in Chapter 9.

3.5.1 Examples

We have already discussed several examples of cyclic dynamics. Both Paredis [161] and Juillé and Pollack [104, 105, 106] describe cyclic dynamics in their work on the majority function. Our early work [59, 60], where we use coevolution for a time-series prediction task, also describes cyclic dynamics between two populations. The two-population intransitivity that exists in both of these problem domains is essentially the same as that found in the matching pennies game. In a different context, Cliff and Miller [36] and Nolfi and Floreano [148] discuss cyclic dynamics in pursuit and evasion contests.

3.5.2 Algorithmic Remedies

A number of different methods have been proposed to address intransitivity and the cyclic dynamics it engenders. In the context of pursuit and evasion, Nolfi and Floreano [148] show that the effect of intransitivity can be diminished by enriching the environment in which the agents behave. By adding various static obstacles to the environment that affect agent fitness, the fitness volatility created by intransitivity is dampened. Of course, this approach is not generically applicable to all domains. Nolfi and Floreano [148] also demonstrate how the use of a *memory mechanism* can dampen cyclic dynamics; we discuss memory mechanisms more below and propose a new mechanism based upon game-theoretic solution concepts in Chapter 8.

Bullock [31] argues that coevolutionary optimization cannot succeed unless selective pressure is sufficiently broad, or "diffuse"; otherwise, the outcomes of coevolution will be overspecialized and brittle. Indeed, the cyclic dynamics obtained from intransitivity display a process of overspecialization. Hornby and Mirtich [96] adopt Bullock's argument in their work on the coevolution of pursuit and evasion behaviors. They implement a diffuse selection pressure by evolving multiple (i.e., > 2), reproductively isolated populations and having each agent interact with members of each population; the greater genetic and behavioral diversity thus maintained broadens selection pressure and dilutes the effect of intransitivity. (In addition, the static aspects of the environment are also enriched: The arena has an obstacle in the center, and a sophisticated 3D physics-based simulator is used. This echoes the work of Nolfi and Floreano [148].) Hornby and Mirtich show that their multi-population approach yields more complex pursuit and evasion behaviors than control experiments that do not use "diffuse" coevolution.

Finally, Rosin and Belew's [173, 172] *competitive fitness sharing* (CFS) provides another method to enhance phenotypic diversity and thereby broaden selection pressure. They prove that the equilibria of conventional and "shared" replicator dynamics are identical for zero-sum games—a clear (yet uncommon) example of explicit consideration for the solution concept in algorithm design. Nevertheless, Rosin and Belew do not provide a theorem regarding how competitive fitness sharing affects the *stability* of these equilibria—does competitive fitness sharing ameliorate the cyclic dynamics produced by the standard discrete-time replicator in the presence of intransitivity? To be fair, we must point out that CFS is not intended to address directly the issue of cycling; rather, Rosin and Belew's goal for CFS is to approximate *teaching sets* as understood in the field of *concept learning* [177], which they apply to coevolutionary optimization of competitive domains. We revisit the equilibrium and stability characteristics of Rosin and Belew's methods (competitive fitness sharing, shared sampling, and phantom parasite) in Chapter 5.

In Chapters 7 and 8, we introduce two entirely new algorithmic methods that systematically broaden selection pressure to subsume intransitive structures and the cyclic dynamics they can create. Our work on Pareto coevolution at once ensures that multiple populations remain engaged and intransitivity is properly handled. Our work on Nash memory mechanisms provides a way to systematically broaden selection pressure to prevent cycling and forgetting.

3.6 Forgetting

The problem of *forgetting* is another common pathology in coevolutionary optimization. As Figure 3.6 illustrates, the pathology of forgetting entails the process of *trait loss*. For our purposes, a "trait" refers to any measurable aspect of behavior. Let us say our population at time t contains some individuals with some trait x. This trait can be lost if 1) the trait is selected against—individuals with the trait are less fit, on average, than individuals without the trait, 2) the trait is not strongly acted upon by selection pressure and is left to drift according to biases in the variational operators, and 3) the trait is selected for, but is difficult to maintain—that is, the variational operators are strongly biased against it, making offspring likely to lack the trait. These causes of trait loss eventually lead to a population at some later point in time $t+k$ where no individual has trait x.

An example of trait-loss becomes an instance of forgetting when, at some yet later point in time $t+k+n$, we have a population where 1) no individual has trait x and 2) some individual would obtain a gain in fitness if it were to obtain trait x. Thus, in forgetting, a previously acquired and subsequently discarded trait is once again desired. This description may remind us of the cycling dynamic; indeed, when a trait is forgotten due to being selected against, an intransitive structure is at work. When a trait is forgotten due to drift, selection pressure has become too narrow; this problem has been called *focusing* [202]. When a trait is forgotten despite being selected for, the coevolutionary system lacks a proper *elitism* strategy.

3.6.1 Examples

The issue of forgetting is discussed by many researchers. In the context of pursuit-and-evasion contests, Cliff and Miller [36] discuss the role of intransitivity in forgetting;

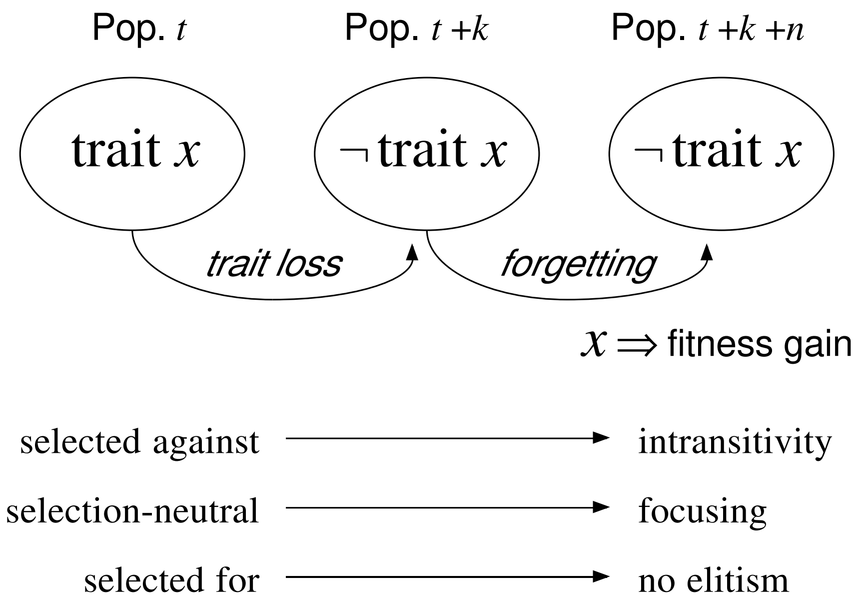


Figure 3.6: Schematic of process of *forgetting*.

Floreano and Nolfi [68] use a shallow "Hall of Fame" memory (discussed below) to help stabilize cycling, but still obtain forgetting due to intransitivity. The role of drift in forgetting is discussed by Cliff and Miller [36], Rosin [172, p. 104] (also [175]), and Watson and Pollack [202]. The numbers games of Watson and Pollack [202] provide vivid illustrations of how forgetting ensues from genetic drift.

3.6.2 Algorithmic Remedies

The general antidote to the problem of forgetting is to maintain a sufficient diversity of selection pressures. Pollack and Blair's [164] work suggests that the game of backgammon naturally provides such diverse selection pressures and is therefore resistant to evolutionary forgetting. Backgammon is believed to have this property not only due to its stochastic nature, but also because learned skills often apply broadly and remain serviceable regardless of the opponent. Thus, once a player acquires a new skill, the game provides ample opportunity to exercise it. This constant utilization of a player's entire gamut of abilities is what creates a diverse selection pressure; any genotype variation that causes skill loss is quickly exposed and culled from the population.

Boyd [25] studies a version of the Iterated Prisoner's Dilemma where the agents can occasionally make mistakes, and defect when they intend to cooperate and *vice versa*. Given the possibility of mistakes, the well-known *tit-for-tat* (TFT) strategy can become embroiled in retaliatory oscillations against itself. A slight modification, known as *contrite* tit-for-tat (CTFT), yields an IPD strategy that is more robust in an environment where mistakes are made; indeed, Boyd proves that CTFT is an evolutionarily stable strategy in such a context. In the ordinary IPD (i.e., without mistakes), we know that exploitable cooperators can invade a population of TFT (or CTFT) strategies through drift, and hence cause forgetting. The presence of mistakes distinguishes CTFT from cooperative strategies that lack punishment mechanisms, as well as from TFT; thus, mistakes provide a selection pressure to prevent forgetting of important skills.

The above examples of backgammon and the IPD show that some domains are naturally disinclined to the problem of forgetting. For domains that are susceptible to the problem, there exist several methods that are generally applicable. For example, forgetting can be ameliorated by using multiple, reproductively-isolated populations (e.g., Hornby and Pollack [96]) or diversity methods such as competitive fitness sharing (e.g., Rosin and Belew [175]).

Research to prevent forgetting also includes a number of *memory mechanisms* that maintain a collection of "good" individuals (according to some organizing principle) discovered over evolutionary time; the memory thus encapsulates a wider range of phenotypes than is typically found in the coevolving population at any one moment. The additional phenotypic variety afforded by the memory is used to augment the evaluation process, broaden selection pressure, and thereby reduce the likelihood of forgetting.

The problem of designing a heuristic memory mechanism to prevent forgetting intrinsically entails the problem of deciding what it is that we are trying to "remember"—what is our *solution concept*? That is, what collection of traits constitutes the desired or "correct" set, and what properties does this collection have? A principled answer to this question is imperative when a domain forces mutual exclusivity between certain traits, or when an evolutionary representation (genotype) cannot simultaneously encode all desired traits. Once we have our solution concept, what organizing principle do we use in the memory mechanism to obtain it? Thus, we view a memory mechanism as an accumulator of traits; a trait enters and remains in the memory only if it is "worth" remembering, according to our organizing principle.

Almost all memory mechanisms in the literature are instances of a general "best of generation" (BOG) model where 1) the most fit individual in each of the m most recent generations is retained by the memory mechanism and 2) l of the m retained individuals are sampled without replacement for use in testing individuals in the current generation. Examples of this approach are found in Sims [185] ($m = 1$, $l = 1$), Cliff and Miller [36, 37] ($m = 1$, $l = 1$—so-called *last elite opponent*), Potter and De Jong [166] ($m = 1$, $l = 1$), Rosin and Belew [175] (e.g., $m = \infty$, $l = 20$), and Nolfi and Floreano [149, 148] ($m = 10$, $l = 10$) (see also [67, 68, 69, 70]). For $m = 1$, the population itself usually acts as the repository of the "champion" (or, elite) individual; thus, the elite contributes genetic material in addition to its role as evaluator. As the value of m increases, however, the evolving population becomes a less appropriate home for the generation elites and a separate data structure is used.

In a contrasting approach, Stanley and Miikkulainen [187] propose that their *dominance tournament* (DT)—a method intended to monitor coevolutionary progress—

can be adapted for use as a memory mechanism. The principle is to retain the most fit individual of the current generation only if it beats all the individuals previously retained by the memory; that is, the "generation champion" must dominate the contents of the memory to be included. (Note, however, that domination is determined only with respect to the memory's contents.) Thus, no intransitive cycles can exist amongst the strategies in the memory. While the dominance tournament is introduced in the context of a symmetric zero-sum game, it can also apply to asymmetric games; a very similar idea is discussed by Rosin [172, p. 25] in the context of asymmetric zero-sum games.

The work of Paredis [159] introduces *life-time fitness evaluation* (LTFE), a memory method that does not collect elites. This method is used with a steady-state evolutionary algorithm. The fitness of an individual is the sum of scores obtained by the individual over its n most recent interactions with others. The LTFE approach integrates performance over evolutionary time; in so doing, it smooths otherwise rapid shifts in fitness landscapes. Rather than broaden selection pressure by retaining champions over evolutionary time, this method broadens selection pressure by integrating performance over evolutionary time. Thus, LTFE operates within the population of individuals that is being used for search.

In Chapter 8, we introduce a new memory mechanism [65] that is built around the solution concept of Nash equilibrium. Our *Nash memory* mechanism accomplishes three goals: First, it detects and incorporates intransitive structures; second, it systematically broadens selection pressure; third, it naturally represents the solution obtained by the coevolutionary effort—that is, it provides a principled elitism method for coevolution. An important result of our work in Chapters 4 and 5 is that the evolving population is not always able to effectively perform search and simultaneously represent the solution of the search effort. For this reason, our Nash memory method creates a collection of individuals that is external to the evolving population. We compare the performance of the Nash memory mechanism to that of BOG and the DT—methods that also use external collections—and contrast their organizing principles, or solution concepts.

3.7 Fitness Deception Obscures Solutions

The task of locating the solution of an interactive domain is complicated by the need to first discover the relevant tests (interactions) through which the solution can be identified. Here we investigate an interesting frequency-dependent effect that obscures the identity of monomorphic Nash equilibria that are attractors of the standard replicator dynamic. To our knowledge, this particular pathology is reported only by Dawid [44] and subsequently Ficici and Pollack [62] (discovered independently). We begin our examination of the frequency effect by looking at a game that lacks it; this game is known as a coordination game. We then introduce the Hawk-Dove-Retaliator game, which displays the effect. Finally, we examine larger, 50-strategy games that heighten the frequency-dependent effect.

3.7.1 Coordination Games

The canonical *coordination game* is a symmetric two-player variable-sum game where both players must play the same pure strategy (i.e., coordinate) to receive maximal payoff. Equation 3.1 gives the payoff matrix for a three-strategy coordination game. Each of these pure strategies forms a Nash equilibrium with itself, where each player receives a payoff of 1. Further, there exists a mixed Nash equilibrium strategy where each pure strategy (A, B, and C) is played with probability 1/3. This mixture guarantees each player a payoff of 1/3, regardless of what the other player does; thus, the mixture does not require coordination, but neither does it deliver maximal payoff.

From a dynamical systems perspective, all four of the Nash equilibria are fixed-points of the standard replicator used in evolutionary game theory. Nevertheless, only the three pure strategies are attractors; the mixed strategy is an unstable fixed-point. Figure 3.7 (top) shows the phase diagram of our example coordination game. In the middle we have our unstable mixed Nash strategy; at the corners are the three pure Nash attractors. Thus, we have three *basins* of attraction, one for each pure strategy. Given a population state $p = \langle p_A, p_B, p_C \rangle$ (where $0 \leq p_A, p_B, p_C \leq 1$ and $p_A + p_B + p_C = 1.0$), we are guaranteed to converge onto one of the pure strategies if no two of the three strategies are present in the same proportion. We can easily determine which basin of attraction p is in: Whichever strategy is the plurality of population state p is also the attractor of p. For example, if $p_A > p_B$ and $p_A > p_C$, then we are in strategy A's basin of attraction and the population will converge onto a state of all-A. In this way, evolutionary game theory provides a natural coordination mechanism.

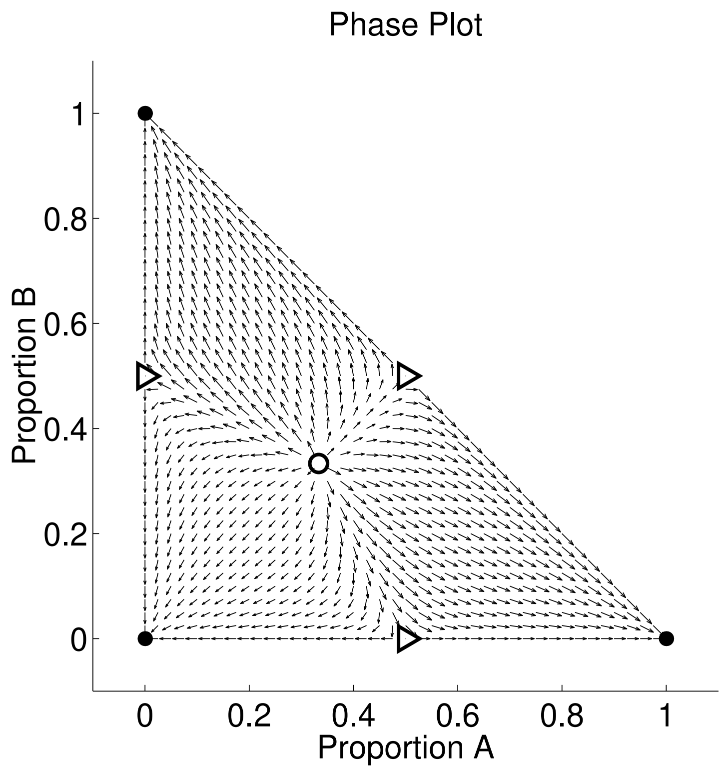
Figure 3.7 (bottom) labels points in phase space according to which of the three pure strategies receives the highest fitness. The region in which a particular pure strategy receives the highest fitness aligns precisely with its basin of attraction. Thus, given some population state p, the following points are true for our example game: 1) if a pure strategy is the plurality, then it is also receiving the highest fitness, 2) if one pure strategy receives higher fitness than the other two, then it is also in the plurality, 3) a pure strategy in the plurality will take-over the population. Therefore, for any population state in which the three pure strategies are present in different concentrations we can immediately identify the solution we will obtain—we need not wait for the population to converge.

$$
G = \begin{array}{c|ccc}
 & \text{A} & \text{B} & \text{C} \\
\hline
\text{A} & 1 & 0 & 0 \\
\text{B} & 0 & 1 & 0 \\
\text{C} & 0 & 0 & 1
\end{array}
\tag{3.1}
$$

3.7.2 Hawk-Dove-Retaliator

Equation 3.2 gives the payoff matrix for the Hawk-Dove-Retaliator (HDR) game [134]. Figure 3.8 (top) shows the phase diagram for this game. Like our coordination game, the HDR game has three monomorphic fixed-points (where the population contains only one pure strategy). Unlike the coordination game, only one monomorphic state—all-Retaliator—is attractive; the other two are unstable. The HDR game also has an unstable fixed point (more precisely, a saddle point, indicated by the triangle) where all three strategies are present. In addition to the monomorphic attractor, the HDR game has a polymorphic attractor where Hawks and Doves each compose half of the population. Of the two attractors, the basin of attraction for all-Retaliator is the larger.

Figure 3.8 (bottom) labels points in phase space according to which of the three pure strategies receives the highest fitness. Unlike our coordination game, we find that the regions indicated by the bottom graph do not align with the basins of attraction. Most particularly, in most of the Retaliator's basin of attraction, the Dove receives the highest fitness. For example, the population state $p_H = 0.1, p_D = 0.5, p_R = 0.4$ (which we can more conveniently denote $\langle 0.1, 0.5, 0.4 \rangle$) is in Retaliator's basin of attraction. Yet, in this state, the pure strategy Dove gives every indication of superiority: Dove is the most frequent pure strategy and it out-scores the other two (unnormalized fitnesses are $f_H = 0.5, f_D = 0.86, f_R = 0.85$).



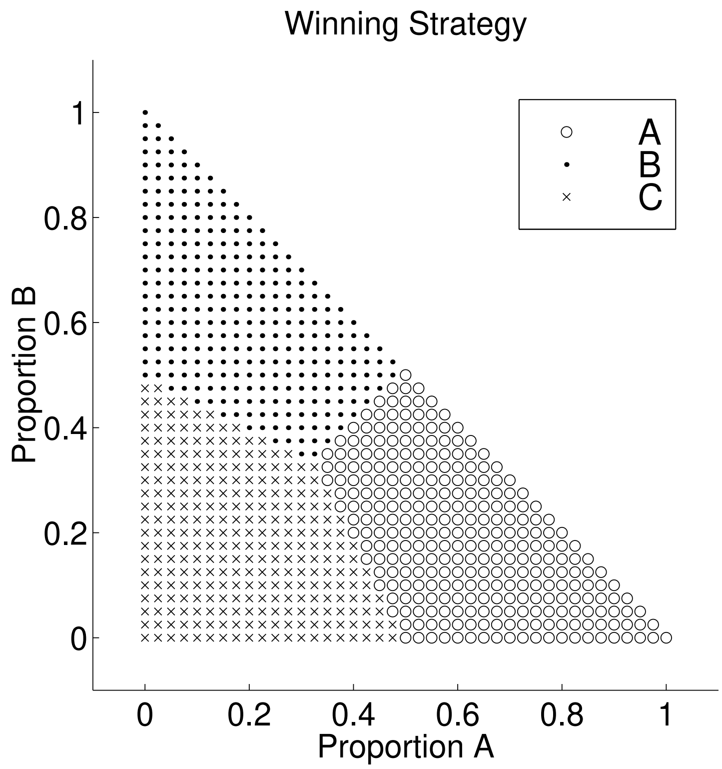


Figure 3.7: Three-strategy coordination game. Top: Phase plot of three-strategy coordination game. Bottom: Phase space labeled by which strategy has the highest fitness.

Figure 3.9 shows an orbit starting at population state $\langle 0.8, 0.01, 0.19\rangle$. We are again in the Retaliator's basin of attraction, yet Retaliator does not out-score Dove until the tenth iteration of the replicator. Thus, even if a monomorphic attractor exists, and we are in its basin of attraction, frequency-dependent effects can cause fitness values to obscure the identity of the attractor. The attractor (i.e., solution) for a particular population state need not obtain the highest fitness in that state. In this sense, fitness values may "deceive" us. Note that this form of deception does not involve the genotype space at all (as it does in GA trap functions [85], for example)—it exists purely within the game-theoretic relationships of phenotype proportions. (Note also that this frequency-dependent effect is unlike that found in Rocks-Paper-Scissors in that RPS lacks a monomorphic attractor.)

The HDR game raises the question of how conventional coevolutionary algorithms might be misled by this frequency-dependent effect in variable-sum games. In each generation, decisions are made regarding which strategies should be kept and form the basis for future search. The evolutionary heuristic maintains that fitter strategies are more likely to yield superior offspring. Yet, strategies that exhibit superior fitness only in transient population states may be less useful guideposts for search than attractor strategies, which at least have some stability properties with respect to the transient.

Thus arise questions regarding the time-scales on which selection and variation should operate. (These questions relate indirectly to Hammerstein's [87] *streetcar theory*, which seeks to reconcile models of Mendelian genetics with those of phenotypic adaptation.) In the conventional coevolutionary algorithm, selection and variation operate on the same time scale. Alternatively, we may delay the application of variation operators until selection alone takes the population past the transient and brings it to a state where fitness values are "believable"; we can then select parents for reproduction based on more accurate assessments of their dynamical stability. Essentially, this approach elongates the process of fitness evaluation through time, rather than use instantaneous estimates. Unfortunately, this approach also will likely eliminate too much genetic diversity for search to be effective. The population is essentially being asked to perform two contradictory tasks: Indicate the solution and maintain genetic diversity. We revisit this theme in Chapters 5 and 8. As an antidote, we can imagine using some external black-box to provide believable fitness values, allowing the population to concentrate on search; an instance of such a black-box could be a secondary population to which we apply only selection.

$$G = \begin{array}{r|rrr} & \mathrm{H} & \mathrm{D} & \mathrm{R} \\ \hline \text{Hawk} & -1 & 2 & -1 \\ \text{Dove} & 0 & 1 & 0.9 \\ \text{Retaliator} & -1 & 1.1 & 1 \end{array} \tag{3.2}$$

3.7.3 Random Variable-Sum Games

Here we explore the extent to which the above frequency-dependent effect can frustrate our ability to identify an attractor, or solution. Figures 3.10 through 3.13 present results from three different random symmetric variable-sum games. Each game has 50 pure strategies and each payoff of each game is sampled from a uniform distribution $\mathrm{U}(0,1)$. Each figure depicts a single orbit, using the standard EGT replicator, that begins at a random population state and converges to a monomorphic attractor. (Our method to discover these orbits is simply to evolve the population for 1000 time-steps and check the dynamic stability of the resulting population state if it is monomorphic. We also check that numerical underflow does not occur during the orbit.)

Figure 3.10 gives results of a sample orbit where all 50 pure strategies are present

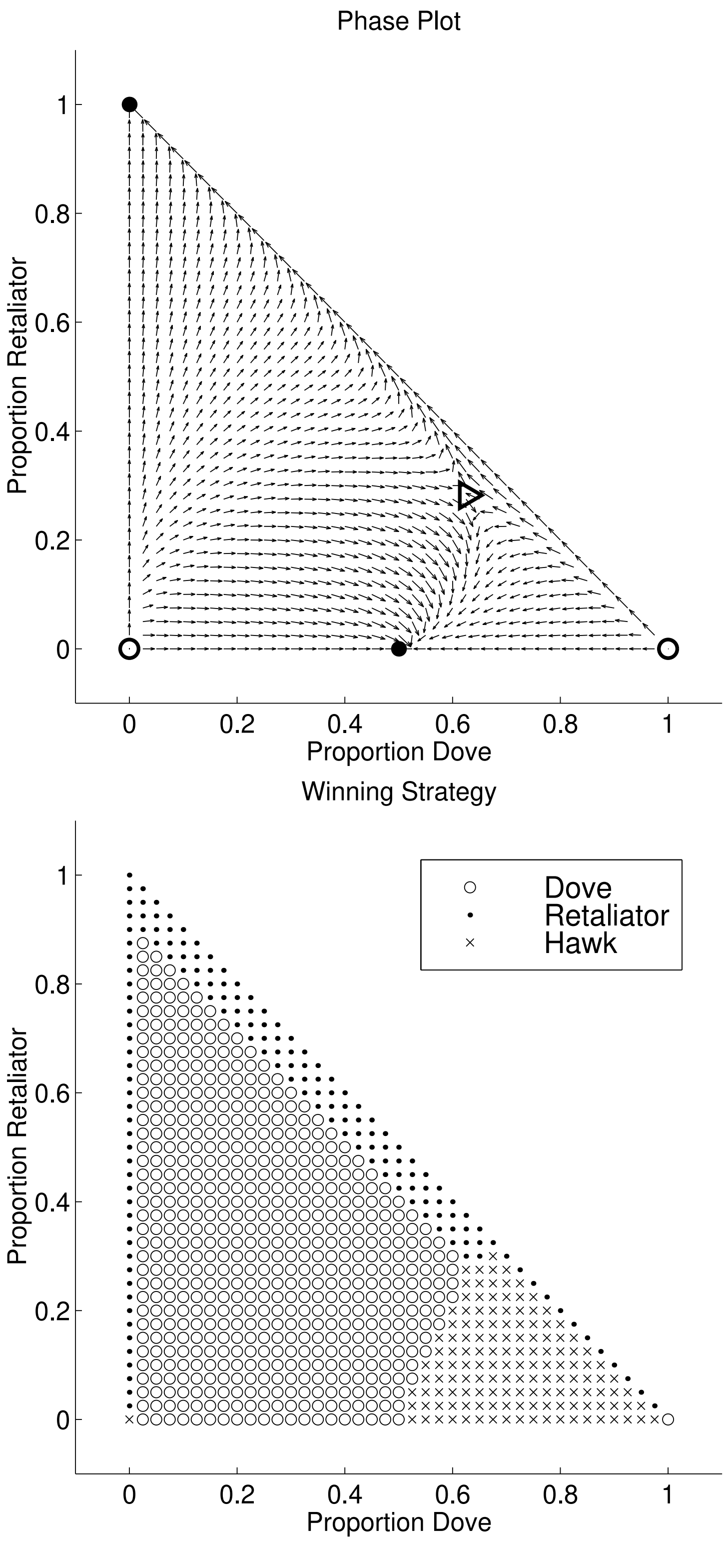


Figure 3.8: Hawk-Dove-Retaliator game. Top: Phase plot of Hawk-Dove-Retaliator game. Bottom: Phase space labeled according to which strategy has the highest fitness.

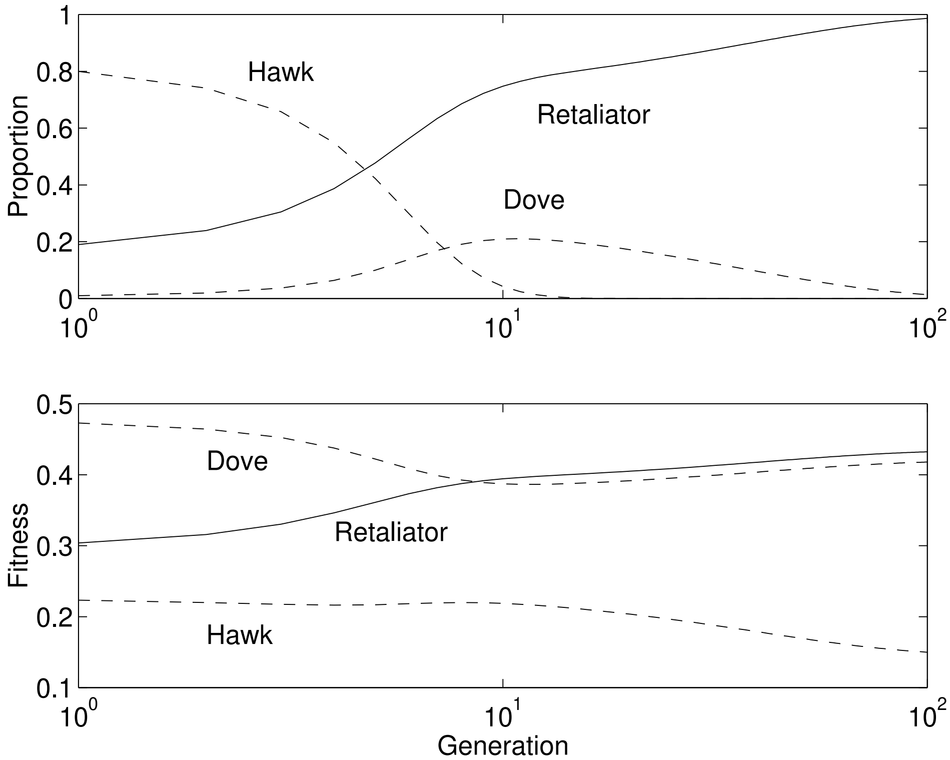


Figure 3.9: Sample orbit of Hawk-Dove-Retaliator game. Time axis is log-scaled to provide better detail of initial transient.

in the initial population state. Figure 3.10 (top left) shows the evolution of strategy proportions over 500 time-steps, with the attractor eventually taking-over the population (indicated by the bold line). In the bottom left of Figure 3.10, we show the fitness values of each strategy over time. The attractor strategy is indicated by the solid bold line, whereas the population's average fitness (frequency-weighted) is indicated by the dashed bold line. We see that the attractor is at or well below average fitness for the first 79 generations. The attractor strategy first becomes the most fit at generation 131, but does not permanently establish this position until generation 289.

Figure 3.10 (top right) shows how many of the 50 pure strategies out-score the attractor; we see that as many as 44 other strategies receive better fitness values. Figure 3.10 (bottom right) shows what proportion of the population out-scores the attractor. This graph reveals an even more dramatic story: over 99.95% of the population out-scores the attractor strategy at generation 61. Figures 3.12 and 3.13 give examples from two other random games.

Figure 3.11 shows the result of a different initial condition on the game used in Figure 3.10; here, we begin with all strategies present in equal proportions ($p_i = 0.02$ for all strategies i). The monomorphic attractor to which we converge is the same as before (strategy 49). In this orbit, the attractor's fitness reaches its nadir at approximately generation 125; this point corresponds to the peak in numbers of strategies that out-score the attractor (27) as well as the peak in the proportion of the population that out-scores the attractor (over 99.96%). The smallest proportion that the attractor has in the population during the orbit is $p_{\text{attractor}} = 5.5722 \times 10^{-5}$— far less than at the initial condition.

All of the results we present here use an infinite population. If we were to have a finite population, then quantization effects may be noticed. For example, a population of size 1000 can represent our initial condition, but not all the subsequent population states our orbit visits on the way to convergence. Thus, our initial condition presents a situation where we have our attractor strategy but will be unable to recognize it—the attractor strategy will be driven out of the population before fitness values reveal it to be our solution. Instead, the finite-population system will converge to some saddle-point. Thus, the size of the attractor's basin is reduced according to the resolution supported by the population size.

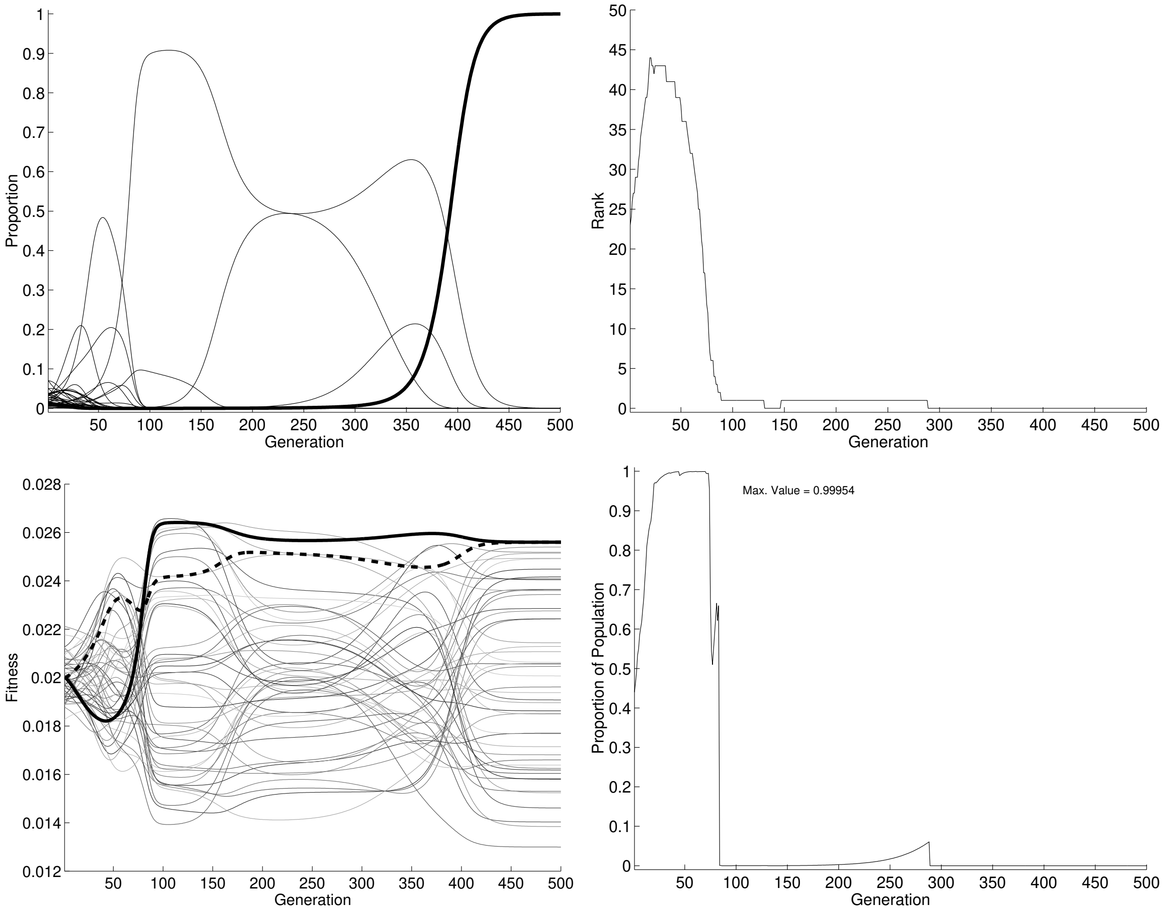


Figure 3.10: Sample orbit of random variable-sum game with 50 pure strategies. Top Left: Strategy proportions over time. Top Right: Fitness rank of monomorphic attractor over time. Bottom Left: Strategy fitness over time. Bottom Right: Proportion of population with higher fitness than monomorphic attractor over time.

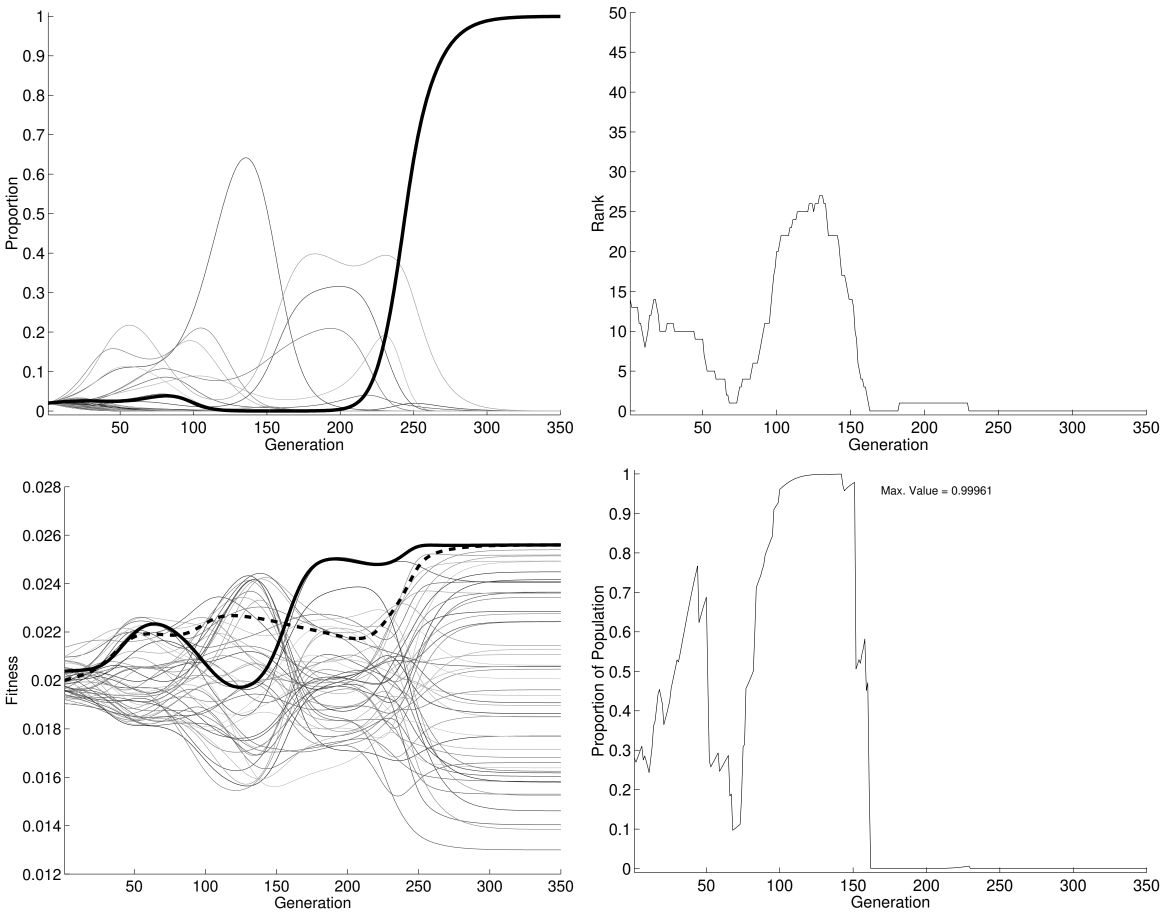


Figure 3.11: Orbit from initial condition where all strategies are present in equal proportions, using same game as that used in Figure 3.10. Top Left: Strategy proportions over time. Top Right: Fitness rank of monomorphic attractor over time. Bottom Left: Strategy fitness over time. Bottom Right: Proportion of population with higher fitness than monomorphic attractor over time.

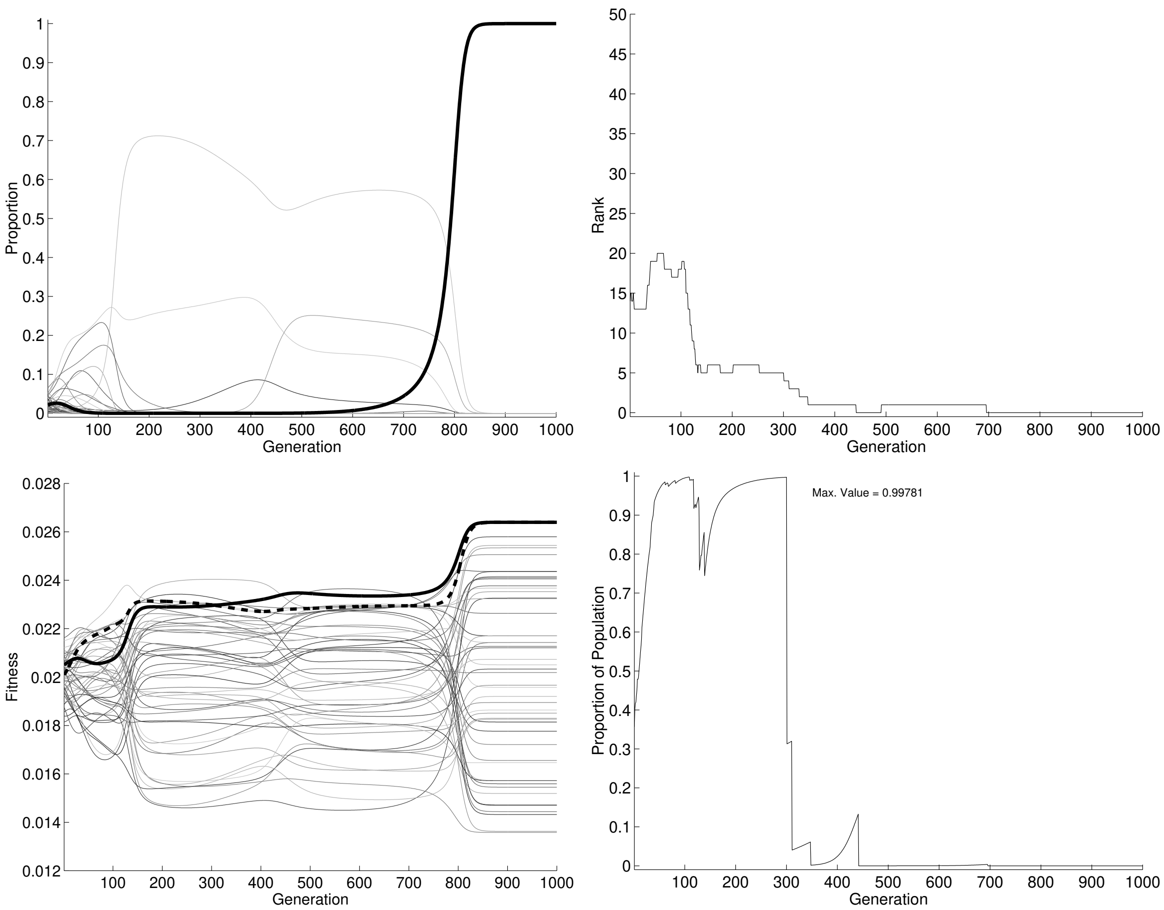


Figure 3.12: Sample orbit of random variable-sum game with 50 pure strategies. Top Left: Strategy proportions over time. Top Right: Fitness rank of monomorphic attractor over time. Bottom Left: Strategy fitness over time. Bottom Right: Proportion of population with higher fitness than monomorphic attractor over time.

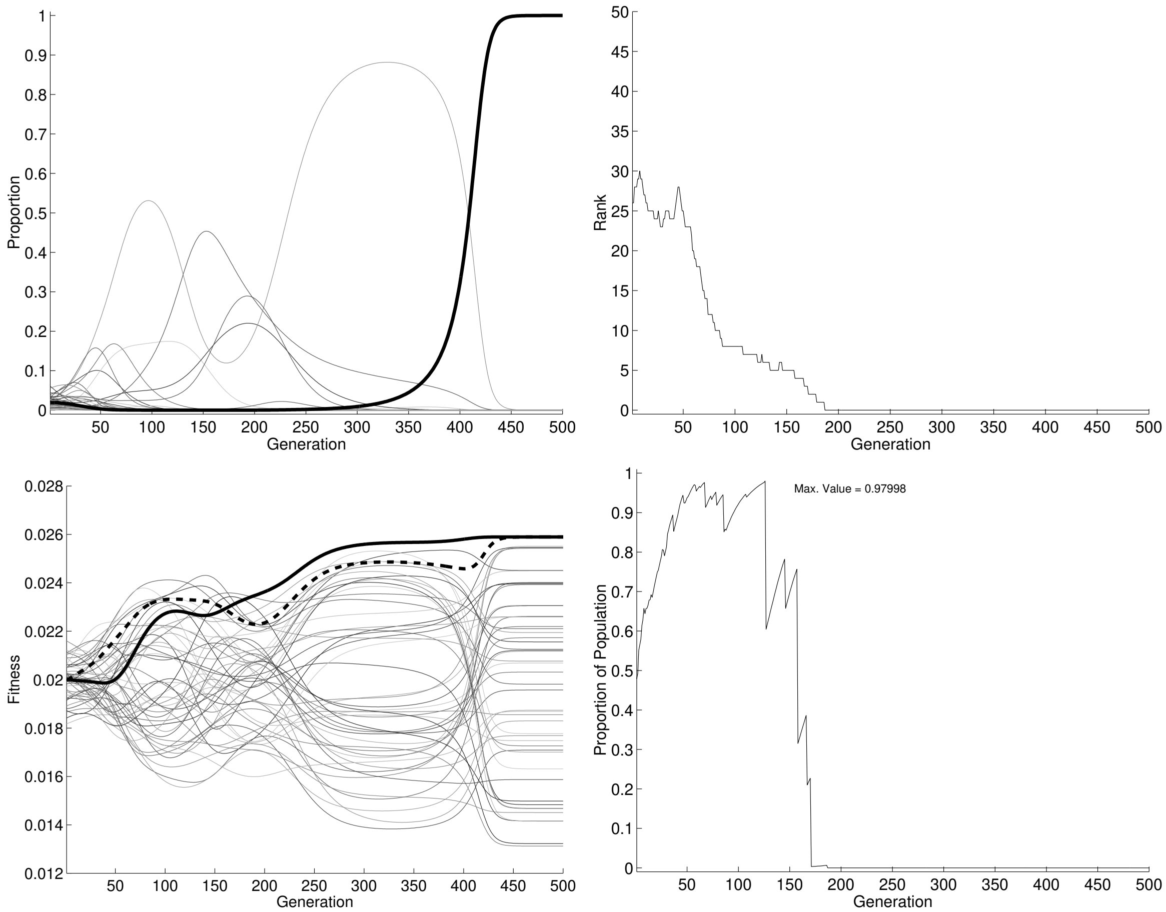


Figure 3.13: Sample orbit of random variable-sum game with 50 pure strategies. Top Left: Strategy proportions over time. Top Right: Fitness rank of monomorphic attractor over time. Bottom Left: Strategy fitness over time. Bottom Right: Proportion of population with higher fitness than monomorphic attractor over time.

3.8 Diversity Maintenance and Teaching

A general antidote to all of the common pathologies we discuss above is the maintenance of genetic and phenotypic diversity. The approaches to maintain diversity that we examine above include methods to slow genetic convergence by halting evolution in a population (e.g., Paredis [162] and Olsson [153]), spatial populations (e.g., Hillis [91] and Pagie and Hogeweg [156]), the use of multiple, reproductively isolated populations (e.g., Bullock [31], and Hornby and Mirtich [96]), assortative mating (a method to achieve speciation, e.g., Todd and Miller [193]), and various types of fitness sharing (e.g., Rosin and Belew [175] and Juillé and Pollack [103, 105]). (In Chapter 5, we show that fitness sharing methods distort polymorphic Nash equilibria.)

The problem of providing gradient for coevolutionary search—what we describe as our secondary search effort—has been addressed by many. Again, the maintenance of phenotypic diversity figures prominently in efforts to improve gradient. Indeed, the need for diversity of experience is stated even in non-evolutionary approaches to game learning, for example in Epstein [55], who argues that a knowledge-based approach to game learning provides systematic exposure to different aspects of a game. Rosin and Belew's heuristics to approximate *teaching sets* include phenotypic-diversity schemes such as competitive fitness sharing and the phantom parasite. Juillé and Pollack [106] show that avoidance of tests that cause learners to perform no better than random is effective in a density classification task. The heuristic we present in Chapter 7 seeks tests that distinguish the performance of learners in the population. While very different in approach, our method shares some affinity with the notions of teaching sets and sequences; the connection is particularly apparent in the follow-up work of Bucci and Pollack [30] and de Jong and Pollack [45].

Chapter 4

Solution Concepts and Selection Methods

4.1 Introduction

In Chapter 1, we briefly discussed evolutionary game theory [134] and pointed out that the replicator equations used in EGT are equivalent to what is known in evolutionary computation as *fitness-proportionate selection*. While the properties of this standard replication scheme are well-studied, alternative selection methods (replicators) used in evolutionary algorithms have escaped careful game-theoretic scrutiny. This chapter investigates the properties of four common EA selection methods from a game-theoretic and dynamical-systems perspective: truncation, (μ, λ)-ES, linear ranking, and Boltzmann selection. The use of these various selection methods in non-coevolutionary settings are reviewed in [86, 88, 141]; they are designed to provide different ways to manage genetic diversity and rates of convergence. But, how do these methods compare to canonical fitness-proportionate selection when used in a coevolutionary settings? Do they exhibit similar dynamics and promote the same fixed-points and attractors? We will use the simple infinite-population model of evolutionary game theory to answer these questions.

In evolutionary game theory, every attractor of the standard (i.e., fitness-proportionate) replicator is a Nash equilibrium [93]. Rather than maintain the dynamics and equilibria of EGT's standard replicator, the alternative selection methods we test yield cyclic dynamics, non-Nash attractors, and even chaos. Only Boltzmann selection (and only at low selection pressures) is faithful to EGT equilibria. We conclude that certain selection methods, while they may be effective in ordinary evolutionary algorithms, are likely to be inappropriate for coevolution. Specifically, the selection methods we test can fail to obtain polymorphic Nash equilibria in variable-sum games.

Thus, we demonstrate how alternative selection methods distort the solution concepts (in this case, Nash equilibrium) that we may otherwise believe to be operating. In so doing, we reveal a virtually unknown class of coevolutionary pathology where the solution concept for the primary search problem is improperly implemented. The particular manifestation of this pathology that we encounter here indicates that certain selection methods used in coevolutionary algorithms prevent the attainment of game-theoretically justifiable results.

We begin with a brief introduction to evolutionary game theory and the *Hawk-Dove* game. The next two sections outline our dynamical systems approach, where we first look at the dynamical features of the Hawk-Dove game, and then consider the role of the selection method (the replicator). We then consider each of the four selection methods, in turn. Discussions of how our results generalize to larger games, and what the results mean for coevolutionary algorithms follow. We then review a general methodology used in dynamical systems to determine the stability characteristics of an arbitrary differentiable replicator function. The work we report in this chapter is first published in Ficici, Melnik, and Pollack [58].

4.2 Evolutionary Game Theory

A central achievement of evolutionary game theory was the introduction of a method by which agents come to play "optimal" strategies in the absence of agent rationality [134]. Through a process of Darwinian selection, a population of agents can converge to an *evolutionary stable strategy* (ESS), which is a Nash equilibrium with an additional "stability" criterion. Despite the name, Maynard-Smith's definition of evolutionary stability is actually a static concept, and since its introduction many other solution concepts have been proposed [126], including those that are more properly rooted in dynamical systems theory [176]. Nevertheless, for the game studied in this chapter (the Hawk-Dove game), the ESS corresponds to a dynamical attractor [93, 188].

4.2.1 Assumptions and Operation

A symmetric two-player game consisting of m "pure" strategies is described by an m by m payoff matrix $\mathbf{G}$. If Player 1 plays strategy i and Player 2 plays strategy j, then their payoffs are given by matrix entries $\mathbf{G}_{i,j}$ and $\mathbf{G}_{j,i}$, respectively. We assume a single infinitely large population of agents, where each agent plays some pure strategy. The state of a population is represented by column vector $\mathbf{p}$, of length m, where element $\mathbf{p}_i$ represents the proportion with which game strategy i appears in the population of agents. Thus, each element $\mathbf{p}_i$ has a value between zero and one (inclusive), and the elements must sum to one.

We assume *complete mixing*: All possible pair-wise interactions between agents take place at each time step (i.e., generation). We define the *cumulative payoff* received by an agent playing strategy i to be the sum of 1) payoffs obtained upon interacting with all other agents (computed by the inner product of row i of the payoff matrix and the proportion vector $\mathbf{p}$), and 2) a constant baseline fitness w_0 (derived in Equation 4.2) endowed to each agent before interaction takes place. The vector $\mathbf{w}$ gives the cumulative payoffs for all strategies and is calculated by matrix multiplication and vector addition, as shown in Equation 4.1; $\mathbf{w}_i$ is the cumulative payoff of strategy i, and hence all agents that play strategy i. (Note: $\mathbf{w_0}$ is either a column vector or matrix, as appropriate, where all elements are w_0.)

Given the population state and cumulative payoff vectors at time t, $\mathbf{p}^t$ and $\mathbf{w}^t$, we compute the new state of our evolving population, $\mathbf{p}^{t+1}$, with Equation 4.3. This difference equation is known as a *replicator*, and causes the number of agents playing each strategy to increase (replicate) in proportion to cumulative payoff. We then re-normalize strategy proportions such that they again sum to one. Note that strategies absent from the initial population cannot appear at later time steps, since we are describing a selection-only system—no variational operators (i.e., mutation or recombination) exist to introduce strategies not already present.

$$\mathbf{w} = \mathbf{G}\mathbf{p} + \mathbf{w_0} \tag{4.1}$$

$$w_0 = |\ \min(\min(\mathbf{G}), 0)\ | + k, k \geq 0 \tag{4.2}$$

$$\mathbf{p}^{t+1} = \text{normalize}(\text{diag}(\mathbf{p}^t)\mathbf{w}^t) \tag{4.3}$$

Agent *fitness* is defined to be proportional to reproductive success. Because an agent can have no fewer than zero offspring, an agent's fitness value f must be $f \geq 0$. Since the standard replicator used in EGT specifies that agents reproduce offspring in proportion to cumulative payoff, we may interpret cumulative payoff to be synonymous with fitness. Thus, we define w_0 to be large enough to ensure that fitness values are non-negative, even if complete mixing leaves an agent with a negative net payoff. As we examine alternatives to the standard replicator equation, we will find that the relationship between cumulative payoff and fitness loses its linearity.

Finally, as shown in Equation 4.4, the baseline fitness w_0 can instead be added to each element in the payoff matrix $\mathbf{G}$ before multiplication with vector $\mathbf{p}$ to achieve the same fitness values as in Equation 4.1 (recall that the elements of $\mathbf{p}$ sum to one). Therefore, given a game $\mathbf{G}$ (with payoffs $a, b, \ldots$) and a baseline fitness w_0, we can define a new game $\mathbf{G}'$ (with payoffs $A, B, \ldots$), as shown in Equation 4.5.

$$\mathbf{w} = (\mathbf{G} + \mathbf{w_0})\mathbf{p} = \mathbf{G}\mathbf{p} + \mathbf{w_0}\mathbf{p} = \mathbf{G}\mathbf{p} + \mathbf{w_0} \tag{4.4}$$

$$\mathbf{G}' = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \mathbf{G} + \mathbf{w_0} = \begin{pmatrix} a + w_0 & b + w_0 \\ c + w_0 & d + w_0 \end{pmatrix} \tag{4.5}$$

4.2.2 Nash Equilibrium

In a symmetric two-player game (e.g., the Hawk-Dove game defined below), a strategy s^* creates a Nash equilibrium iff it is its own *best reply* [145, 83]:

$$\forall s \in \mathcal{S} : \mathrm{E}(s, s^*) \leq \mathrm{E}(s^*, s^*) \tag{4.6}$$

where $\mathcal{S}$ is the set of possible pure and mixed strategies, and $\mathrm{E}(s_1, s_2)$ is the expected payoff given to strategy s_1 after interaction with strategy s_2.

That is, if Player 2 chooses to play s^*, then the maximal expected payoff that Player 1 can achieve is acquired by also playing strategy s^*, and *vice versa*. The strategy s^* may be pure or "mixed" (a probability distribution over the set of pure strategies). If s^* is a unique best reply, that is, if there exists no other strategy that does as well when interacting with s^* as s^* itself, then the Nash equilibrium is *strict*.

Otherwise, it is *weak*. All finite games have at least one mixed Nash equilibrium (pure Nash equilibria being degenerate mixtures).

4.3 Standard Replicator

We are interested in how the dynamics and equilibria of the standard discrete-time replicator (Equation 4.3) change when the calculation of strategy fitness is modified to implement various selection methods (and forms of fitness sharing in Chapter 5). In this section, we note the replicator's normative behavior, that is, in the absence of modifications.

The results we present throughout this chapter are illustrated with the Hawk-Dove game, the payoff matrix for which is shown in Equation 4.7. This is a symmetric, variable-sum game for two players that has two pure strategies (Hawk and Dove). Such games always cause the standard replicator to converge to a point attractor that is an ESS, and hence a Nash equilibrium [134]. Point attractors are either *monomorphic* or *polymorphic*. A monomorphic attractor is a population state comprised of a single pure strategy (where all the other strategies are driven to "extinction," that is, an infinitely small proportion of the infinitely large population). A polymorphic attractor is a population state comprised of more than one pure strategy, where the surviving strategies have reached a *fitness equilibrium* at some particular strategy proportion. As we will see below, the Hawk-Dove game has a polymorphic attractor.

$$G = \begin{array}{c|cc} & \mathrm{H} & \mathrm{D} \\ \hline \mathrm{H} & -25 & 50 \\ \mathrm{D} & 0 & 15 \end{array} \tag{4.7}$$

4.3.1 Equilibrium

Given an arbitrary symmetric game of two pure strategies x and y, the population state vector is composed of two scalars $\mathbf{p} = [p_x \;\; p_y]^{\mathrm{T}}$, as is the fitness vector $\mathbf{w} = [w_x \;\; w_y]^{\mathrm{T}}$. We can find the proportion at which the strategies reach fitness equilibrium (assuming that such a proportion exists) by setting $w_x = w_y$ and solving for p_x; the solution is given by Equation 4.8. If the game in question does not have a polymorphic equilibrium, then Equation 4.8 will give a value outside of the interval $(0, 1)$ or have a zero denominator. Because a two-strategy game yields a one-dimensional system, the value of p_y can be inferred: $p_y = 1 - p_x$.

The important feature of Equation 4.8 is that the fitness equilibrium ratio between p_x and p_y does not depend on the value of w_0. Indeed, the polymorphic equilibria of Equation 4.3 are independent of w_0 regardless of the number of strategies [176]. If this is the case, we may ask why we need w_0. Though the value of w_0 is unimportant at fitness equilibrium, it is important when the population state is away from equilibrium—recall that we need it to prevent negative fitness values. Thus, there exist uncountably many games with the same set of fitness equilibria.

The Hawk-Dove game has a polymorphic Nash equilibrium; the proportion of Hawks at this equilibrium is $p_H = (15-50)/(-25-50-0+15) = \frac{7}{12}$. For population states $p_H < \frac{7}{12}$, Equation 4.1 shows that $w_H > w_D$; for population states $p_H > \frac{7}{12}$, we find $w_H < w_D$. (Note that the Hawk-Dove game has an additional Nash equilibrium where one player plays Hawk and the other plays Dove; given such a configuration of strategy choices, neither player is incited to unilaterally change strategies. Nevertheless, we cannot represent this Nash equilibrium with a single evolving population, and so will not consider it in our present investigation. In contrast, a two-population framework can obtain this Nash equilibrium in the standard EGT framework.)

$$p_x = \frac{D-B}{A-B-C+D} = \frac{d+w_0-b-w_0}{a+w_0-b-w_0-c-w_0+d+w_0} =$$

$$\frac{d-b}{a-b-c+d} \tag{4.8}$$

4.3.2 Dynamics

Since two-strategy games give one-dimensional systems, we can depict the dynamics of the Hawk-Dove game with a *return map*; the x-axis represents the state of the population p_H (proportion of Hawks in the population) at time t and the y-axis represents p_H at time $t+1$. Figure 4.1 shows the return map of the Hawk-Dove game at two different values of w_0. Both curves intersect the diagonal line at the same three points; each of these intersections represents a *fixed point* of the dynamical system, where the population state does not change from one time-step to the next. The fixed-point in the middle of the graph is the polymorphic Nash equilibrium of the Hawk-Dove game, where $w_H = w_D$.

The slope of the curve at a fixed-point determines the fixed-point's *stability*. If the absolute value of the slope is less than one, then the fixed-point is stable. We see that both curves in Figure 4.1 meet this requirement where $p_H = \frac{7}{12}$, and so the polymorphism is an (indeed, the only) attractor of the dynamical system. Though w_0 affects neither the stability nor the location of the polymorphism, we see that larger values of w_0 bring the curve closer to the diagonal, which results in a dynamical system that moves more slowly to the attractor.

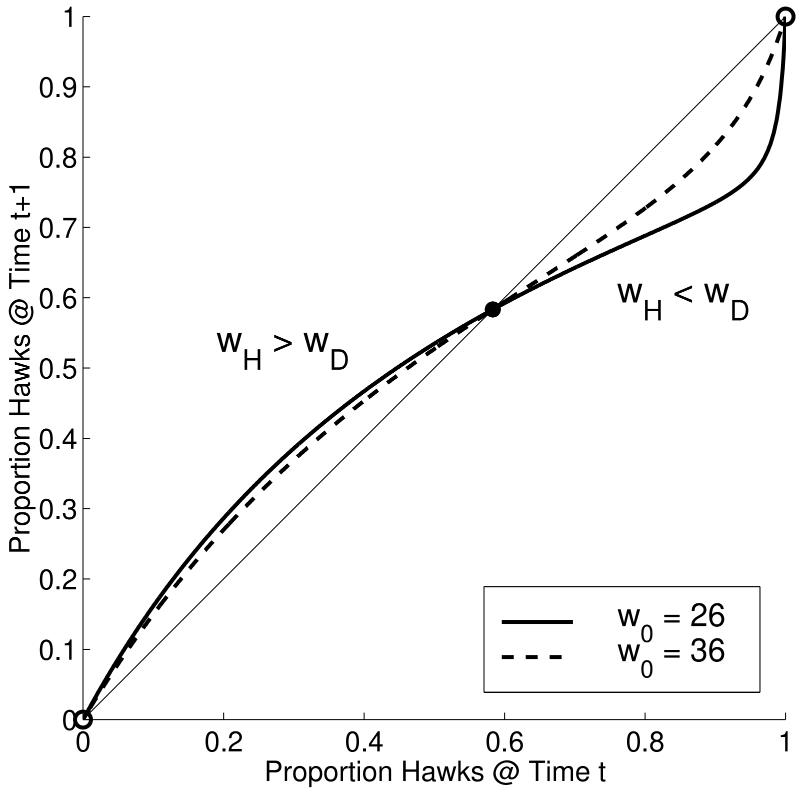


Figure 4.1: Return map of Hawk-Dove game using different values of w_0.

4.4 Truncation Selection

Truncation selection is frequently used in a type of evolutionary computation known as *evolutionary programming* [71]. Truncation operates by first sorting the population according to fitness and then removing the worst k percent of the population and replacing them with variations of the best k percent; the value k expresses a selection pressure. For example, given a population of size 200 and selection pressure of $k = 25$, we will remove the worst 50 individuals and replace them with variations of the best 50 individuals. This approach requires a selection pressure range of $0 \leq k \leq 50$, where higher values of k give higher selection pressure. Since evolutionary game theory assumes selection only, we exclude variation operators and simply replace the worst $k\%$ with exact copies of the best $k\%$.

We can easily implement truncation selection for the infinite population that evolutionary game theory assumes. All agents that play the same strategy receive the same cumulative payoff; therefore, we need only sort the strategies by their cumulative payoffs, given the state of the population, and note the proportions with which each strategy appears in the population.

More precisely, given the population state $\mathbf{p}$, strategy i exists with proportion $\mathbf{p}_i$ and receives a cumulative payoff of $\mathbf{w}_i$. We sort the strategies of the game according to cumulative payoffs and obtain a new vector $\mathbf{q}$. The element $\mathbf{q}_{s_j}$ is the proportion with which strategy s_j (the strategy with sorted rank j) appears in the population; the index j ranges between 1 and m, and s_1 signifies the strategy with the highest cumulative payoff.

We construct a vector $\mathbf{r}$ to indicate the proportion of each strategy we are to remove from the population; with $\mathbf{r}$ we represent the worst k percent of the population. We construct a similar vector $\mathbf{b}$ to indicate the proportion of each strategy we are to add to the population; we use $\mathbf{b}$ to represent the best k percent of the population. The new strategy proportions are $\mathbf{q}' = \mathbf{q} - \mathbf{r} + \mathbf{b}$.

Figure 4.2 illustrates a game of three strategies where each strategy initially occupies 1/3 of the population; truncation is applied with maximum selection pressure ($k = 50$). The worst half of the population contains all of the agents that play strategy s_3 and one half of those that play s_2; the best half contains all of the agents that play s_1 and the other half of those who play s_2 (note that we do not make special accommodations for ties). After truncation, the new proportions $\mathbf{q}'$ indicate that s_1 and s_2 compose 2/3 and 1/3 of the population, respectively; strategy s_3 has been eliminated.

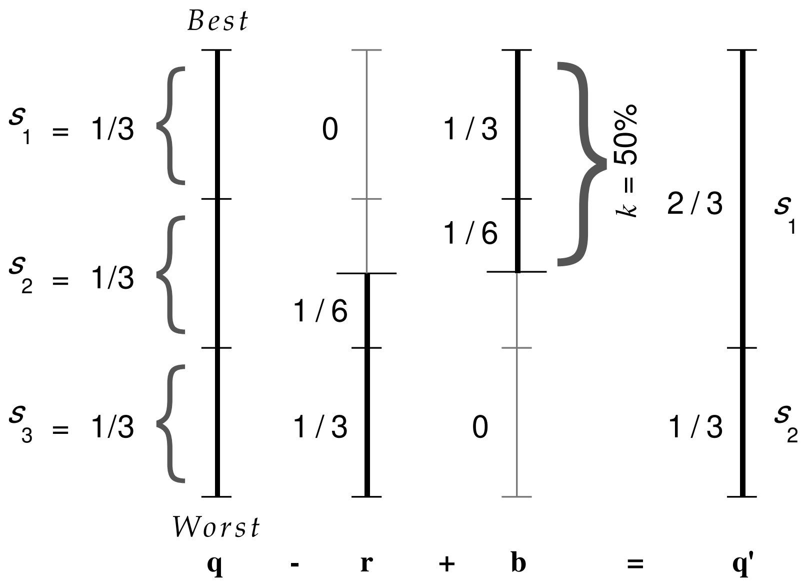


Figure 4.2: Truncation selection on an infinite population.

What does truncation selection do at a fixed point that involves more than one strategy, such as the polymorphic Nash equilibrium of the Hawk-Dove game? Recall that, at a fixed point, all strategies present in the population receive the same fitness. Since no special accommodation is made for fitness equilibria (ties), the result of sorting, and therefore truncation selection, is ill-defined. Truncation selection is unable to maintain arbitrary fixed points unless special precautions are taken to deal with fitness equilibria.

When we use truncation selection instead of the standard replicator, we observe three regimes of behavior as selection pressure is varied. For higher selection pressures, in the range $42\% \leq k \leq 50\%$, truncation selection causes the system to converge onto a new attractor of all Hawks. Figure 4.3 (top) shows the return map produced by truncation when $k = 50$. We find a large discontinuity precisely at the population state where the attractive Nash equilibrium should be. The example orbit eventually converges to the state of all Hawks.

Equation 4.1 gives both strategies the same cumulative payoff at the Nash equilibrium proportion of $p_H = \frac{7}{12}$; below this proportion, the Hawks outscore the Doves and above this proportion the Doves outscore the Hawks. By virtue of this payoff relationship, the standard replicator enables a simple feedback mechanism that gives the Nash its stability. But, truncation selection breaks this feedback mechanism. As an illustration, let us consider a population state where the proportion of Hawks is $\frac{1}{2} \leq p_H < \frac{7}{12}$. Because p_H is below the Nash equilibrium proportion, the Hawks receive higher cumulative payoff than the Doves. But, the Hawks also comprise more than half of the population. Thus, the best 50% of agents in the population can only be playing the Hawk strategy, and the next generation will be 100% Hawks. The orbit of most initial conditions $0 < p_H^0 < 1$ will eventually fall into the critical interval $\frac{1}{2} \leq p_H < \frac{7}{12}$ and converge to the non-Nash attractor of all Hawks. Nevertheless, cyclic behavior is possible as well; for example unstable period-two and period-three cycles exist. If the population state begins at the Nash equilibrium, then we must account for fitness ties and maintain the proper ratio of Hawks and Doves in the **r** and **b** vectors; otherwise, the fixed-point will be lost.

For moderate selection pressures $31\% \leq k \leq 41\%$, truncation selection produces chaos. Figure 4.3 (middle) shows one such chaotic orbit where $k = 36$. The *Liapunov exponent* of a dynamical system is a measure of sensitivity to initial conditions, and is an indicator of chaotic behavior if it is greater than zero. To calculate the Liapunov exponent, we normally measure the derivative of the map at each time step of an orbit. But, the truncation map has a discontinuity, and so is not differentiable. The map is piece-wise linear, however, and the lack of smoothness is negligible. Therefore, we use the slope of the line segment. This yields a Liapunov exponent of $\lambda = 0.69$.

For lower selection pressures, $0 < k \leq 30\%$, truncation selection gives neutrally stable cycles. Figure 4.3 (bottom) provides a sample orbit where $k = 15$. The discontinuity at the Nash proportion remains; for $p_H < \frac{7}{12}$ the map is above the diagonal, and for $p_H > \frac{7}{12}$ the map is below. Therefore, all of the cycles go around the Nash proportion. The exact location of the cycle is determined by where the orbit first enters the cycle-inducing region of the map. As selection pressure decreases, the limit cycles exhibit tighter orbits around 7/12.

4.5 (μ, λ)-ES Selection

The (μ, λ)-ES selection method is used in the branch of evolutionary computing known as *evolution strategies* [17]. Given a population of λ offspring, the best μ offspring are chosen to parent the next generation. Normally, variational operators are applied during reproduction, but we omit variation here, as we did with truncation selection.

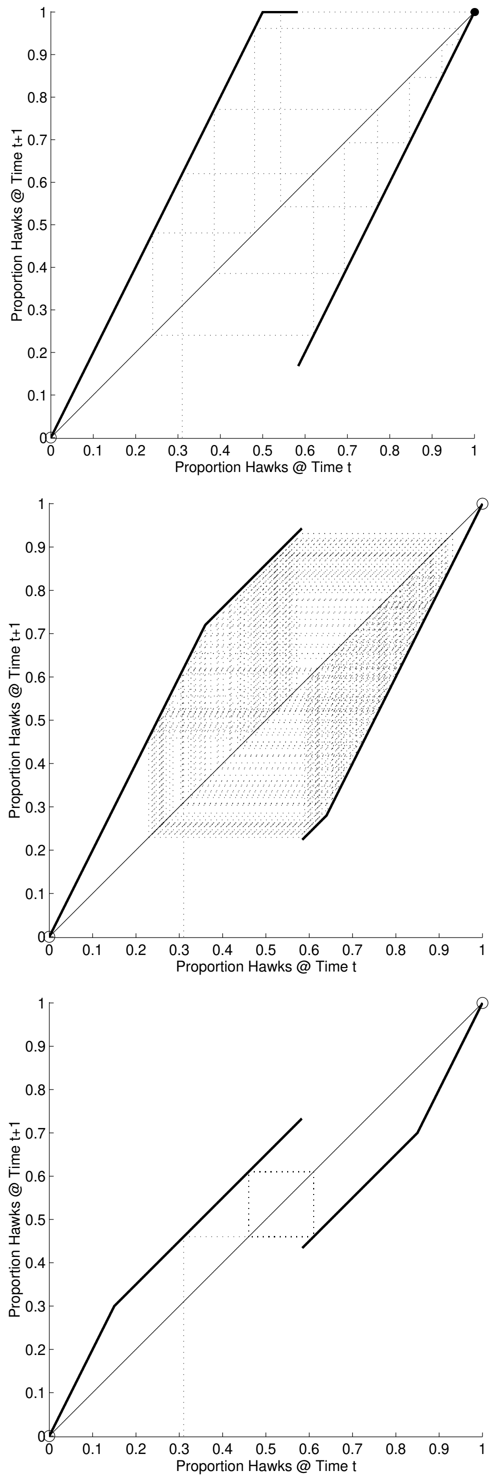


Figure 4.3: Return maps of truncation selection method with selection pressure at 50% (top), 36% (middle), and 15% (bottom).

The (μ, λ)-ES selection method is similar to truncation selection, but more drastic. In (μ, λ)-ES selection, the best $\frac{\mu}{\lambda}\%$ expands to replace the entire population rather than just the worst $\frac{\mu}{\lambda}\%$.

The implementation of (μ, λ)-ES selection for infinite populations is identical to that of truncation selection, except that now vector w is discarded and vector b is normalized to create the new population. The fraction μ/λ determines the selection pressure—the actual values of μ and λ are unimportant for an infinite population. Thus, for (μ, λ)-ES selection, the selection pressure can be in the range $0 < \frac{\mu}{\lambda} \leq 1.0$, where lower values indicate higher selection pressure.

For the Hawk-Dove game, (μ, λ)-ES selection also has three regimes of behavior, none of which are able to maintain arbitrary fixed points and all of which include some initial conditions that lead to cycles. For the range $.59 \leq \frac{\mu}{\lambda} < 1.0$, the behavior is usually chaotic. Figure 4.4 (top) shows an example orbit where $\frac{\mu}{\lambda} = 0.6$. The measured Liapunov exponent is 0.52. In the range $.42 \leq \frac{\mu}{\lambda} \leq .58$, the system converges to all Hawks in a manner very similar to truncation selection. The range $0 < \frac{\mu}{\lambda} \leq .41$ introduces the additional possibility of converging onto all Doves. This is shown in Figure 4.4 (bottom), where $\frac{\mu}{\lambda} = 0.3$. Because the map of (μ, λ)-ES selection is not differentiable, the stability properties of the all-Hawk and all-Dove fixed points are unusual—they are attractors, but they are not locally stable. The same is true for the all-Hawk attractor seen in truncation selection.

4.6 Linear Rank Selection

A common selection method used in genetic algorithms is *ranking* [85]; we sort agents according to score and then assign fitness values according to their rank, such that superior ranks yield higher fitness values. In *linear* ranking, fitness changes linearly

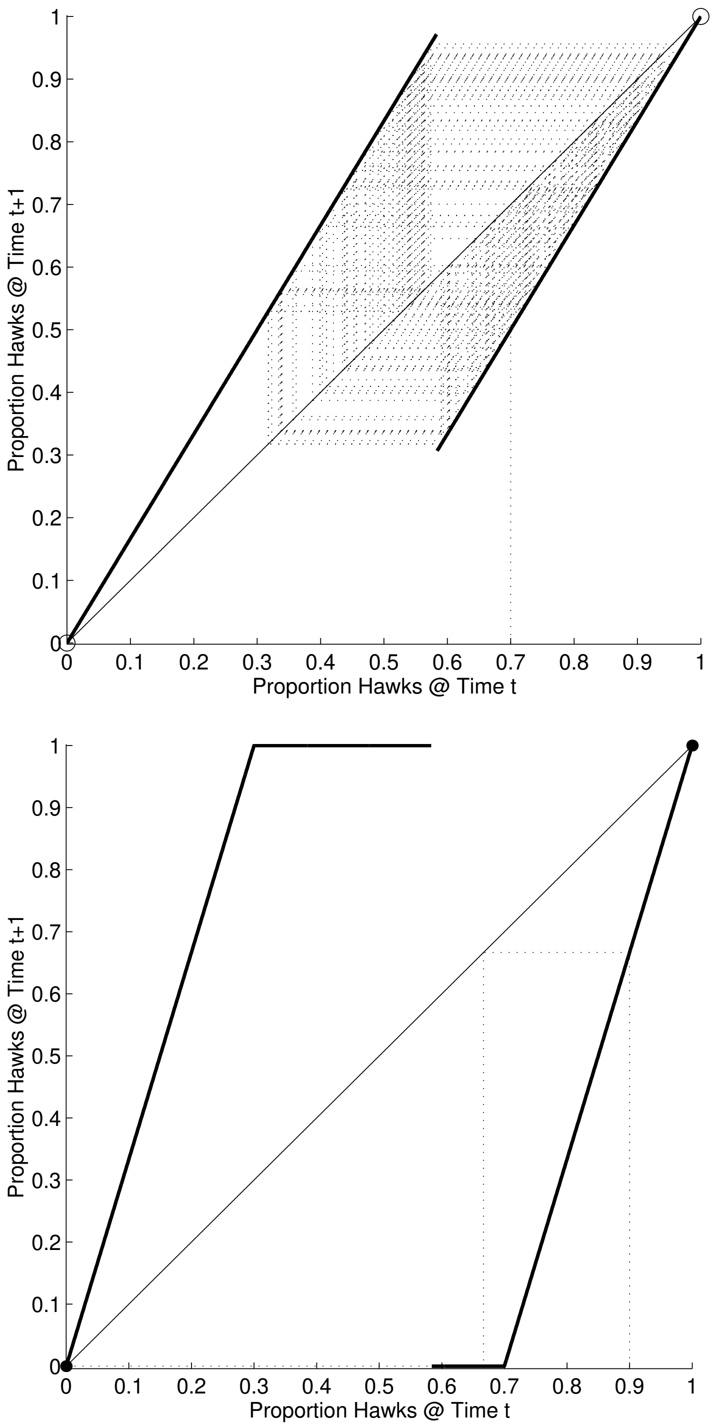


Figure 4.4: Map diagrams of (μ, λ)-ES selection with $\frac{\mu}{\lambda} = 0.6$ (top) and 0.3 (below).

with rank; agents are then selected in proportion to fitness. Ranking methods are used to reshape the population's fitness distribution. In the absence of ranking, differences between agents' fitnesses can become too small for the standard roulette wheel to resolve, given a finite population; superior genetic material may be lost or insufficiently exploited. Alternatively, a very large difference in fitness may lead to premature convergence to local optima. With ranking, however, excessively large fitness differences are attenuated while very small differences are expanded.

We implement linear ranking for our infinite population with ease; we simply rank the strategies' cumulative payoffs and then assign each agent a fitness value according to the rank of the strategy it is playing. In the case of the Hawk-Dove game, we have only two strategies; whichever of the two strategies obtains the higher cumulative payoff for the given population state provides its agents a fitness of two, while the other strategy provides a fitness of one. After normalization, the fitness values will either be $f_H = \frac{1}{3}, f_D = \frac{2}{3}$, or $f_H = \frac{2}{3}, f_D = \frac{1}{3}$; if special care is taken to handle ties (at Nash equilibrium), then we obtain $f_H = \frac{1}{2}, f_D = \frac{1}{2}$. Selection then proceeds in proportion to these fitness values.

Figure 4.5 shows the return map for linear rank selection. Again, at the population state that is the Nash equilibrium ($p_H = \frac{7}{12}$), we find a discontinuity instead of an attractor. Further, we find that linear ranking produces neutrally stable cycles around the Nash equilibrium proportion; this is the only behavior that linear ranking produces. Because the fitnesses, and hence rates of growth, of the two strategies are exactly inverted as the Nash proportion is crossed, a period two cycle must result.

Ranking maps all possible fitness proportions to a single proportion. As a result, rates of growth can never approach equality, which makes attractive polymorphic fixed points impossible, as is visually obvious in Figure 4.5. Another version of rank-based selection assigns fitness values that vary exponentially with rank. This method has the same properties as linear ranking that prevent convergence to polymorphisms.

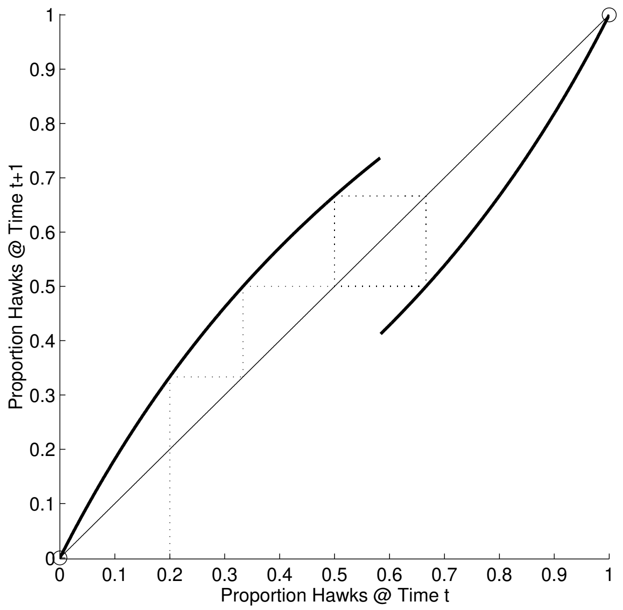


Figure 4.5: Return map of linear rank selection.

4.7 Boltzmann Selection

In *Boltzmann selection*, a method inspired by the technique of *simulated annealing*, selection pressure is slowly increased over evolutionary time to gradually focus search [141]. Given a fitness of f, Boltzmann selection assigns a new fitness, f', according to the differentiable function:

$$f' = e^{\beta f} \tag{4.9}$$

where $\beta > 0$ and higher values of β give higher selection pressure.

Agents are then selected in proportion to their new fitnesses, f'. In contrast to the selection methods seen above, this selection method can maintain arbitrary fixed points without modification. But, it too exhibits multiple regimes of behavior. For low selection pressures, Boltzmann selection can preserve the ESS attractor. Figure 4.6 (top) shows the map of Boltzmann selection in the Hawk-Dove game with $\beta = 0.05$; we see that the ESS is intact. If we increase the selection pressure to $\beta = 0.2$, as in Figure 4.6 (middle), a true limit cycle results and the ESS becomes an unstable fixed point. A higher pressure of $\beta = 0.5$ brings the system to the edge of chaos, seen in Figure 4.6 (bottom), yielding a small but positive Liapunov exponent. The analogy between low annealing temperature and high selection pressure is strained—too low a "temperature" actually destabilizes the system.

4.8 Large Games

What generalizations can we make from the results obtained with the two-strategy Hawk-Dove game? Because truncation, (μ, λ)-ES, and rank-selection utilize sorting, their dynamics are unable in principle to converge to a polymorphism, regardless of the number of strategies involved. Rather, these selection methods introduce discontinuities where attractors should be. Because sorting introduces discontinuities, selection methods that use sorting cannot converge onto polymorphic Nash equilibria. For Boltzmann selection, the destabilizing effect of high selection pressure is dampened as the number of strategies increases. The higher number of dimensions of large games tends to bring fitness values closer together. Thus, higher selection pressure is required to induce chaos in large games.

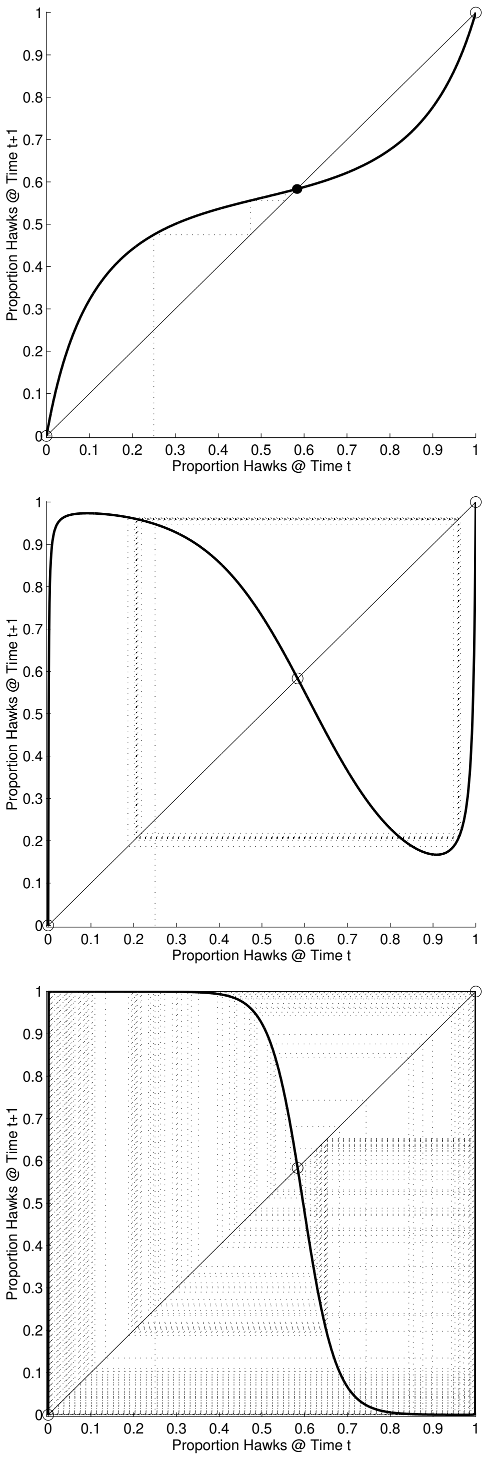


Figure 4.6: Map diagrams of Boltzmann selection method with $\beta = 0.05$ (top), 0.2 (middle), and 0.5 (bottom).

4.9 When does a map have a stable fixed-point?

We have analyzed the stability properties of four particular selection methods. In this section, we review a standard test that can be applied to any differentiable selection function to determine its stability properties [54]. This test is based on the Hartman-Grobman theorem, which allows us to treat a system as if it were linear in the vicinity of the fixed-point. By doing so, we can apply the simple stability tests of linear maps to the fixed points.

Let $p_{t+1} = M(p_t)$ be a map with a fixed point at p^{fix}. We first linearize the system by calculating its first derivative at p^{fix}. For a game of n strategies, we have an $m = n - 1$ dimensional map. If $n > 2$, then the map is a *multi-variable* function, and we need to calculate its *Jacobian Matrix*:

$$\partial M(p) = \begin{pmatrix} \frac{\partial M_{p^1}}{\partial p^1} & \cdots & \frac{\partial M_{p^1}}{\partial p^m} \\ \vdots & \ddots & \vdots \\ \frac{\partial M_{p^m}}{\partial p^1} & \cdots & \frac{\partial M_{p^m}}{\partial p^m} \end{pmatrix} \tag{4.10}$$

where $\frac{\partial M_{p^i}}{\partial p^j}$ is the partial derivative of function variable p^i with respect to variable p^j.

The test for convergence is to check whether the eigenvalues of $\partial M(p^{\text{fix}})$ are within the interior of the unit-circle. That is, we check each eigenvalue, λ, of the Jacobian to see if $\|\lambda\| < 1$ (where λ is potentially a complex number). If all eigenvalues fall within the unit circle, then the fixed point is stable. If one of the eigenvalues falls outside, then the fixed point is unstable.

To gain some intuition about this test, consider a one-dimensional map, such as the one we saw for fitness-proportionate selection in Figure 4.1. This map contains a stable fixed-point, such that if we iterate the map from a point near the fixed-point, we will converge onto the fixed-point. Why do points in the neighborhood of the fixed-point converge onto it? Take a point at an offset from the fixed point, $p_0 = p^{\text{fix}} + \mu$, close enough to the fixed point that we can treat the map as linear. The linearization of the map in the region allows us to approximate the map as $M(p) \approx p^{\text{fix}} + \lambda(p - p^{\text{fix}})$, where λ corresponds to the slope of the map at the fixed point. An iteration of the map has the effect of multiplying the offset by the slope, $p_1 = M(p_0) \approx p^{\text{fix}} + \lambda(p^{\text{fix}} + \mu - p^{\text{fix}}) = p^{\text{fix}} + \lambda\mu$. If $|\lambda|$ is less than 1, then application of the map will cause p_1 to be closer to p^{fix} than p_0. Therefore, multiple iterations of the map cause the offset to be multiplied repeatedly by the slope, $p_n = p^{\text{fix}} + \lambda^n\mu$, and (if $|\lambda| < 1$) bring it closer and closer to the fixed-point.

Thus, the one-dimensional map test is whether the absolute value of the derivative (slope) at the fixed point is less than one. The multi-dimensional test is based on the same convergence properties as the one-dimensional case. In a way, taking the eigenvalues of the Jacobian matrix is equivalent to breaking the multi-dimensional system down into constituent one-dimensional systems, where each eigenvalue represents the rate of change (derivative) of each one-dimensional degree of freedom of the multi-dimensional system. These eigenvalues can be complex numbers, rather than reals. Nevertheless, the test stays the same in the sense that we still test whether multiplying an offset by an eigenvalue will shrink the offset. Thus, in the multi-dimensional case, the test becomes whether all the eigenvalues' magnitudes are less than one, i.e., in the unit circle.

4.10 Discussion

The importance of understanding that certain mechanisms distort the expected evolutionary dynamics is appreciated with the following case study (which stimulated the work we report here). Since the strong assumptions made by evolutionary game theory are, of course, not to be found in the real world, D. Fogel and G. Fogel [73, 74, 75] investigate the effects of finite populations, noisy payoffs, and incomplete mixing upon the equilibria and dynamics that EGT predicts. They use the Hawk-Dove game in their simulations, but they also use truncation selection for several of their experiments. The results they report are consistent with those we discuss above, but they attribute their results to the factors they study, such as finite populations; they conclude that evolutionary game theory loses predictive power in the real world. We know from our experiments, however, that truncation violates the expected outcome even when the strong assumptions of evolutionary game theory are met.

As another case study, we can examine the body of research concerning the iterated prisoner's dilemma (IPD); the IPD is a variable-sum game, and a finite collection of IPD strategies can easily have a polymorphic Nash equilibrium. Meuleau and Lattaud [137] use (μ, λ)-ES as well as fitness-proportionate selection in their IPD experiments; they note dramatic differences in their results that are consistent with ours. Indeed, even the seminal evolutionary investigation of Axelrod [14] (what is probably the first instance of the conventional coevolutionary algorithm) uses a selection method that introduces discontinuities: Agents with cumulative scores within one standard deviation of the population average each receive one offspring; those one standard deviation above average receive two offspring, and those one standard deviation below average receive none. Clearly, this method performs a sorting operation that loses information about payoff convergence. This method is in contrast to the standard EGT framework Axelrod [13, p. 50–51] uses to analyze his earlier computer tournament results.

Thus, in light of our results, we believe that the many coevolutionary (and game-theoretic) investigations in the literature that use alternative selection methods may require a second look, especially where a single population is used and the domain under investigation is a variable-sum game.

4.11 Summary and Conclusion

With the Hawk-Dove game as a backdrop, we use evolutionary game theory to establish the normative behavior and outcome for a coevolutionary algorithm. The Hawk-Dove game has a single polymorphic Nash equilibrium that is an attractor of the standard replicator dynamic used in EGT. Against this norm we contrast the operation of a coevolutionary algorithm using four alternatives to the standard replicator equation; these methods are truncation, (μ, λ)-ES, Boltzmann, and linear rank selection.

When applied to variable-sum games with attractive polymorphic Nash, these methods alter selection pressures such that the system either fails to converge or converges to solutions that lack game-theoretic justification. Truncation, (μ, λ)-ES, and linear rank selection cannot attain Nash equilibria that involve more than one strategy. Instead, these methods exhibit cyclic behavior, chaos, or fixed points that are essentially unrelated to the values of the payoff matrix. Boltzmann selection, however, is able to retain the dynamics and equilibria seen in evolutionary game theory, provided that the selection pressure is not too high.

Our results indicate that selection methods cannot be moved whole-sale from evolutionary to *co*-evolutionary frameworks without careful consideration. Specifically, three of the four methods we consider in this chapter appear to be pathological in the context of single-population coevolution in variable-sum games. Nevertheless, a test from dynamical systems theory allows one to determine analytically the appropriateness for coevolution of arbitrary differentiable selection functions, without the need for empirical investigation. Indeed, any number of differentiable functions can be substituted for fitness proportional selection and provide variable selection pressure while maintaining “proper” operation.

Our future work will expand the results presented here in two directions. First, we will extend our investigation to a two-population framework, where the asymmetric Nash equilibrium arises (i.e., where one player plays Hawk and the other Dove). Some of our results are easily transferred to this situation. Truncation selection (at 50% selection pressure) provides an easy example; if both populations have proportions of Hawks $0.5 \leq p_H \leq \frac{7}{12}$ at time t, then both populations will converge to all-Hawks in the next time-step—clearly not the asymmetric Nash equilibrium. Second, we will extend our investigation to polymorphisms that involve more than two strategies. Such games are easily constructed and our preliminary results indicate that truncation selection becomes even more prone to chaotic behavior in these circumstances.

Chapter 5

Solution Concepts and Fitness Sharing Methods

5.1 Introduction

In this chapter, we use the methodology of the previous chapter to investigate *fitness sharing methods*. Fitness sharing mechanisms are designed to maintain a higher level of genetic diversity than ordinary fitness-proportionate selection can maintain alone. We examine two mechanisms in particular: Similarity-based fitness sharing [85] and competitive fitness sharing [172]. We find that both of these methods prevent convergence onto polymorphic Nash equilibria in variable-sum games. Our investigation uses the Hawk-Dove game to illustrate the effects of fitness sharing. We also discuss two methods by Rosin and Belew [172], phantom parasite and shared sampling, that are intended for use with competitive fitness sharing. The work presented in this chapter is also found in Ficici and Pollack [63].

5.2 Standard Replicator

For convenience, we reproduce here our main results from the previous chapter that concern the behavior of fitness-proportionate selection in the Hawk-Dove game. For additional detail, please see Chapter 4. Equation 5.1 presents the payoff matrix for the Hawk-Dove game; this is a symmetric variable-sum game for two players. Given a two strategy game, such as in Equation 5.1, we can calculate the strategy proportions at which the two strategies are in equilibrium (if such an equilibrium exists) with Equation 5.2. The dynamics of fitness-proportionate selection operating on the Hawk-Dove game are shown in Figure 5.1. The system has a point attractor where the proportion of Hawks in the population is 7/12; this is the polymorphic Nash equilibrium of the Hawk-Dove game.

$$G = \begin{array}{c|cc} & \mathrm{H} & \mathrm{D} \\ \hline \mathrm{H} & -25 & 50 \\ \mathrm{D} & 0 & 15 \end{array} \tag{5.1}$$

$$p_x = \frac{D-B}{A-B-C+D} = \frac{d+w_0-b-w_0}{a+w_0-b-w_0-c-w_0+d+w_0} = \frac{d-b}{a-b-c+d} \tag{5.2}$$

5.3 Competitive Fitness Sharing

Competitive fitness sharing is a diversity maintenance technique for coevolution proposed by Rosin [172]. The games Rosin considers are a subset of zero-sum games—a

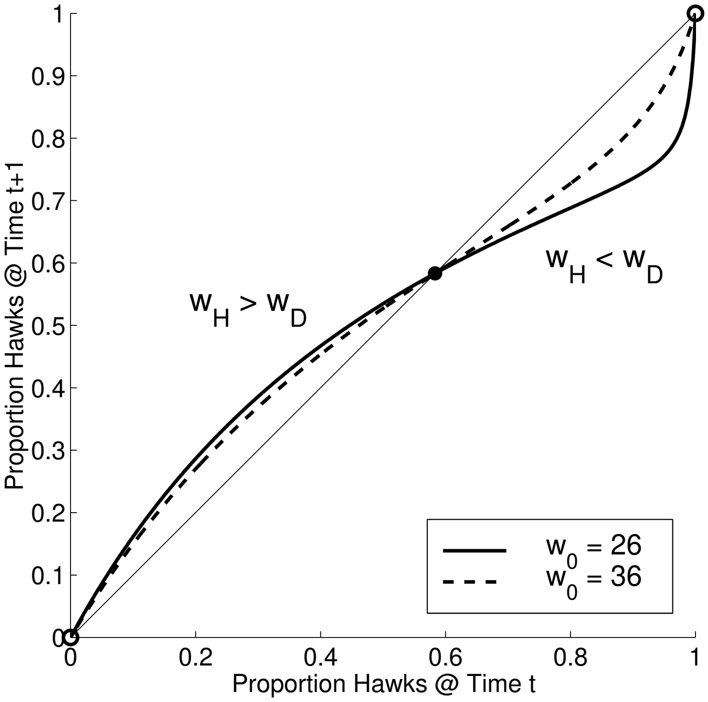


Figure 5.1: Return map of Hawk-Dove game using different values of w_0.

strategy either wins or loses to another strategy. When an agent α defeats an opponent β, the reward given to α is $1/n$, where n is the total number of agents capable of defeating β.

5.3.1 Equilibrium

We can easily extend competitive fitness sharing to variable sum games. The payoff agent α (playing strategy i) receives from interacting with agent β (playing strategy j) is $\frac{\mathbf{G'}_{i,j}}{d}$, where d is the total payoff awarded to all agents by interacting with agent β. In a two-strategy game, we calculate the "shared" fitnesses for strategies x and y as shown in Equation 5.3.

Given these equations, we set $w_x = w_y$ and solve for p_x to find the proportion at which the strategies achieve fitness equilibrium, assuming an equilibrium exists; the solution for p_x is given by Equation 5.4. As with the standard replicator, if a polymorphism does not exist, then Equation 5.4 will give a value outside of the interval $(0, 1)$. Rosin [172] proves that the equilibria for shared and non-shared fitness are the same in win-lose games. But, as Equation 5.4 shows, this is not true in the general case. We see instead that, for an arbitrary two-strategy game, the equilibrium point depends upon the value of w_0. Further, as w_0 gets larger, the equilibrium approaches that of the standard replicator (i.e., non-shared fitness).

$$\begin{aligned} w_x &= \frac{p_x A}{p_x A + p_y C} + \frac{p_y B}{p_x B + p_y D} \\ w_y &= \frac{p_x C}{p_x A + p_y C} + \frac{p_y D}{p_x B + p_y D} \end{aligned} \tag{5.3}$$

$$p_x = \frac{C(D-B)}{BA - 2BC + CD} = \frac{c(d-b) + w_0(d-b)}{ab - 2bc + cd + w_0(a-b-c+d)} \qquad (5.4)$$

5.3.2 Dynamics

Figure 5.2 shows the return map of the Hawk-Dove game under competitive fitness sharing with the same two values of w_0 used in Figure 5.1. Unlike the selection methods we examine above, the map in Figure 5.2 is continuous; further, the fitness equilibrium is an attractor, though it is not the Nash equilibrium of the game. As Equation 5.4 indicates, the fitness equilibrium moves closer to 7/12 as w_0 increases. Again we see that the curves approach the diagonal as w_0 increases, meaning that the difference between two consecutive population states becomes smaller and the population dynamic slows. A value of w_0 sufficiently large to reasonably approximate the equilibrium of 7/12 will cause the dynamics to move very slowly, indeed.

5.4 Similarity-Based Sharing

An older method for diversity maintenance discounts an agent's fitness based upon how genotypically or phenotypically redundant it is with respect to the rest of the population. This method is known as *similarity-based sharing* [85] and was originally used in non-coevolutionary algorithms. Since this method requires a similarity metric to measure redundancy, and we can imagine a number of similarity metrics, there exists many ways to formulate this procedure. Here we investigate one very simple formulation that considers two agents to be similar only if they are playing the same strategy. This metric is analytically convenient, but unlikely to be used in real-world domains.

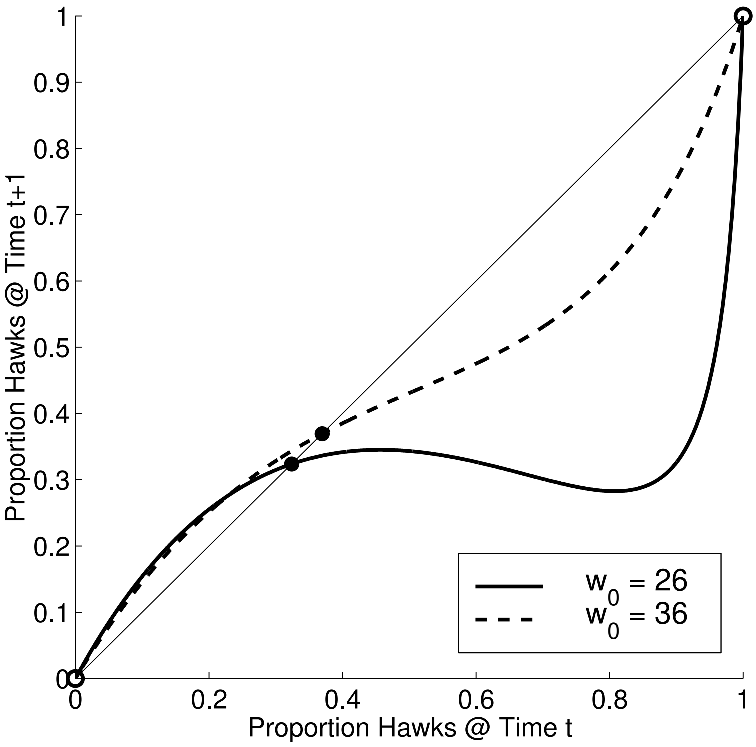


Figure 5.2: Return map of Hawk-Dove game with competitive fitness sharing using different values of w_0.

5.4.1 Equilibrium

Given a game of m pure strategies $\mathbf{G}'$, the shared fitness of an agent playing strategy i is the inner product of payoff matrix row i and population state vector $\mathbf{p}$, divided by the number of agents playing strategy i, thus $\mathbf{w}_i = (\sum_j \mathbf{G}'_{i,j}\mathbf{p}_j)/\mathbf{p}_i$. Equation 5.5 shows how we calculate shared fitness based on similarity in a two-strategy game.

$$
\begin{aligned}
w_x &= \frac{p_x A + p_y B}{p_x} = A + \frac{p_y}{p_x} B \\
w_y &= \frac{p_x C + p_y D}{p_y} = \frac{p_x}{p_y} C + D
\end{aligned}
\tag{5.5}
$$

Given Equation 5.5, we can solve for p_x to find the proportion at which two strategies reach fitness equilibrium, if they do at all. This is somewhat more complex than in competitive fitness sharing because we must find the roots of the polynomial in Equation 5.6. Using the quadratic equation, we get Equation 5.7. We again find that the location of the fitness equilibrium is dependent upon the magnitude of w_0.

Indeed, as w_0 approaches infinity, p_x approaches 1/2 regardless of the payoffs in the game. For this to be true, the square root term must behave as shown in Equation 5.8. The $(a-d)^2$ term in Equation 5.8 becomes insignificant as w_0 grows, so we concentrate on the remaining terms, as shown in Equation 5.9. The square root of Equation 5.9 is approximated by the square root of Equation 5.10 as w_0 increases, which is the right side of Equation 5.8. We use our approximation to simplify Equation 5.7 and obtain Equation 5.11; thus, at $w_0 = \infty$, we get $p_x = \frac{1}{2}$.

$$\mathbf{p}_x^2(-A + B - C + D) + \mathbf{p}_x(A - 2B - D) + B = 0 \tag{5.6}$$

$$\mathbf{p}_x = \frac{-a+2b+d+2w_0}{2(-a+b-c+d)} \pm \frac{\sqrt{(a-d)^2+(2b+2w_0)(2c+2w_0)}}{2(-a+b-c+d)} \tag{5.7}$$

$$\sqrt{(a-d)^2+(2b+2w_0)(2c+2w_0)} \equiv_{w_0 \to \infty} b+c+2w_0 \tag{5.8}$$

$$(2b+2w_0)(2c+2w_0) = 4bc+4bw_0+4cw_0+4w_0^2 \tag{5.9}$$

$$(b+c+2w_0)^2 = b^2+bc+c^2+4bw_0+4cw_0+4w_0^2 \tag{5.10}$$

$$\frac{-a+2b+d+2w_0-(b+c+2w_0)}{2(-a+b-c+d)} = \frac{1}{2} \tag{5.11}$$

5.4.2 Dynamics

Figure 5.3 shows the return map for the Hawk-Dove game using our formulation of similarity-based fitness sharing. The values of w_0 we use are 26 and 56; we see that the polymorphic equilibrium moves towards 1/2 as w_0 increases. Like the map for competitive fitness sharing, the map in Figure 5.3 is continuous at the fitness equilibrium; the equilibrium is an attractor, but not the Nash equilibrium of the game. Because the slope of the curve is negative at fitness equilibrium, this system approaches equilibrium via a damped oscillation rather than a monotonic convergence. Another feature of this map is that it is discontinuous at $p_x = 0$ and $p_x = 1$, unlike the previous maps.

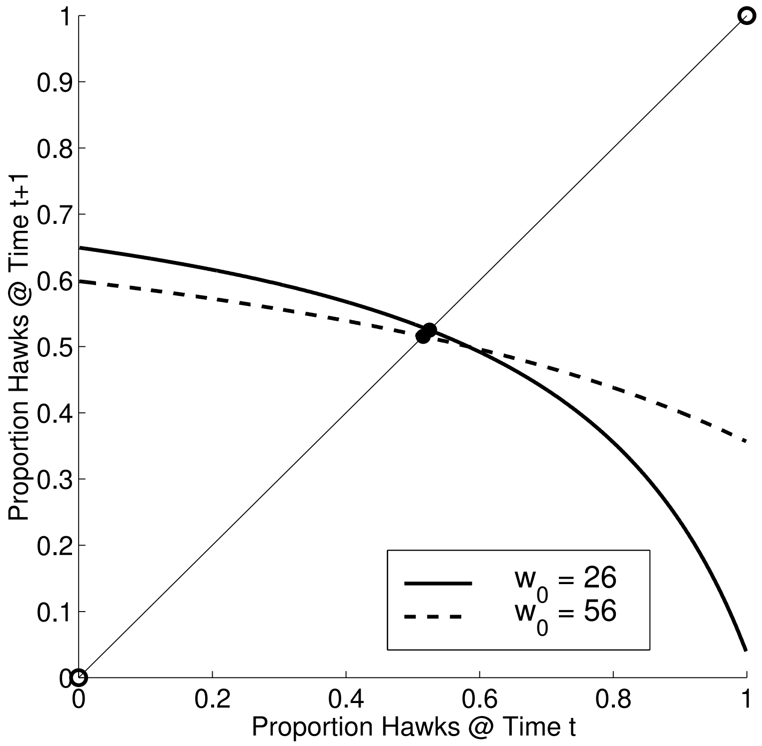


Figure 5.3: Return map of Hawk-Dove game with similarity-based fitness sharing using different values of w_0.

5.5 Phantom Parasite and Shared Sampling

Rosin and Belew consider asymmetric zero-sum games where the outcomes are either win or lose. They are careful to verify that, for such games, the equilibria obtained under standard fitness-proportionate selection are unchanged by the addition of competitive fitness sharing [175]. Nevertheless, they do not perform similar verifications of two additional techniques that they offer for use in conjunction with competitive fitness sharing; these techniques are *shared sampling* and *phantom parasite*.

The phantom parasite method is designed to help prevent disengagement between two populations in an adversarial relationship. Let us arbitrarily designate one population as "hosts" and the other as "parasites." If the host population contains some individuals that beat all the individuals in the parasite population, then these more accomplished hosts will eventually crowd-out the weaker hosts (the accomplished hosts are more fit, even under competitive fitness sharing). If parasites capable of defeating the accomplished hosts are not discovered before the weaker hosts are removed, then the two populations will become disengaged. As a result, both populations will be subject to genetic drift and evolutionary forgetting may occur. Thus, when some hosts defeat all parasites, the weaker hosts are asserted to defeat a phantom parasite that the strong hosts do not. In this way, a niche is developed for the weaker hosts that, when combined with competitive fitness sharing, gives the weaker hosts a fitness advantage when they become too few.

Rosin [172, p. 71] offers the following example and analysis, which we illustrate in Figure 5.4. We have $N-L$ superior hosts that beat all P parasites and L inferior hosts, each of which beat the same set of $P-K$ parasites and lose to the same set of K parasites. With competitive fitness sharing and the phantom parasite, Rosin shows that each superior host receives a fitness of $\frac{K}{N-L}+\frac{P-K}{N}$; each inferior host gets a fitness of $\frac{1}{L} + \frac{P-K}{N}$, where $\frac{1}{L}$ represents the fitness contribution of the phantom parasite. Rosin shows that the inferior hosts will have a fitness advantage over the superior hosts when $L < \frac{N}{K+1}$. Thus, while disengagement is addressed by the combination of competitive fitness sharing and phantom parasite, we clearly see that it is at the cost of distorting the equilibrium. Of course, if the solution we seek is a singleton, then we do not have a problem; if we are open to polymorphisms, then we will misread our result.

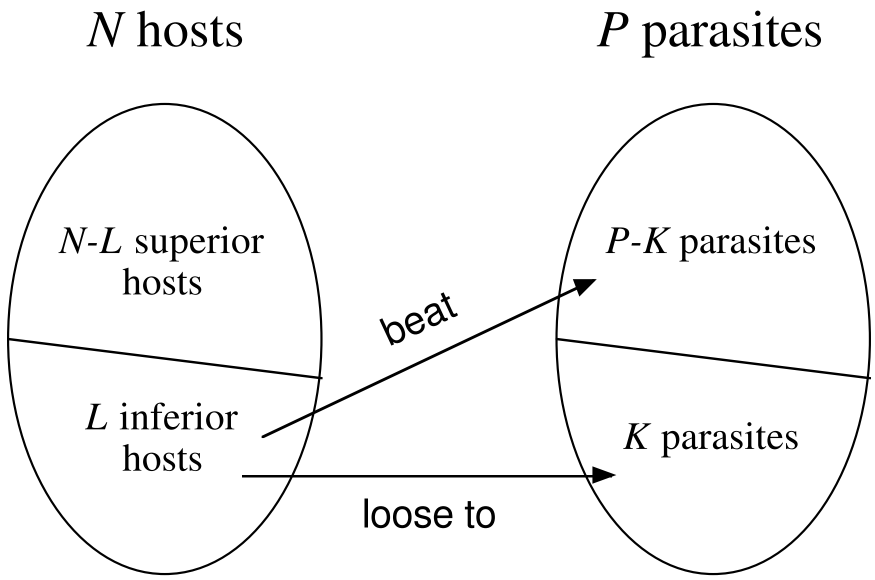


Figure 5.4: Rosin's [172, p. 71] example for phantom parasite operation.

If the cost of computing the outcome of an interaction is too costly, then we will be unable to examine all pair-wise interactions between members of the two populations. In this case, we may resort to sampling. Shared sampling is a heuristic to maximize the diversity of the samples we take. With shared sampling, the distribution used to take the samples follows the fitness values computed with competitive fitness sharing; that is, individuals with higher competitive fitness are more likely to be sampled as interaction partners. Clearly, one effect of this approach is to distort the apparent population state, which in turn distorts fitness values and therefore the equilibria of the game.

5.6 Summary and Conclusions

The purpose of fitness-sharing methods is to maintain genetic diversity, and thereby provide the evolutionary algorithm more genetic raw material with which to perform variation and search. The methods we review above achieve their goal by manipulating the proportions with which the various genotypes, and hence phenotypes, appear in the population. By doing so, fitness-sharing methods may easily act contrary to the requirements of the solution we seek. Indeed, the fitness-sharing procedures cause the polymorphic attractor of the Hawk-Dove game to deviate such that it is no longer Nash. Within the confines of zero-sum games—where competitive fitness sharing does not distort equilibria—we point out that the associated techniques of phantom parasite and shared sampling introduce their own equilibrium distortions.

We should note that the degree of distortion introduced by competitive fitness sharing depends upon not only the magnitude of w_0, but also the number of strategies in the polymorphism. We see considerable distortion when the polymorphism involves only two strategies. But, as more strategies become involved, we get more equations like those in Equation 5.3 and the divisors of those equations involve more and more terms; more importantly, the shared-fitness equations for any two strategies (involved in the polymorphism) will have more and more of their divisor terms in common. Thus, as the polymorphism includes more strategies, the divisors tend to become more similar, leaving us with an approximation of non-shared fitness. The quality of the approximation still depends upon the payoffs of the game, however, and one generally does not know *a priori* the size of the polymorphism(s) a game will have.

Finally, our preliminary research on spatial populations shows that they too are unable to converge onto polymorphic equilibria. The spatial pattern of interaction typically provides an individual too few partners to form an accurate reflection of the population's true strategy distribution. This creates an effect similar to shared sampling; a distorted experience of the population state leads to distorted fitness values.

Chapter 6

Effects of Finite Populations on Polymorphisms

6.1 Introduction

The primary contribution of evolutionary game theory (EGT) is the concept of the *evolutionary stable strategy* (ESS) [134]. The ESS is a refinement of Nash equilibrium that does not require agents to be rational to attain it. Rather, agents achieve equilibrium through a process of Darwinian selection.

At least three strong assumptions are made in the EGT formalism. First, the evolving population is assumed to be infinitely large. Second, the payoffs that agents receive are assumed to be without noise. Third, each agent is assumed to play against every other agent to determine its fitness—there is *complete mixing*. Recently, Fogel, et al, have questioned the usefulness of evolutionary game theory in real-world situations where these assumptions, particularly the first, do not hold [73, 74, 75]. They begin their simulations with the population precisely at the ESS, and discover that the population consistently moves away from it. At best, their simulation results differ from theoretical ESS values with statistical significance; at worst, their results bear no semblance to the ESS whatsoever. Thus, they conclude that evolutionary game theory loses predictive power once these assumptions are relaxed.

In this chapter, we concentrate on the first (and perhaps strongest) of the above assumptions and revisit the question of whether ESS dynamics can exist in finite populations. Fogel, et al, report using two different selection methods, truncation and proportional roulette-wheel selection. By paying particular attention to the operation of the selection mechanism, we are able to account for the divergence between ESS predictions (based on infinite populations) and results observed in our own finite-population simulations. We then examine Baker's SUS selection method [20] that corrects the divergence to a great extent. We thus conclude that the dynamics of evolutionary game theory, and particularly the ESS, can indeed apply to finite-population systems. Further, the selection method used in a simulation can distort, or even disrupt completely, the dynamics we may expect to see.

We begin with a brief introduction to evolutionary game theory in Section 6.2, and then review and analyze relevant previous work in Section 6.3. In Section 6.4 we introduce the methodology used in our own finite-population simulations and give results. Section 6.5 analyzes the operation of our selection method and Section 6.6 constructs a Markov model to predict the amount of divergence to expect for a particular population size. Section 6.7 examines the performance of Baker's SUS selection method. Section 6.8 focuses on the role of map asymmetry in the divergence observed with finite populations. Section 6.9 offers final remarks. The work in this chapter is first published in Ficici and Pollack [61].

6.2 Evolutionary Game Theory

The Hawk-Dove game forms the backdrop for our investigation (as in [73, 74, 75]); it has two *pure strategies*, H (hawk) and D (dove). The *payoff matrix*, G, for this game is shown in Equation 6.1. Each entry, $E(i, j)$, in row i, column j, is the *expected value* of the payoff given to an agent playing strategy i when matched against an agent playing strategy j—payoffs are assumed to be without noise. The evolving population of agents is assumed to be infinitely large. The proportions with which the strategies of G are used in the population can be represented by a column vector, p; the elements of p sum to 1.0. The *fitnesses*, f, of the strategies are determined by a weighted sum of the payoffs in G according to the proportions in p, and can be computed by matrix multiplication, as in Equation 6.2. This equation assumes *complete mixing*, that is, all agents play against all others. The next generation of agents is formed through *fitness-proportionate selection*—each strategy increases its representation, or "reproduces," in direct proportion to its fitness. The reproductive process is described by the difference equation shown in Equation 6.3.

$$G = \begin{array}{c|cc} & \mathrm{H} & \mathrm{D} \\ \hline \mathrm{H} & -25 & 50 \\ \mathrm{D} & 0 & 15 \end{array} \tag{6.1}$$

$$f = G * p + w_0 \tag{6.2}$$

$$p' = \frac{p \times f}{p \bullet f} \tag{6.3}$$

where $w_0 = 26$ is a constant added to fitnesses such that they are all greater than zero, '$\times$' is element-wise multiplication, and '$\bullet$' is inner product. The lower term in Equation 6.3 is for normalization.

At the heart of evolutionary game theory is the concept of the evolutionary stable strategy. A population of agents that play an ESS cannot be "invaded" by a small number of agents that play some mutant strategy; hence, the population is evolutionarily stable. A *polymorphic* ESS actually involves two or more pure strategies, all of which receive the same fitness. The Hawk-Dove game has a polymorphic ESS where $7/12^{ths}$ of the population plays hawk ($p_{\mathrm{H}} = \frac{7}{12}$), and $5/12^{ths}$ dove ($p_{\mathrm{D}} = \frac{5}{12}$). According to Maynard-Smith, for a strategy (or polymorphic population), i, to be an ESS it must fulfill one of these two requirements against a mutant strategy, j, for all $j \neq i$[134]:

$$E(i,i) > E(j,i) \quad \textbf{OR} \tag{6.4}$$

$$E(i,i) = E(j,i) \quad \textbf{AND} \quad E(i,j) > E(j,j) \tag{6.5}$$

6.3 Review of Previous Work

In this section, we review the experiments reported in [73, 74, 75], where Fogel, et al, examine various factors that may influence EGT dynamics. All experiments have finite populations and generational reproduction. Two different selection methods are used. The *truncation* selection method operates by sorting agents according to fitness and then replacing the worst v percent of the agents with copies of the best v percent. The value of v lies in the interval [0, 50]; higher values exert a higher selection pressure. The *fitness-proportionate roulette-wheel* selection method biases a random variable (the roulette wheel) in proportion to the agents' fitnesses. The wheel is then "spun" n times to create an offspring population of size n.

6.3.1 Truncation Selection

In [73], truncation selection is used with maximum selection pressure, $v = 50\%$. This means that the worst half of the population is replaced by copies of the best half to form the next generation. Three experiments are described. Population sizes of 60, 600, and 6000 are used in each experiment and all pair-wise encounters occur during fitness evaluation (complete agent mixing). The first experiment adds a form of noise to the payoffs; the second introduces an additional, but slight, mutation bias (such that a hawk might become a dove, and vice versa). The third experiment is presented as a control—payoffs without noise and no mutation. In every case, the population is begun at the ESS and observed to move away. The results of the first two experiments are reported as essentially indistinguishable: for population sizes of 60 and 600, both give apparently chaotic behavior or a series of short-lived, quasi limit cycles. A population of size 6000, however, converges to all hawks in the first experiment (the second experiment always introduces some number of mutations). In the control experiment, the population is reported to jump immediately to all hawks. Given the clear absence of an ESS, Fogel, et al, conclude that the dynamics of evolutionary game theory not only assume, but *require* an infinite population.

Nevertheless, EGT also assumes fitness-proportional selection—not truncation. Let us consider the behavior of truncation selection in the case of noiseless payoffs. At the ESS, both strategies (and hence all agents) receive the same fitness: $p_{\mathrm{H}} = \frac{7}{12} \Rightarrow f_{\mathrm{H}} = f_{\mathrm{D}}$. In contrast, $p_{\mathrm{H}} < \frac{7}{12} \Rightarrow f_{\mathrm{H}} > f_{\mathrm{D}}$, and $p_{\mathrm{H}} > \frac{7}{12} \Rightarrow f_{\mathrm{H}} < f_{\mathrm{D}}$. This describes a simple feedback mechanism (that regulates properly with proportional selection).

Let us consider a population where the proportion of hawks is $.5 \leq p_{\mathrm{H}} < \frac{7}{12}$. Because the proportion of hawks is below the ESS, the hawks will receive higher fitness than the doves. Yet, the hawks comprise at least 50% of the population. Thus, the top 50% of the population can only contain hawks, and so, with truncation selection, the next generation must contain 100% hawks. This is true regardless of the size of the population—indeed, *even an infinite population*. At the ESS itself, the behavior of truncation selection is ill-defined because all agents have the same fitness—who are in the top 50%? Unless special care is taken to account for ties, the fixed point of all hawks will again emerge. The results of the third experiment (control) are consistent with this analysis.

Now let us consider the first experiment, which adds the following noise to the payoffs: rather than give both hawks in a hawk-hawk confrontation the *expected* payoff of -25, one is randomly chosen to receive a payoff of -100 while the other receives 50; similarly, when two doves meet, one is randomly chosen to receive a payoff of 40 while the other receives -10, instead of the expected payoff of 15. The payoffs of hawk-dove encounters are unchanged. Thus, after all pair-wise encounters occur, the two *strategies* still obtain equal fitnesses at the ESS; but, individual *agents* may do better or worse than others. We strongly suspect that the population of 6000 converges to all hawks because the higher number of encounters per agent allows expected payoff values to emerge. With a good enough approximation of expected payoff, the dynamics of the system are similar to noiseless payoffs, above.

As we know from Chapter 4, truncation selection cannot converge onto a polymorphic equilibrium, even when the population is of infinite size. Instead, we find a fixed-point of all Hawks, cyclic behavior, or chaos, depending upon the selection pressure; these infinite-population results agree with those reported by Fogel, et al. Given that truncation selection so completely disrupts the dynamics of the Hawk-Dove game with an infinite population, we believe the experiments in Fogel, et al, [73, 74, 75] that use truncation implicate the selection method more than other factors; the effects of noisy payoffs, incomplete mixing, and finite populations are inadequately isolated to warrant the stated conclusions.

6.3.2 Roulette Selection

The simulations of Fogel, et al, that do not use truncation selection use fitness-proportionate roulette-wheel selection instead [74]. Their core concern is whether evolutionary game theory can be applied to biological field study. Thus, if evolutionary game theory is applied to a finite population of physically embodied agents, such as animals, then clearly no pair-wise encounter will match an individual against itself. In this case, computation of fitness is slightly different than in Equation 6.2 because the "self" must be subtracted from the counts. For the Hawk-Dove game, we have:

$$\begin{aligned} f_{\mathrm{H}} &= E(H,H) * (p_{\mathrm{H}} - \frac{1}{n}) + E(H,D) * p_{\mathrm{D}} + w_0 \\ f_{\mathrm{D}} &= E(D,H) * p_{\mathrm{H}} + E(D,D) * (p_{\mathrm{D}} - \frac{1}{n}) + w_0 \end{aligned} \tag{6.6}$$

where f_i is the fitness of strategy i, $E(i,j)$ is the payoff for strategy i against strategy j, p_i is the proportion of strategy i in the population, and n is the population size.

Fogel, et al [74], recognize that variations in population size cause Equation 6.6 to change the proportion at which the two strategies reach equal fitness. In the limit of an infinite population, Equation 6.6 converges to the familiar ESS proportion of $\frac{7}{12}$ Hawks. For finite population sizes, however, the fitness-equilibrium proportion is actually higher. Their experiments that use fitness-proportional roulette-wheel selection give data that deviate with statistical significance from the theoretical ESS. But, the null hypothesis they choose to measure against is the equilibrium for an infinite population, and not that predicted by Equation 6.6, which varies according to population size. (Note that having an equilibrium higher than $\frac{7}{12}$ does not invalidate the above argument regarding truncation selection.)

6.4 The Effects of Finite Populations: A Second Look

6.4.1 Assumptions

We now proceed with our own experiments. Since truncation selection appears pathological in the context of evolutionary game theory, we will concentrate on proportional roulette-wheel selection in our simulations. We assume complete mixing and noiseless payoffs. We will also assume that an agent *can* play against itself. While this may lack biological plausibility, it is entirely possible—indeed common—in *coevolutionary algorithms*. The important mathematical consequence is that we can revert back to Equation 6.2 and avoid the problems caused by Equation 6.6, above. Thus, we can better isolate the effects of noise due to quantization and stochastic sampling that arise with a finite population. Rather than use Maynard-Smith's static conception of an ESS, we choose instead to use the formalism of dynamical systems to describe stability concepts. As we discuss below, the ESS of the Hawk-Dove game is an *attractive fixed point*.

6.4.2 Methods

We implement proportional roulette-wheel selection as follows: the fitness scores, f_i, are normalized, such that they sum to 1.0; a vector of sum *prefixes* is computed, such that the prefix value for agent i is the sum:

$$r_i = \sum_{k=1}^{i} f_k \tag{6.7}$$

We draw a value, x, from a uniform distribution and select the first agent whose prefix value is $\geq x$ to produce one offspring. This step is performed n times, where n is the size of the desired population.

We begin each experiment with the population at the theoretical ESS ratio of $p_{\mathrm{H}} = \frac{7}{12}$ (and $p_{\mathrm{D}} = \frac{5}{12}$). The population sizes for all experiments are chosen such that the ESS ratio is precisely representable by whole numbers of agents. Five different population sizes are tested: 60, 120, 300, 600, and 900. Each simulation is run for 2000 generations. The mean number of hawks in a run constitutes a single data point in an experiment; each experiment is repeated 100 times.

6.4.3 Results

Our results are shown in Figure 6.1. The **x** and **y** axes indicate the population size used in an experiment and the proportion of hawks in the population, respectively. Each circle represents the mean proportion of hawks seen during a single run. The dashed line indicates the ESS that evolutionary game theory predicts for an infinite population. The solid curve indicates the mean value of the 100 trials in each experiment. We see that for all population sizes, the mean value of hawks over all trials is consistently lower than the theoretical ESS proportion. As the population size increases, the mean value asymptotically approaches the theoretical ESS. The second column of Table 6.1 lists the observed means and the fifth column gives t-values obtained when applying the t-test to the data from each experiment with respect to a null hypothesis $H_0 = .58333\ldots$ (the ESS). In all cases, the observed data deviate with statistical significance from the theoretical ESS.

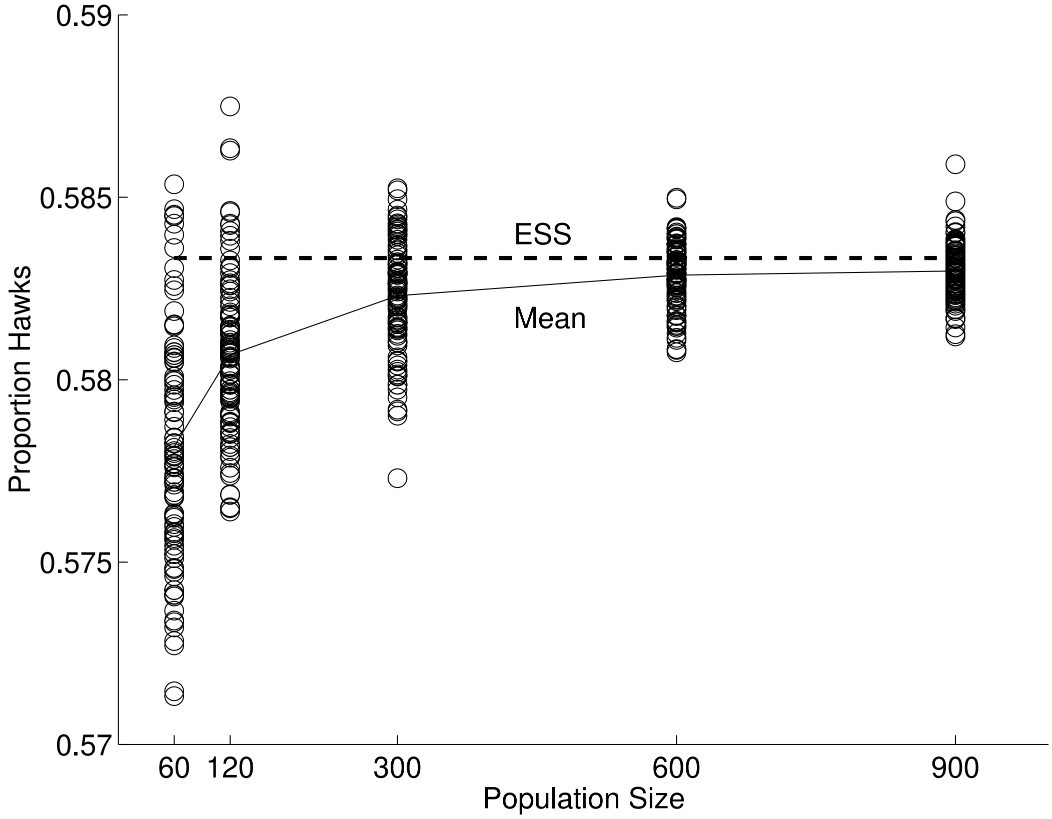


Figure 6.1: Data from finite-population simulations of Hawk-Dove game. Five experiments are conducted, each repeated 100 times. Each circle represents the mean proportion of hawks observed during a single 2000-generation simulation. The dashed line is the theoretical ESS. The solid curve indicates the mean values of the 100 data points in each experiment.

6.5 Analysis of Roulette Selection

That the data means come closer to the theoretical ESS as population size increases is perhaps easy to accept. What is less intuitive, however, is why the approach should be asymptotic from below the ESS value. To answer this question, we turn to dynamical systems theory [163, 188]. Figure 6.2 (Top) shows the *map diagram* for the Hawk-Dove game. Given a proportion of hawks, represented by the x-axis, one iteration of the evolutionary difference equation (Equation 6.3) will produce a new proportion, represented by the y-axis. The curve depicts the function that maps the proportion of hawks from one generation to the next. Intersections of the curve with the diagonal line indicate *fixed points*. If the slope of the curve at an intersection has an absolute value less than one, then the fixed point is *stable*; if the absolute value is greater than one, then the fixed point is *unstable*. The Hawk-Dove game has three fixed points: two unstable, where the population is either all hawks or all doves, and one stable, the polymorphic evolutionary stable strategy.

To see the dynamics of the Hawk-Dove game, one simply picks an initial point on the x-axis, draws a vertical line to the curve, and then alternately draws a horizontal line to the diagonal and then a vertical line to the curve until the ESS is reached. This procedure produces a *cobweb diagram* that indicates the *orbit* of the initial point. Two such orbits are shown in Figure 6.2 (Top), which begin ± 0.25 away from the ESS ($p_{\mathrm{H}} = \frac{7}{12}$). The key observation is that the orbit that begins below the ESS requires more iterations to reach the ESS than does the orbit that begins above. Figure 6.2 (Middle) shows the number of iterations required for every initial condition from 0 to 1, at intervals of 0.001, to arrive within $\epsilon = 0.0001$ of the ESS. For any δ, orbits starting at $\frac{7}{12} - \delta$ require $\geq$ iterations than orbits starting at $\frac{7}{12} + \delta$.

This observation is important because the operation of the roulette-wheel method of proportional selection produces a *binomial distribution* around the desired proportion of hawks indicated in the wheel. The binomial distribution obtained for n spins of a roulette wheel that gives a hawk with probability p is:

$$\mathrm{bin}(n,p) = \left(\begin{array}{c} n \\ i \end{array} \right) * p^i * (1-p)^{n-i}, \textbf{for } i = 0 \ldots n \tag{6.8}$$

That is, the probability of having exactly i hawks (and therefore $n-i$ doves) in the next generation of n agents is the probability of getting i hawks times the probability of getting $n-i$ doves times the number of ways n spins can result in i hawks. The resulting distribution for all possible outcomes, from no hawks to no doves, has a mean value of $p * n$ hawks—the desired proportion indicated in the wheel.

Figure 6.2 (Bottom) shows the "spread" of the binomial distribution for values of $n = \{60, 120, 300, 600, 900\}$ (the population sizes used in our simulations, above) and $p = \frac{7}{12}$ (the desired proportion of hawks at the ESS). Each curve in Figure 6.2 (Bottom) represents the possible outcomes that sum to 99 percent of likelihood. The spread clearly widens as the population size decreases (and the law of large numbers exerts less influence).

Returning to Figure 6.2 (Middle), we see that an increase in spread at the ESS will cause the asymmetry in convergence time to grow. This is what causes the observed population averages to tend to fall below the theoretical ESS by an amount inversely related to population size. But, how might we approximate the actual divergence without resorting to empirical methods? We offer an approach based on Markov chains.

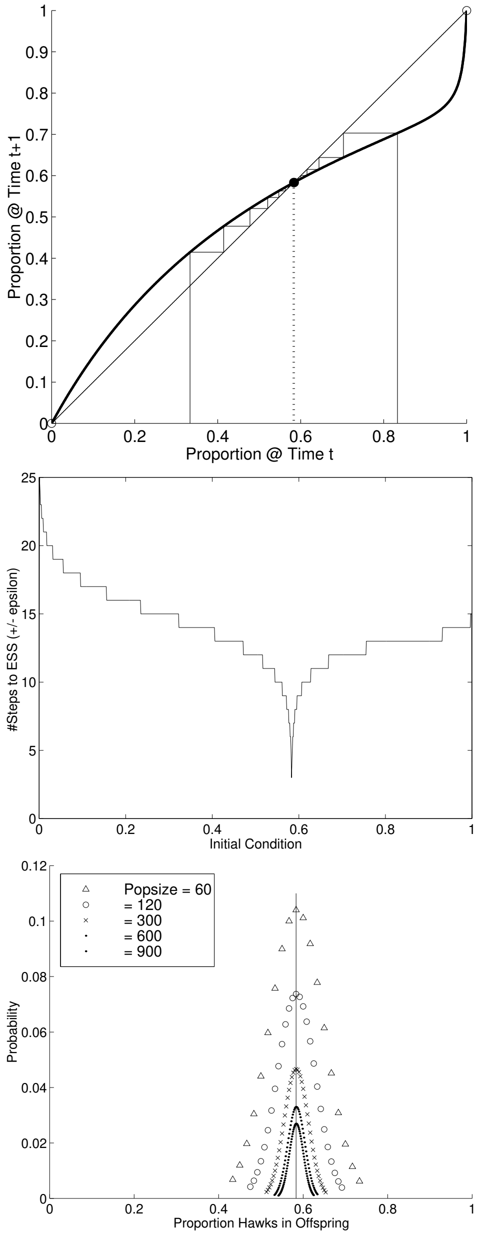


Figure 6.2: Interaction of the map with the binomial distribution. Top: Map diagram of Hawk-Dove game. The curve depicts the function that maps the proportion of hawks from one generation to the next. Middle: Number of iterations needed to arrive within $\epsilon = 0.0001$ of the ESS. Bottom: Spread of binomial distribution at ESS for various population sizes.

6.6 Calculating the Divergence

6.6.1 Method

In this section, we develop the intuition from above to predict how experimental data will deviate from the theoretical ESS value. For the Hawk-Dove game (or any two-strategy game), there are only $n+1$ possible states in which a population of size n can be, from 0 hawks to n hawks. These can be considered the states of a Markov chain. At each population state, i, the binomial distribution of the roulette wheel gives a set of *transition probabilities* to states $j = 0 \ldots n$. This information is represented by a *transition matrix*, M:

$$M = \begin{bmatrix} \Pr(0|0) & \ldots & \Pr(0|n) \\ \vdots & \ddots & \vdots \\ \Pr(n|0) & \ldots & \Pr(n|n) \end{bmatrix} \tag{6.9}$$

where each column gives the transition probabilities, $p(j|i)$, from state i to states $j = 0 \ldots n$. Each column sums to 1.0.

To compute the matrix columns, we need only calculate the appropriate binomial distribution for each possible state of the population. Recall that for any *desired* proportion of hawks, d, the binomial distribution $\text{bin}(n, d)$ tells us the likelihoods of all *possible* proportions of hawks, given n spins of the roulette wheel. For a population with i hawks (a proportion of i/n), the desired number of hawks for the next generation is $d = \text{map}(i/n)$, where the function *map* corresponds to an iteration of the map diagram in Figure 6.2 (Top). Thus, the columns of the transition matrix, M, are:

$$M = [\text{bin}(n, \text{map}(\frac{0}{n})) \ldots \text{bin}(n, \text{map}(\frac{n}{n}))] \tag{6.10}$$

Let us consider an initial population composed of exactly $\frac{7}{12}$ hawks, and represent this population as a column vector, b, of probabilities on the different possible population states:

$$b = \begin{bmatrix} \Pr(0) \\ \vdots \\ \Pr(\frac{7}{12}n - 1) \\ \Pr(\frac{7}{12}n) \\ \Pr(\frac{7}{12}n + 1) \\ \vdots \\ \Pr(n) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6.11}$$

The distribution of possible populations after t generations is M raised to power t times b:

$$b^t = (M)^t * b \tag{6.12}$$

Therefore, the expected proportion of hawks after t generations is given by the weighted sum:

$$\mathrm{E}[\%hawks]^t = \sum_{i=0}^{n} \frac{b_i^t * i}{n} \tag{6.13}$$

What is the limit behavior, as time goes to infinity? Observe that roulette selection always gives a non-zero probability of arriving at a state of all hawks or all doves. And, since a population of all hawks will remain all hawks, and a population of all doves will remain all doves, we know that, in infinite time, one of these two *absorbing states* is an inevitable outcome. The probabilities of these two absorbing states are sensitive to the initial state.

Thus, the distribution, b^∞ must have the form:

$$b^\infty = [\Pr(0); 0; 0; \ldots; 0; 0; \Pr(n)] \tag{6.14}$$

where only states 0 and n have non-zero probability.

Referring back to Equation 6.13, this implies that the expected proportion of hawks is simply the probability of ending in the absorbing state of all hawks. Thus, $\mathrm{E}[\%hawks]^\infty = b_n^\infty$.

6.6.2 Predictions and Data

For the population sizes used in our experiments, the expected number of hawks for even small values of t gives an excellent approximation of the limit behavior: $\mid b_n^\infty - b_n^t \mid < \epsilon$ for small t. Figure 6.3 shows the values of Equation 6.13 for values of $t = 1 \ldots 30$. We see that the limiting distribution is approached asymptotically. Our approximate expected values (computed at $t = 100$) are listed in column three of Table 6.1. We see that they are very close, indeed, to the actual means observed in our experiments (shown in column two). Further, using these predictions as null hypotheses in the t-test, the resulting t-values (shown in column 4) indicate that none of the predictions can be rejected (95% confidence level). Thus, the dynamics and equilibria of evolutionary game theory have been demonstrated to exist and apply to finite populations.

6.7 Stochastic Universal Sampling

While the predicted divergence from the theoretical ESS is clear when we look at an ensemble of 100 data points, an individual run can show a rather large amount

Table 6.1: Simulation results and predictions.

Size	*Mean Hawks*	*Adj. ESS*	$t_{Adj.}$	t_{ESS}
60	0.578166	0.578008	0.4922	-16.09
120	0.580682	0.580755	-0.3289	-11.99
300	0.582301	0.582320	-0.1267	-6.72
600	0.582864	0.582829	0.3953	-5.31
900	0.582981	0.582998	-0.2132	-4.47

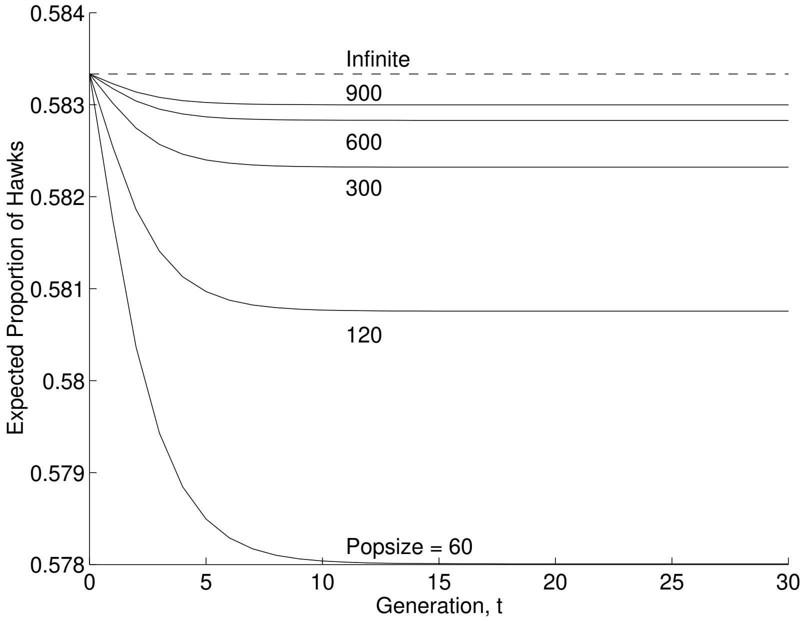


Figure 6.3: Adjustment of ESS. Though small in magnitude, the adjusted values vary from the ESS enough to cause statistically significant deviation from $\frac{7}{12}$ in simulation data (indicated by column 5 of Table 6.1).

of noise due to the roulette wheel. Figure 6.4 (top) shows 500 generations of the Hawk-Dove game beginning at the ESS for a population of size 60. Though noise is reduced with a larger population, there exists an alternative selection scheme that reduces the noise even further while maintaining a small population.

Baker's *Stochastic Universal Sampling* (SUS) [20] provides fitness-proportionate selection with minimal use of a stochastic process. Simply, rather than spin a roulette wheel with one "pointer" on it n times, we spin a roulette wheel with n equallyspaced pointers just once. This method guarantees that an individual that should appear with proportion, p_{Ideal}, will appear with proportion $\lfloor p_{\text{Ideal}} * n \rfloor / n \leq p_{\text{Actual}} \leq \lceil p_{\text{Ideal}} * n \rceil / n$. Baker's analysis of this method indicates that is has excellent statistical properties.

Indeed, SUS produces far less noisy results than ordinary proportional roulette selection, as seen in Figure 6.4 (bottom). Because the ESS can be exactly represented by a population of size 60, we instead choose a population of 61. We see that the actual proportions remain as close to the theoretical ESS as the resolution of the population allows. In fact, SUS will converge to the ESS within the resolution of the population regardless of the initial condition ($0 < p_{\text{H}}^{Initial} < 1.0$). For a population of size 100, SUS selection gives data that cannot be rejected by the t-test (at 95% confidence level) with a null hypothesis of $H_0 = \frac{7}{12}$. That is, because SUS uses only a single "spin," the distribution of possible outcomes is made so narrow that the calculation of divergence from the theoretical ESS is unnecessary.

6.8 Map Asymmetry and Divergence

In the Hawk-Dove game we examine above, we observe three types of asymmetry. First, the polymorphic attractor is asymmetric in the sense that it is located at $p = \frac{7}{12}$ rather than at 0.5. Second, as we note above, the map is asymmetric about the polymorphic attractor; that is, the curve in Figure 6.2 (top) is closer to the diagonal on the left of the attractor than it is on the right. Thus, for any $\delta \in (0, \frac{5}{12})$, an orbit starting at $\frac{7}{12} - \delta$ will take longer to converge onto the attractor than an orbit starting at $\frac{7}{12} + \delta$. Third, the binomial distribution at $p = \frac{7}{12}$ is asymmetric; though the expected value of this distribution is $\frac{7}{12}$, there is actually more probability mass on the right side of the expected value than on the left (the difference depends upon

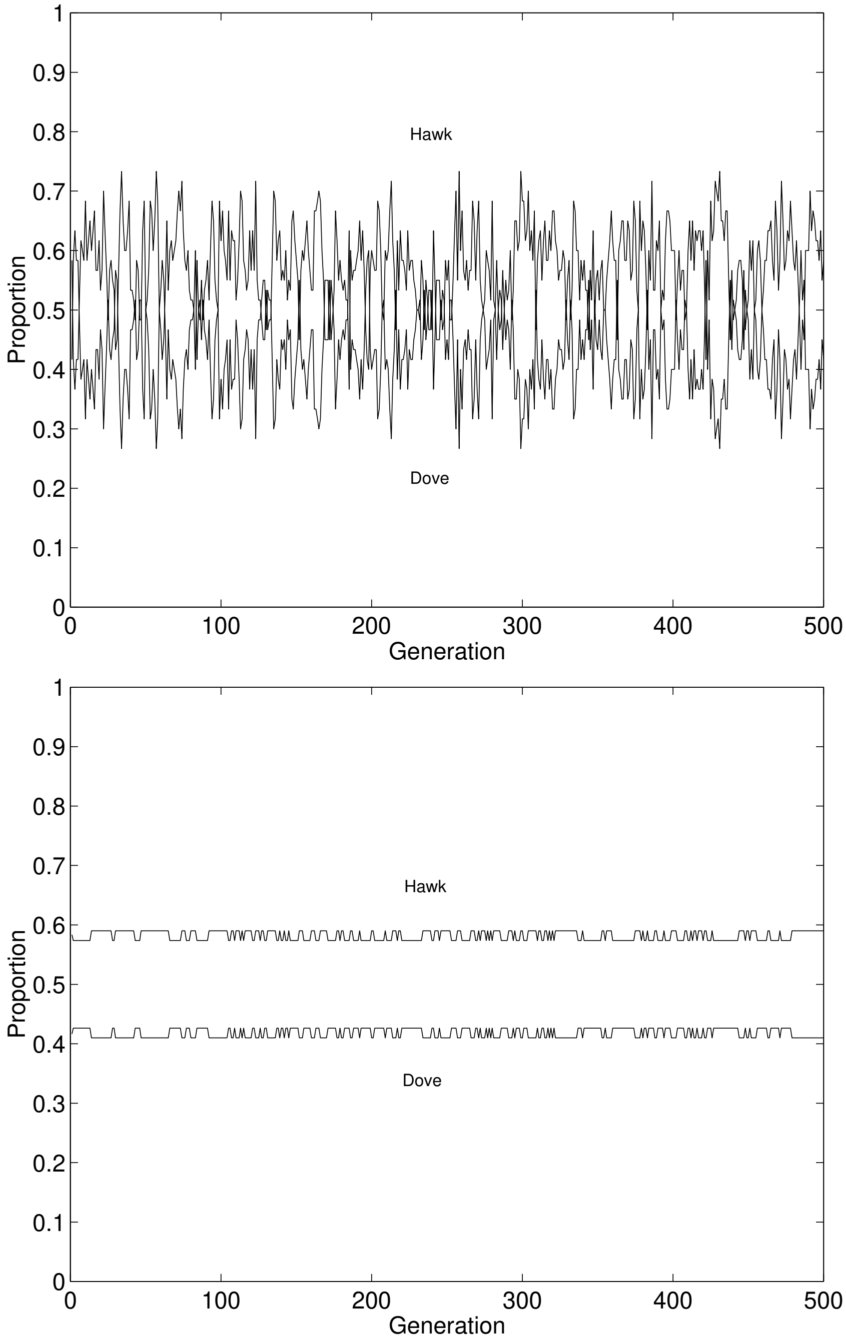


Figure 6.4: Different implementations of proportional selection. Top: Sample run with population size 60 using standard roulette-wheel selection. Bottom: Sample run with population size 61 using Baker's *Stochastic Universal Sampling* selection [20].

population size). In this section we remove the first and third types of asymmetry to concentrate on how the shape of the map affects the expected population state.

The three games in Equations 6.15–6.17 all have a polymorphic attractor at $p = 0.5$; thus, with respect to equilibrium, these three games are identical. Since the polymorphism occurs at $p = 0.5$, the first type of asymmetry we describe above is eliminated. Further, the binomial distribution around $p = 0.5$ is symmetric, eliminating the third type of asymmetry described above.

Despite these games having identical equilibria, the games have very different map symmetries. Our "reference" game, G_{ref}, has a symmetric map, as shown in Figure 6.5 (top). This symmetry suggests that a population begun at the attractor should remain there, on average; indeed, the Markov chain model shows that the expected population state remains at $p = 0.5$. In contrast, game G_{high} is asymmetric about the attractor, as shown in Figure 6.5 (middle). Specifically, the map tends to be nearer the diagonal for population states $p > 0.5$ than below. Thus, the rate of return to the attractor is slower for population states above equilibrium than below. For a population of size 60 starting at $p = 0.5$, the Markov model (at steady-state) shows that the expected population state is $p = 0.5995$—a considerable distortion of the infinite-population equilibrium. Finally, the game G_{low} simply inverts the asymmetry of G_{high}, shown in Figure 6.5 (bottom); the Markov model accordingly gives an expected population state of $p = 0.4004$.

$$G_{ref} = \begin{array}{c|cc} & \text{A} & \text{B} \\ \hline \text{A} & 1 & 2 \\ \text{B} & 2 & 1 \end{array} \tag{6.15}$$

$$G_{high} = \begin{array}{c|cc} & \text{A} & \text{B} \\ \hline \text{A} & 10 & 2 \\ \text{B} & 11 & 1 \end{array} \tag{6.16}$$

$$G_{low} = \begin{array}{c|cc} & \text{A} & \text{B} \\ \hline \text{A} & 1 & 11 \\ \text{B} & 2 & 10 \end{array} \tag{6.17}$$

6.9 Conclusion

There is no question that the introduction of noise, be it from the finiteness of a population, variation in payoffs, or incomplete mixing, affects the dynamics and equilibria in evolutionary game theory. And, Fogel, et al, are correct to highlight the issue. But, the selection method distorts the process, as well—it can never be made transparent. In simulations that deal with finite populations, reproduction generally cannot occur in exact proportion to fitness—agents can only replicate whole-numbers of offspring. Thus, the question of how one is to *implement* reproduction, fitness-proportionate or otherwise, arises. All these sources of "noise" determine whether the essential character of evolutionary game theory remains intact.

Truncation selection is shown to be pathological in the context of evolutionary game theory—it can neither attain nor maintain a polymorphic ESS (even with infi-

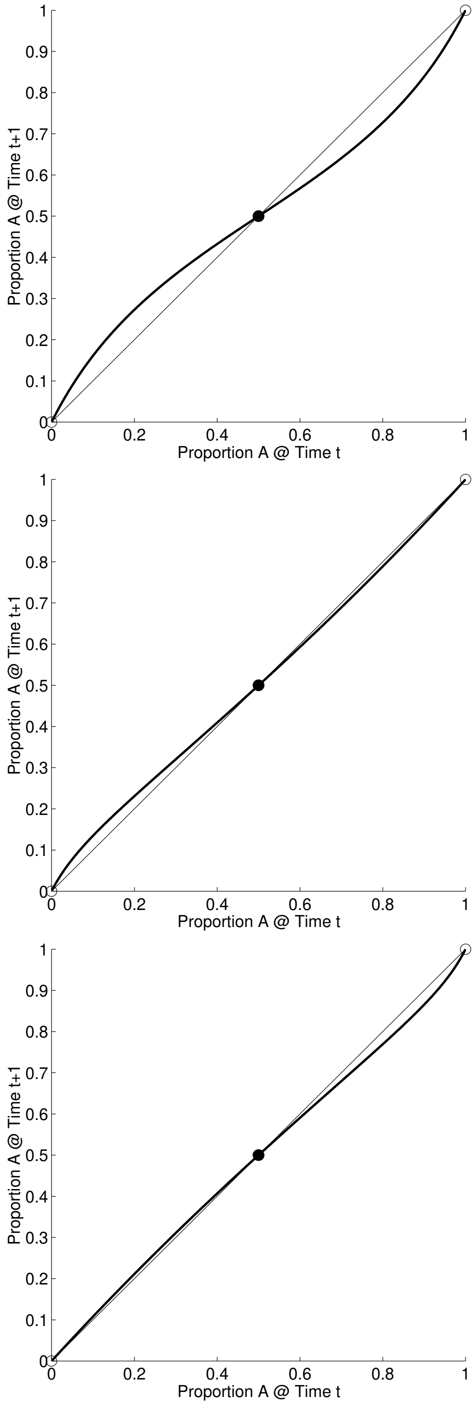


Figure 6.5: Three maps with identical equilibria ($p = 0.5$) yet different symmetry. Top: Reference game G_{ref}; this map is symmetric about the equilibrium. Middle: Game G_{high}. This map is asymmetric about the equilibrium and tends to be nearer the diagonal for population states $p > 0.5$ than below; thus, the rate of evolution is slower for population states above equilibrium than below. Bottom: Game G_{low}. This map has precisely the opposite asymmetry as that of game G_{high}; thus, the rate of evolution is slower for population states below equilibrium than above.

nite populations). Roulette-wheel selection is shown to diverge from the theoretical ESS due to the interaction between the wheel's binomial distribution and the convergence properties of the Hawk-Dove map (Figure 6.2, top)). We show how the expected divergence can be computed without empirical data. Finally, we show that Baker's SUS selection method allows a finite population of modest size to approximate the ESS without statistically significant deviation. If the population size n can be divided into whole numbers that precisely represent the ESS ratio, then SUS selection allows the population to converge to the exact ESS. Thus, we have demonstrated that the dynamics and equilibria of evolutionary game theory can persist with finite populations, provided that the selection method is appropriately chosen and implemented.

Of course, this study has only addressed the smallest of games; what happens when the number of strategies in a population is comparable to the population's size? This is akin to representing the state of a dynamical system with very low resolution, and we will need to return to dynamical systems theory for a satisfactory answer.

While the primary concern of Fogel, et al, is the applicability of evolutionary game theory to biological study, we make no attempt to relate our results and methods to a biological context—we are primarily interested to learn what affects we might encounter by using a finite population in the context of optimization, where we may assume that an agent can play against itself.

Chapter 7

Pareto Optimality: A Solution Concept for Coevolution

> **Inventor,** ***n.*** A person who makes an ingenious arrangement of wheels, levers and springs, and believes it civilization. *Ambrose Bierce* [1]

7.1 Introduction

The challenge facing an agent situated in a coevolutionary domain can be described as a form of multi-objective optimization (MOO) where every other agent encountered constitutes an objective to be optimized. That the techniques of conventional MOO, most notably those incorporating the notion of *Pareto optimality*, may be usefully applied to coevolution has recently been suggested [62, 201], but has yet to be deeply explored. This chapter begins our investigation into the connection between coevolution and multi-objective optimization, paying particular attention to how the Pareto optimality concept may lead to methods that address issues unique to co-

[1] *The Devil's Dictionary.* Neale Publishing Company, 1911. (Dover 1993 republication.)

evolution. (See Noble and Watson [147] for another example of a "first-generation" Pareto coevolutionary algorithm.)

All coevolutionary systems involve concurrent processes of gradient creation and gradient following. The central themes of coevolution research concern the ways in which these processes interact to dynamically modify the scope of interactions that occur between coevolving entities. Hillis' [92] seminal work on coevolving a population of sorting networks against a population of input vectors gives a compelling illustration of how these processes may interact. Hillis recognizes that the feedback loops between the processes of gradient creation and gradient following can behave like an optimizer with a dynamically adjustable evaluation function; by judiciously selecting the range of test cases applied to the sorting networks, coevolution can make more effective use of finite computational resources to achieve better results.

Nevertheless, the ability to dynamically alter a learning landscape does not by itself guarantee that coevolution will lead to effective learning. Indeed, conventional coevolutionary methods are known frequently to exhibit a number of irksome modes of behavior that hinder learning. Intransitive superiority relationships [36] and mediocre-stable states [164] are two examples.

Our experiments involve the coevolution between cellular automata (CA) rules for a density classification task (the *majority function*) and initial condition densities. We choose this problem because an extensive literature on the majority function exists (e.g., [152, 204, 6, 119, 142]) and the best rules currently known derive from coevolution (by Juillé and Pollack, with success rates of 85.1%, 86.0% [105], and 86.3% [104]), allowing us to more meaningfully ascertain the significance of our results. We are able to discover a rule that has a competitive success rate of 84.0%. While not superior to the results of Juillé and Pollack, our rule is significantly better than all results published elsewhere, and we anticipate improved results, as we discuss below.

More importantly, we argue that our method offers a potentially more general and comprehensive approach towards coevolution.

This chapter is organized as follows: Section 7.2 details how we employ the Pareto optimality criterion in our coevolutionary algorithm; Section 7.3 describes the majority function and how we deploy our Pareto coevolutionary algorithm to work on it; Section 7.4 gives additional details of our experiment setup; Section 7.5 reviews results; Section 7.6 discusses plans for new experiments and concludes. The work in this chapter is first published in Ficici and Pollack [64].

7.2 Pareto Coevolution Methodology

7.2.1 Learning from Teaching

We equate the processes of gradient following and gradient creation, discussed above, with the roles of *learning* and *teaching*, respectively. Every agent with which a learner interacts is a teacher, every agent with which a teacher interacts is a learner. An agent in a coevolutionary framework usually, though not always, plays both roles simultaneously; and, when both roles are played by an agent, they are usually not performed with equal success. Our approach to coevolution considers the roles of learning and teaching to be orthogonal.

The only evidence that a learner has to indicate success at following gradient comes from achieving good outcomes (high payoffs) through interaction with agents (teachers). But, the learning gradient is typically of very high dimension in coevolution, since each agent with which a learner can interact represents a dimension to be optimized. Therefore, some principled method of integrating this space is required, and this is the reason we look towards the Pareto optimality concept for help.

We define the role of the teacher to be one that is awarded fitness for providing gradient for learners. But, what evidence can we gather to infer success at this job? We define the *learnability* of a teacher, with respect to a particular learner, to be the probability that the learner can be transformed, over some number of variation steps, to become competent (or more competent) at the task posed by the teacher. (Note that learnability becomes sensitive to the state of an entire population if recombinative methods are used in variation.) Thus, the task posed by a teacher is unlikely to be learned if the learner is too remote from regions in variation space where learners are competent at the task—the teacher is "too difficult." Teachers that are completely mastered by a learner also have a low learnability in the sense that the learner has no chance of improving its performance on the task, since it is already perfect.

Rather than try to measure teacher learnability directly, our approach is to discover teachers that can demonstrate *gaps* in learner competence—i.e., show one learner to be more proficient than another at a particular task. We operate on the intuition that teachers that fill competence gaps are likely to be learnable because they expose and explore pre-existing gradients of learner ability in different dimensions of behavior. We rely on the variation process to open new dimensions. This way, we can hope always to have relevant challenges that are of appropriate difficulty. Note that, if an evolving population contains a learner (call it L^*) that is superior to all other learners in the population with respect to every teacher, then no competence gaps exist "above" L^* to fill with teachers that are certain to challenge it. We must wait until some variation occurs to generate a new learner that outperforms L^* in some dimension(s) and creates new gaps in competence. This approach to creating and maintaining gradient for coevolutionary learning is substantially different from those of Rosin [172], Juillé [101], Olsson [153], and Paredis [162].

7.2.2 Learning: Following Gradient

This section describes how we measure success at following a high-dimensional gradient using the Pareto optimality concept. We name the set of learners $\mathcal{R}$ and the set of teachers $\mathcal{S}$. The payoff matrix $\mathbf{G}$ describes the performance of all learners against all teachers, where $\mathbf{G}_{i,j}$ is the payoff earned by learner i when interacting with teacher j. The solution concept for our learners (i.e., our primary search problem) is defined as follows.

Definition 3 (Pareto Dominance) *Learner x* Pareto dominates *learner y with respect to the set of teachers $\mathcal{S}$, denoted as $x \overset{\mathcal{S}}{\succ} y$, iff $\forall w \in \mathcal{S} : \mathbf{G}_{x,w} \geq \mathbf{G}_{y,w} \quad \wedge$ $\exists v \in \mathcal{S} : \mathbf{G}_{x,v} > \mathbf{G}_{y,v}$.*

That is, learner x dominates learner y if and only if x does at least as well as y with respect to all teachers in $\mathcal{S}$ and does strictly better than y with respect to at least one teacher.

Definition 4 (Mutual Non-Domination) *Learners x and y are* mutually non-dominating, *denoted as $x \overset{\mathcal{S}}{\simeq} y$, iff $\exists w, v \in \mathcal{S} : \mathbf{G}_{x,w} > \mathbf{G}_{y,w} \quad \wedge \quad \mathbf{G}_{x,v} < \mathbf{G}_{y,v}$.*

That is, learners x and y mutually non-dominate if there simultaneously exists a teacher in $\mathcal{S}$ with respect to which x out-performs y and another teacher with respect to which y out-performs x.

Definition 5 (Pareto Front) *The* Pareto front *of a set of learners $\mathcal{R}$, denoted as $F^0(\mathcal{R})$, is the subset of all non-dominated learners in $\mathcal{R}$ (with respect to the set of teachers $\mathcal{S}$); that is, $F^0(\mathcal{R}) = \{x \in \mathcal{R} : \neg\exists w \in \mathcal{R},\ w \overset{\mathcal{S}}{\succ} x\}$.*

Definition 6 (Dominated Subset) *The* dominated subset *of $\mathcal{R}$, denoted as $D^0(\mathcal{R})$, is the subset of learners that are dominated by some learner in $\mathcal{R}$ (with respect to the set of teachers $\mathcal{S}$); that is, $D^0(\mathcal{R}) = \{x \in \mathcal{R} : \exists w \in \mathcal{R},\ w \overset{\mathcal{S}}{\succ} x\}$.*

Note that a learner belongs exclusively either to the front or the dominated set: $F^0(\mathcal{R}) \cap D^0(\mathcal{R}) = \emptyset$ and $F^0(\mathcal{R}) \cup D^0(\mathcal{R}) = \mathcal{R}$.

Once we identify the Pareto front F^0 and the set of dominated learners D^0, we may compute the next *Pareto layer*, $F^1(\mathcal{R}) = F^0(D^0(\mathcal{R}))$—the set of learners that are non-dominated once we exclude the Pareto front. We may continue this process until every learner is understood to belong to a particular Pareto layer; this recursive process is illustrated in Figure 7.1. Pareto layers indicate both generality and uniqueness in learner competence: Every learner in F^n is less broad in competence than some learner in F^{n-1}, and every learner in F^n can do something better than some other learner in F^n.

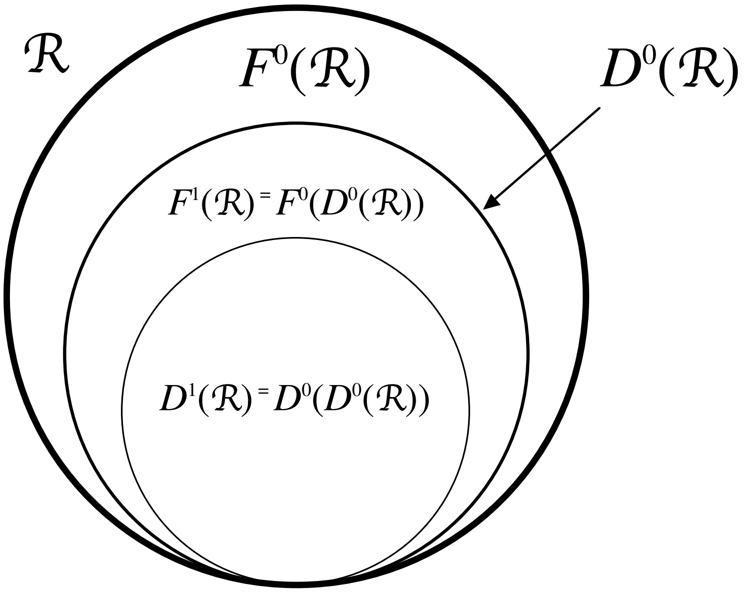


Figure 7.1: Construction of Pareto layers.

Intra-Layer Ranking

High-dimensional spaces of competing objectives are known to cause problems for Pareto ranking in ordinary (non-coevolutionary) EAs [76]. As we note above, a large number of teachers gives a high-dimensional gradient. This creates potentially many ways in which a learner may excel and earn a place in a particular Pareto layer. Indeed, we find in our experiments that the number of learners in F^0 tends to increase over evolutionary time and may ultimately include as much as 75% of the entire population.

We therefore require some form of *intra-layer ranking* to differentiate learners within a particular (possibly crowded) layer. We consider two approaches, both stemming from diversity-maintenance techniques, and find them to behave similarly (the experiments we report in this chapter use the second approach). Our first approach is similar to Juillé and Pollack's [103] *competitive evolution paradigm.* In comparing two learners from the same layer, we give each learner a point for each dimension (teacher) in which it out-scores the other learner. (Alternatively, if learners x and y out-score each other in n_x and n_y dimensions, respectively, then the better of the two agents gets $|n_x - n_y|$ points and the other gets 0 points.) We accumulate points over all pair-wise comparisons of learners *that belong to the same layer*, and then rank learners according to the sums. Our second approach applies Rosin's [172] *competitive fitness sharing* method within each layer. The learners within a layer are then ranked with respect to each other according to the results of the fitness sharing. Regardless of which intra-layer ranking approach is used, *inter*-layer ranking is achieved by giving the highest-ranked learner(s) of Pareto layer F^n a global rank just below the lowest-ranked learner(s) of Pareto layer F^{n-1}.

7.2.3 Teaching: Creating Gradient

This section describes how we measure success at creating gradient for learning. We begin with the m by n payoff matrix $\mathbf{G}$, where m is the number of learners and n is the number of teachers. Matrix entry $\mathbf{G}_{i,j}$ is the payoff received by learner i when it interacts with teacher j. If learner x performs better than learner y with respect to teacher j (i.e., $\mathbf{G}_{x,j} > \mathbf{G}_{y,j}$), denoted $x \overset{j}{>} y$, then we say that teacher j *distinguishes* the learner pair (x, y) in favor of x. By causing a pair of learners to receive different payoffs, a teacher exposes a gap in the proficiencies of the two learners. We are interested to identify all such proficiency gaps in the population of learners, as made apparent by the population of teachers.

To identify all learner pairs that are distinguished by each teacher, we construct a new n by $m^2 - m$ matrix $\mathbf{M}$. Each column of this matrix corresponds to a particular pair-wise comparison of learners across all n teachers. We exclude self-comparisons, since there cannot exist any proficiency gaps between a learner and itself, and we treat the learner pairs (A, B) and (B, A) as distinct—(A, B) is reserved for teachers that distinguish A and B in favor of A, while (B, A) is for teachers that distinguish in favor of B. The matrix entry $\mathbf{M}_{j,k}$ equals one if teacher j distinguishes the learners in pair $k = (x, y)$ in favor of x. Clearly, if entry $\mathbf{M}_{j,k}$ is non-zero, then entry $\mathbf{M}_{j,k'}$ must be zero, where $k = (x, y)$ and $k' = (y, x)$.

$$\mathbf{G} = \begin{array}{c|ccc} & \alpha & \beta & \gamma \\ \hline \mathrm{A} & 1 & 1 & 3 \\ \mathrm{B} & 2 & 3 & 2 \\ \mathrm{C} & 1 & 2 & 1 \end{array} \qquad \mathbf{M} = \begin{array}{c|cccccc} & (\mathrm{A},\mathrm{B}) & (\mathrm{A},\mathrm{C}) & (\mathrm{B},\mathrm{A}) & (\mathrm{B},\mathrm{C}) & (\mathrm{C},\mathrm{A}) & (\mathrm{C},\mathrm{B}) \\ \hline \alpha & 0 & 0 & 1 & 1 & 0 & 0 \\ \beta & 0 & 0 & 1 & 1 & 1 & 0 \\ \gamma & 1 & 1 & 0 & 1 & 0 & 0 \end{array} \tag{7.1}$$

Once we obtain matrix $\mathbf{M}$, we have identified all the ways in which each teacher demonstrates utility as a touchstone of learner competence. But, how shall we use this information to create a selective pressure for teachers? To begin with, a teacher that fails to reveal any variation in learner ability is dubious.

One approach is to give all teachers that distinguish at least one learner pair a score of one, and then divide each teacher's score by the number teachers with an identical profile (to better maintain diversity). But, we feel this method to be too coarse-grained. Our compromise approach is to perform fitness sharing much like Rosin [172]. (Subsequent work by Bucci and Pollack [29] and de Jong and Pollack [45] applies the Pareto criterion to teachers to obtain their selective pressure, as well. In this case, in Equation 7.1, we would prefer teacher β to teacher α because β makes every distinction that α does, and more.)

Equation 7.2 shows that the score received by teacher j is the sum, across all learner pairs distinguished by it, of the *value* of a learner pair divided by the pair's *discount factor*. The discount factor of a pair is the total number of teachers that distinguish it. For the value of a pair, we experiment with two possibilities. Our first approach (Equation 7.3, left) simply gives every pair an equal value ($v_k = 1$). Our second approach (Equation 7.3, right) recognizes certain pairs as more significant than others. For example, we may argue that a teacher should get more reward for distinguishing between two good learners than for distinguishing between two poor ones. Further, a teacher should be rewarded more for showing a generally good learner to do something worse than a generally poor learner than the other way around. Therefore, our second approach assigns the value of a learner pair $k = (x, y)$ to be the fitness of the loser y.

$$s_j = \sum_k \mathbf{M}_{j,k} \frac{v_k}{d_k} \qquad d_k = \sum_i \mathbf{M}_{i,k} \tag{7.2}$$

$$v_k = 1 \qquad \textbf{OR} \qquad v_k = \text{fitness}(y), \text{where } k = (x, y) \tag{7.3}$$

7.2.4 Pareto Optimality and Intransitivity

Here we briefly review some properties of our solution concepts, Pareto optimality and distinctions, with respect to intransitive cycles. In Chapter 3, we show that intransitive structures cause conventional coevolutionary algorithms to exhibit oscillations and forgetting; in particular, we discuss the emergence of oscillatory dynamics in the majority-function domain. We revisit our canonical examples of intransitivity, Rocks-Paper-Scissors and Matching Pennies.

Equation 7.4 gives the payoff matrix for the RPS game and the corresponding teaching matrix $\mathbf{M}$. The RPS game is a symmetric zero-sum game. We can easily see that each of the three strategies must be in the non-dominated front F^0: Each strategy out-performs the other two with respect to at least one strategy. For example, R out-performs P and S with respect to S, and R out-performs S with respect to R. The teaching matrix reveals that each teacher distinguishes three learner pairs. Of these three, two are distinguished by some other teacher and one is not. Thus, all three strategies receive identical rewards in their roles as learners and teachers. The asymmetric zero-sum game of matching pennies yields similar results: Both strategies (heads and tails) receive equal reward. This is in contrast to the oscillatory dynamic that arises in the conventional coevolutionary algorithm.

$$\mathbf{G} = \begin{array}{c|ccc} & \mathrm{R} & \mathrm{P} & \mathrm{S} \\ \hline \mathrm{R} & 0 & -1 & 1 \\ \mathrm{P} & 1 & 0 & -1 \\ \mathrm{S} & -1 & 1 & 0 \end{array} \qquad \mathbf{M} = \begin{array}{c|cccccc} & (\mathrm{R},\mathrm{P}) & (\mathrm{P},\mathrm{R}) & (\mathrm{R},\mathrm{S}) & (\mathrm{S},\mathrm{R}) & (\mathrm{P},\mathrm{S}) & (\mathrm{S},\mathrm{P}) \\ \hline \mathrm{R} & 0 & 1 & 1 & 0 & 1 & 0 \\ \mathrm{P} & 0 & 1 & 0 & 1 & 0 & 1 \\ \mathrm{S} & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \tag{7.4}$$

7.3 Majority Function

7.3.1 Description

Density classification tasks are a popular area of study in cellular automata research [152, 204, 104, 105, 6, 5, 119, 142]. The objective is to construct a rule that will cause a one-dimensional, binary CA to converge, within some pre-determined number of time-steps, to a state of all ones if the percentage (or *density*) of ones in the initial condition (IC) is greater than or equal to some pre-established value, $\rho \in [0, 1]$. Otherwise, the rule should cause the CA to converge to a state of all zeros. The majority function uses a value of $\rho = 0.5$.

Though Land and Belew [119] prove that no rule correctly classifies all initial conditions, the highest possible success rate remains unknown. Currently, the best rules (of radius three, operating on a CA lattice of 149 bits) achieve success rates of 85.1%, 86.0%, and 86.3% over a uniform sampling of initial conditions, and were discovered by Juillé and Pollack [105, 104, 106] through coevolution. These rules represent a significant improvement over all earlier rules, for example ABK (82.4%), DAS (82.3%), and GKL (81.5%), as discussed in [105, 152].

7.3.2 Coevolution of Rules and Densities

Following Juillé and Pollack [105, 104], we use two-population coevolution to find CA rules of radius three that operate on a one-dimensional lattice of 149 bits. A radius of n means that lattice positions $i - n$ through $i + n$ (with wrap-around) are used by the CA rule to determine the next state of lattice position i. A radius of three gives a "window" of seven bits, meaning the rule must contain $2^7 = 128$ bits, one for each possible window state. This gives us a search space of 2^{128} possible rules.

A lattice of 149 bits has 2^{149} possible states, and therefore initial conditions. While one population evolves rules (bit-strings of length 128), the other population evolves initial condition *densities* (floating-point numbers)—not actual initial conditions.

Our experiments depart from those of Juillé and Pollack by using our Pareto coevolution methodology, outlined above. As with many two-population coevolutionary domains, ours has an intrinsic asymmetry in the difficulties faced by the two populations: the discovery of good rules is much harder than the discovery of challenging IC densities. Though the space of possible initial conditions is much larger than the space of possible rules, there exist only 150 distinct IC densities (from all-zeros to all-ones). Further, densities generally become more difficult as they approach 0.5 [142, 119], so the space in some sense approximates a unimodal landscape. For these reasons, rules are assigned only the role of learners and IC densities are assigned only the role of teachers. We can imagine other coevolutionary domains where coevolving entities would be called upon to fulfill both roles, such as Tic-Tac-Toe.

7.3.3 Derivation of Payoff Matrix: Test Case Sampling

All ranking methods potentially impose severe non-linearities by expanding small differences in performance and compressing large ones. Pareto ranking is no exception and may even be considered more extreme in some ways. Because of this non-linear behavior, the payoff matrix $\mathbf{G}$, which forms the basis of our approach, should be as accurate as possible. Thus, our approach is at least as sensitive to non-deterministic domains as conventional coevolutionary methods.

Our CA domain, however, is deterministic. Because we evolve rules against densities rather than actual initial conditions, rule performance against a particular density is expressed as the percentage of correctly classified ICs of that density class. But, we generally cannot afford to sample a particular density class exhaustively. This impedes our ability to compare rules of similar ability. Further, as rules improve in performance, we generally rely on densities closer to 0.5 to distinguish them. Yet, the distribution of initial conditions over densities is binomial, which makes our sampling of ICs significantly more sparse as densities approach 0.5 (and our estimation of rule performance less accurate).

We desire a method to extract meaningful information about rule performance with relatively few samples. Fortunately, for our Pareto coevolution methodology to work, we do not need to know precisely how well a particular rule (learner) performs against a particular IC density (teacher); we only need relative ranking information. Our solution is to once again turn to the Pareto optimality criterion. Note that our use of Pareto ranking to derive the payoff matrix (described here) is not to be confused with our use of Pareto ranking to rate learners once payoffs are known (described in Section 7.2.2).

We compute each column j of the payoff matrix $\mathbf{G}$ in the following manner. We generate some number q of initial conditions ($q = 40$ in our experiments) that are representative of density (teacher) j; all m rules (learners) are tested on this set of ICs. The test results are placed in an m by q matrix $\mathbf{H}$, where $\mathbf{H}_{u,v} = 1$ if rule u correctly classifies IC v and $\mathbf{H}_{u,v} = 0$ otherwise. We then perform Pareto ranking of rule performance based on $\mathbf{H}$. Rules on the Pareto front F^0 are given the highest payoff, those in layer F^1 the next highest, and so on. These payoffs form column j of the payoff matrix $\mathbf{G}$.

All rules cover some subset of ICs that belong to a particular density class. These subsets may overlap in any number of ways. We require that a rule dominate another in terms of *measured coverage* in order to receive a higher payoff; this is more stringent than mere comparison of *measured success rates*. For example, two rules that each cover 50% of some density class may overlap entirely on the one extreme, or not at all on the other extreme. Regardless of the amount of actual overlap, the chance that our measured coverage will indicate one rule to dominate the other is extremely small, even though the measured success rates (number of ICs solved by the two rules) will very likely be unequal. But, as the actual performance rates of two rules grow apart, the more likely it becomes for the better of the two to dominate the other in measured coverage (especially if actual coverage of the stronger rule overlaps heavily with that of the other).

Our method of test-case sampling removes the need for an explicit similarity metric—similarity is guaranteed by the process of generating multiple ICs from a density value. Juillé and Pollack [104, 105] require a similarity metric to cluster IC densities so that rule performance can be gauged with respect to density.

7.4 Experimental Setup

A rule is given 320 time-steps to converge the CA to the correct state. The sizes of our populations of rules and IC densities are $N_R = 150$ and $N_{IC} = 100$, respectively. All rules are tested against all densities in every generation. Each density is sampled 40 times, giving a total of 6×10^5 evaluations per generation.

The initial population of rules is composed of random bit-strings, distributed uniformly over the range of string densities (0 to 128 ones). The initial population of IC densities is distributed uniformly over the interval [0, 1]. Rules are varied by one-point crossover and a 2% per-bit mutation rate. IC densities are varied by the addition of Gaussian noise of zero mean and standard deviation of 0.05.

Rule ranks are squared before they are normalized for the roulette wheel. The next generation of rules is created by using Baker's [20] SUS method to select 75 rules that remain unaltered and another 75 rules to which the variation operators are applied. The next generation of IC densities is created by using SUS to select 50 densities that remain unaltered and another 25 that are varied. The remaining 25 densities are picked at random from a uniform distribution.

An IC density is converted to an actual initial condition by first adding Gaussian noise of zero mean and standard deviation of 0.05. We multiply the result by 149 and take the floor to arrive at an integer between 0 and 149. The initial condition will be a random bit-string of exactly that number of ones. All rules see the same set of initial conditions.

7.5 Results and Discussion

7.5.1 Rule Performance

We have conducted six runs of our experiment, three for each of our two methods of valuating learner pairs (see Section 7.2.3 and Equation 7.3). Our best result to-date is a rule that correctly classifies 84.0% of initial conditions, shown in Table 7.1. We determine this success rate by testing the rule against 4×10^7 randomly generated ICs (that is, a binomial distribution of IC densities). The rule took approximately 1300 generations to evolve. Two of the other runs each exceed 81% success; our worst result is 78.8% success. While our best result comes from our first method of valuating learner pairs (see Equation 7.3), the data do not distinguish the performance of the two methods.

Figures 7.2 through 7.5 show four example runs of the rule in Table 7.1. The first two runs are each examples of correct classification. Notable in these examples is that the density of the cellular automaton lattice actually crosses the 0.5 threshold; since

Bits 0–31	00010000	01010011	00000000	11010010
Bits 32–63	00000000	01010001	00001111	01011011
Bits 64–95	00011111	01010011	11111111	11011111
Bits 96–127	00001111	01010101	11001111	01011111

Table 7.1: Currently best evolved rule using Pareto coevolution; achieves 84.0% classification success. Bit i corresponds to the rule's action when the binary integer interpretation of a window equals i. (Rule bits are read left-to-right.)

the classification is correct, the number of crosses must be even. All the lattice states we visit on the opposite side (from where we begin) of the 0.5 threshold represent initial conditions that the rule will incorrectly classify. In contrast to the first two examples, Figure 7.4 shows are run where lattice density monotonically increases. Finally, Figure 7.5 shows a run in which the rule incorrectly classifies the initial condition.

7.5.2 Dynamics of Teacher Population

Figure 7.6 shows four histograms of the IC population over time for a sample run (unfortunately, we do not have this data for the run from which our best rule originates). As we specify in Section 7.4, the distribution of initial conditions in Generation 0 is uniform. As early as Generation 5, we find the distribution to heavily emphasize density extrema; only five of the 100 individuals in the IC population fall in the interval $[0.4, 0.6]$. By Generation 50, low and high density ICs are not longer emphasized; instead, we have peaks approximately at 0.4 and 0.6 density, with the immediate neighborhood around 0.5 still being rather sparse (six individuals in the interval $[0.47, 0.53]$ and thirteen in the interval $[0.45, 0.55]$). In Generation 250, the distribution is heavily concentrated (60% of ICs) in the interval $[0.4, 0.6]$; 14 individuals are now found in the interval $[0.47, 0.53]$ and 25 in the interval $[0.45, 0.55]$. Thus, we see a clear movement from uniform testing to the easiest densities, then to ICs of

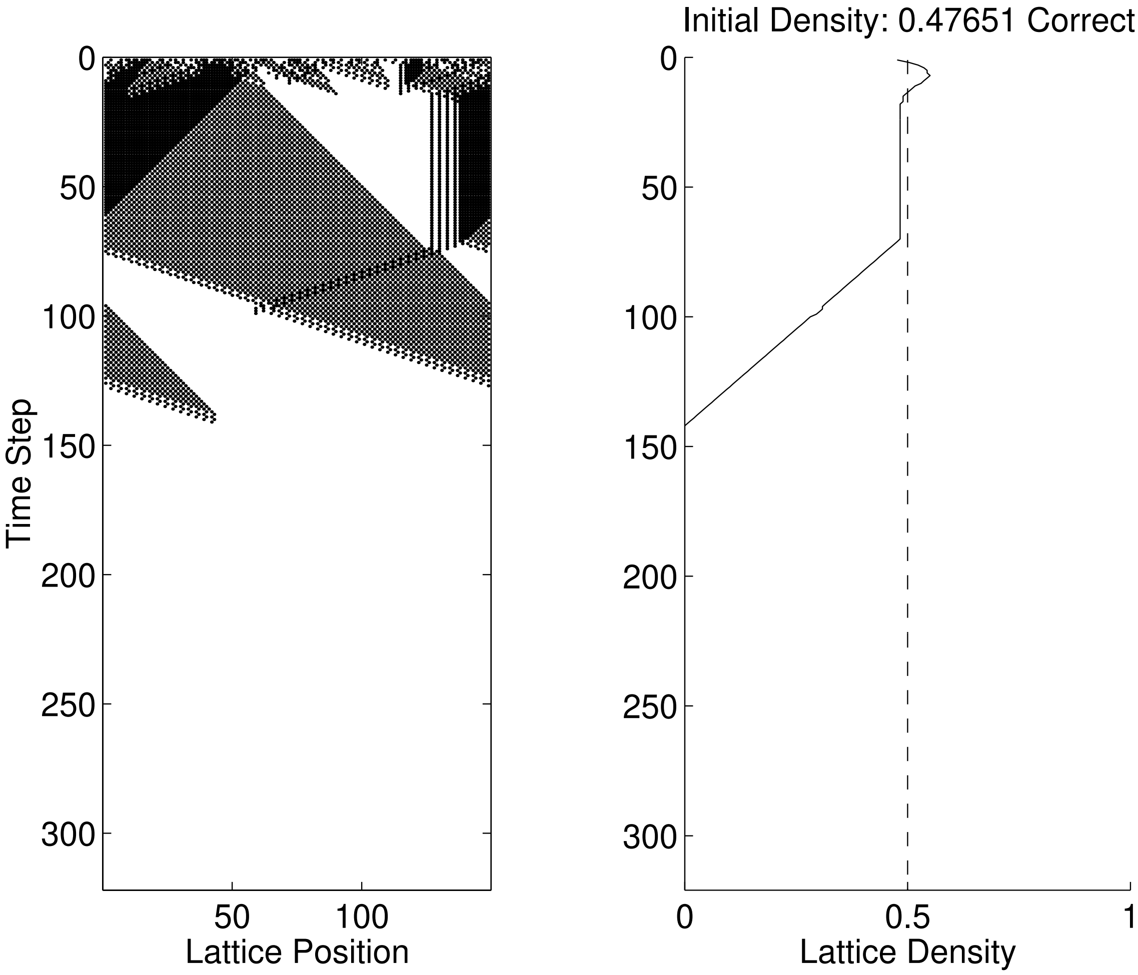


Figure 7.2: First example run of rule in Table 7.1. Left graph shows the time evolution of the cellular automaton lattice; the right graph shows the density of the lattice over time.

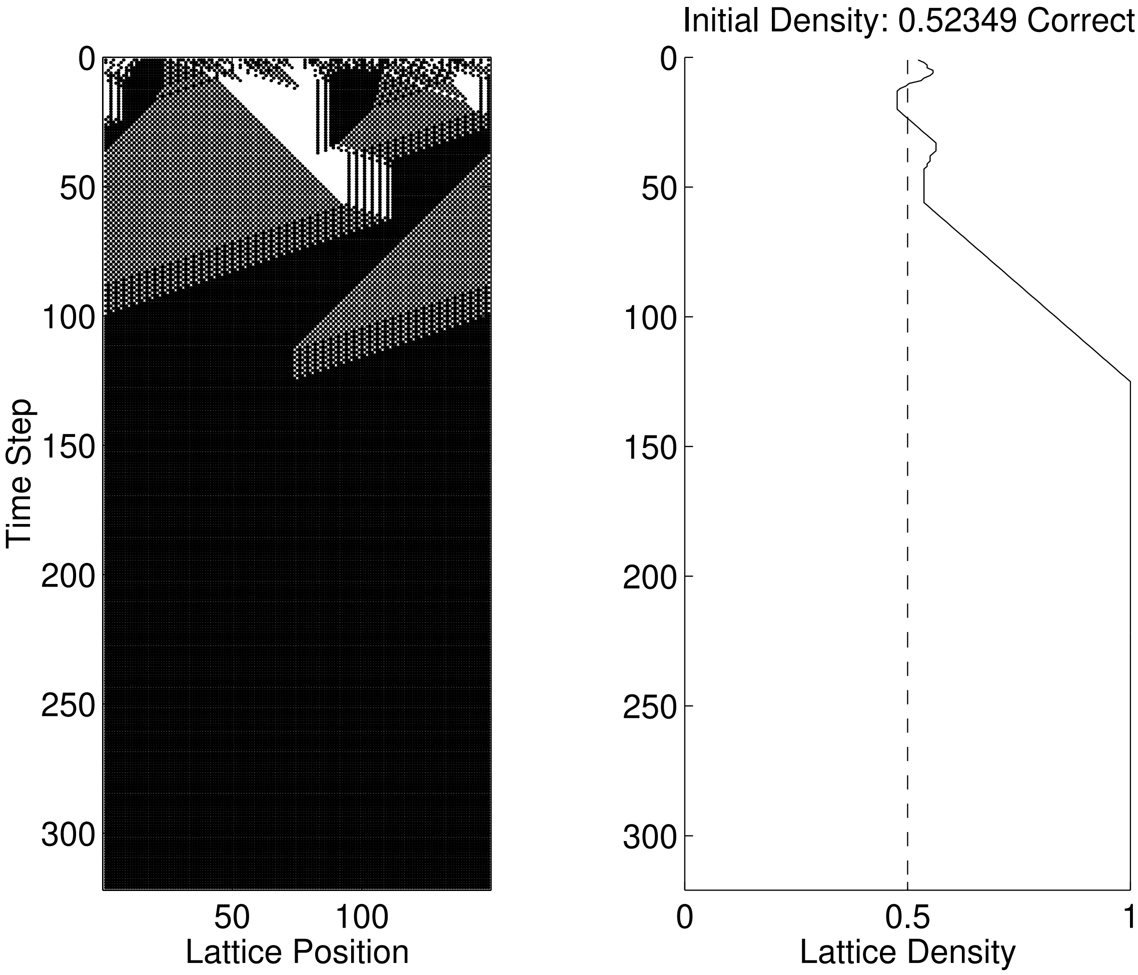


Figure 7.3: Second example run of rule in Table 7.1. Left graph shows the time evolution of the cellular automaton lattice; the right graph shows the density of the lattice over time.

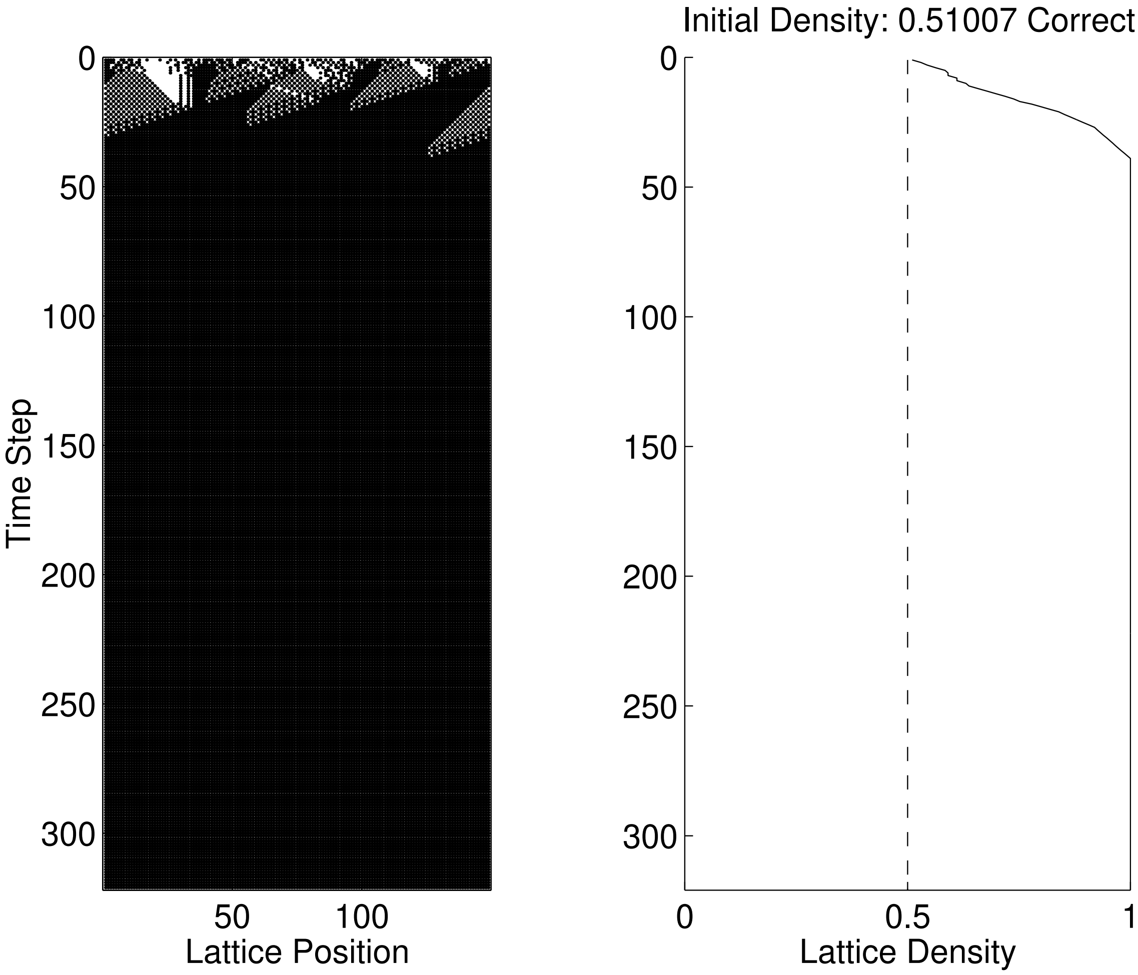


Figure 7.4: Third example run of rule in Table 7.1. Left graph shows the time evolution of the cellular automaton lattice; the right graph shows the density of the lattice over time.

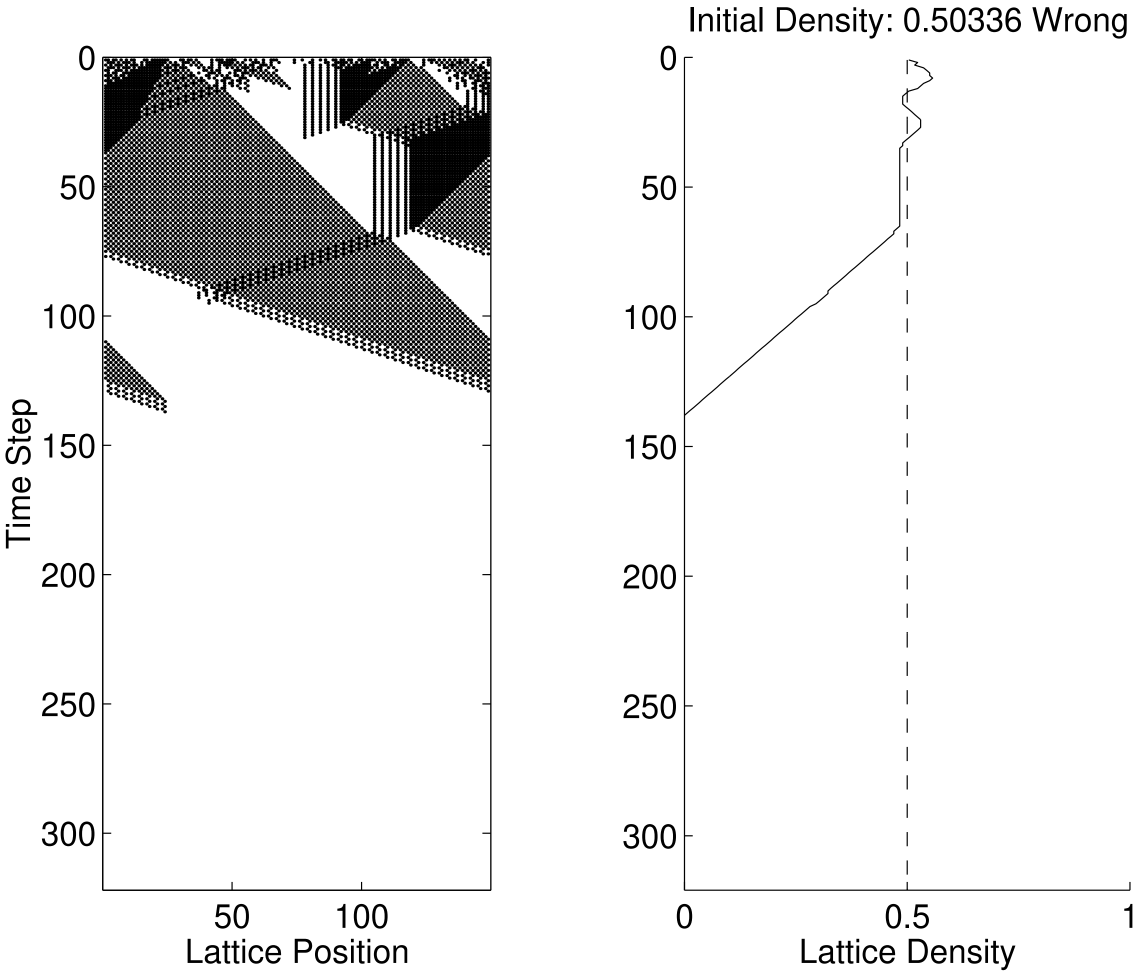


Figure 7.5: Fourth example run of rule in Table 7.1. Left graph shows the time evolution of the cellular automaton lattice; the right graph shows the density of the lattice over time.

moderate difficulty, and then to the most difficult ICs. Along with the quality of the rules we are able to obtain, these data suggest that our notion of distinctions is able to capture some meaningful information about where testing effort should concentrate.

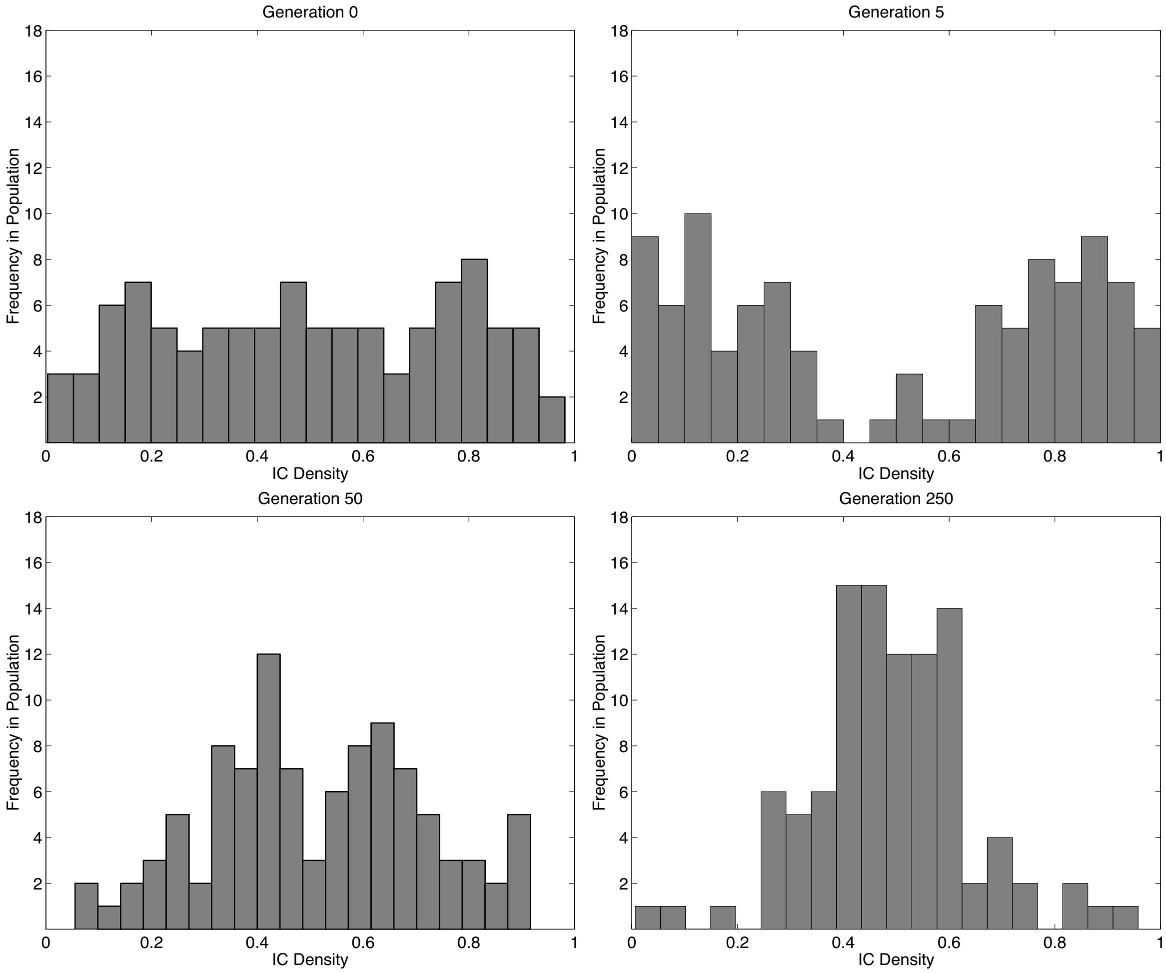


Figure 7.6: Histograms of initial condition densities for a sample run at Generations 0 (top left), 5 (top right), 50 (bottom left), and 250 (bottom right).

7.5.3 Comparison with Juillé and Pollack

In experiments where $N_R = N_{IC} = 400$ [105], Juillé and Pollack discover a rule that achieves 85.1% success (some runs give $\leq 76\%$). Their best results (86.0% and 86.3%

success) are discovered in experiments where $N_R = N_{IC} = 1000$ [105, 104] (experiments last 5000 generations; all exceed 82.0% success). They test all rules against all densities, but each density is sampled only once. In contrast, our experiments use much smaller population sizes ($N_R = 150, N_{IC} = 100$), but we sample each density 40 times. Thus, the total amount of computation in our experiments falls in between those of Juillé and Pollack. But, in exchange for a more expensive IC density sampling procedure (see Section 7.3.3), we avoid the explicit similarity metric required by Juillé and Pollack to classify ICs, and thereby arrive at an approach that should more easily generalize to other problem domains (e.g., sorting networks). Indeed, we intend ultimately to apply our Pareto coevolution methodology to variable-sum games, in addition to zero-sum games such as those studied by Rosin [172] and Juillé [101].

While we do not improve upon the results of Juillé and Pollack [105, 104], we improve significantly upon all results published elsewhere. The next most effective rule is by Andre, et al [6, 5], which performs at 82.4%. This rule was discovered with genetic programming in experiments using a rule population size of 51,200, each tested on 1000 different ICs (ICs were not coevolved) per generation. We believe that we can improve our results with larger populations.

7.6 Conclusions

We build a novel coevolutionary algorithm based upon the principle of Pareto optimality that implements distinct solution concepts for the primary (learner) and secondary (teacher) search efforts. Our algorithm distinguishes the role of the gradient follower from that of the gradient creator, even though both may coexist within the same agent. We use our algorithm to coevolve cellular automata rules for the majority function and discover a rule that correctly classifies 84.0% of initial conditions. Additional experiments to improve our result will consider larger populations and perhaps different inter-generational replacement schemes. Of more interest to our Pareto approach, we require control experiments to identify the contributions of each component of our overall methodology. In these controls, we will substitute one or two of our methods (for rewarding learners and teachers, and for sampling densities) with more conventional mechanisms, for example using a problem-specific similarity metric instead of our more expensive sampling method.

Despite our encouraging result, this chapter does little more than suggest an interesting line of research. Specifically, we offer little formal (and indeed empirical) justification for our approach; this work is ongoing. Nevertheless, a more formal account of our Pareto coevolution methodology can be found in the follow-up work of Bucci and Pollack [29] and de Jong and Pollack [45].

Finally, the work of Dolin, Bennett, and Rieffel [53] empirically compares our method to evaluate teachers with Rosin's shared sampling method in a variety of problems. Using genetic programs as the evolutionary substrate, they test the two methods on three symbolic regression problems; we reproduce the results in Table 7.2. At worst, our method of distinctions requires approximately 2.2 times more evaluations than shared sampling; at best, our method requires slightly less than one third the number of evaluations. They next look at a robotic control task, the results of which we reproduce in Table 7.3. Here, our method requires approximately 3.2 times the number of fitness evaluations as shared sampling. To discover a control program that correctly solves $\leq$ 70% of out-of-sample test cases (i.e., tests not used during learning), the distinction method requires approximately twice as many fitness evaluations as random test-case sampling; the distinction method out-performs random sampling for higher out-of-sample thresholds.

Problem	Shared Sampling	Distinctions	Random Test-Case Sampling
1	33,840,000	50,700,000	600,600,000
2	334,800,000	100,440,000	not solved
3	21,120,000	47,040,000	56,550,000

Table 7.2: Experimental data from Dolin, et al. Minimal number of fitness evaluations required (over 120 trials) to solve regression problem with probability 0.9. Data reproduced from Dolin, et al, [53].

Percentage Solved	Shared Sampling	Distinctions	Random Test-Case Sampling
0.60	850,000	1,470,000	700,000
0.70	1,275,000	2,970,000	1,410,000
0.80	5,390,000	17,325,000	21,780,000
0.82	18,810,000	60,520,000	not solved

Table 7.3: Experimental data from Dolin, et al. Minimal number of fitness evaluations required (over 120 trials) to solve given percentage of out-of-sample tests with probability 0.9. Data reproduced from Dolin, et al, [53].

Chapter 8

A Game-Theoretic Memory Mechanism

8.1 Introduction

Among the problems one often confronts when using a coevolutionary algorithm is that of *forgetting*, where one or more previously acquired traits (i.e., components of behavior) are lost only to be needed later, and so must be re-learnt. Since selection pressure determines which traits have the opportunity to reproduce, the disappearance of a trait has but a few explanations. First, a trait is selected *against* when individuals with that trait are less fit, on average, than individuals without it. Second, a trait is subject to *drift* when individuals with that trait are equally fit, on average, as individuals without it. This drift may occur in either of two ways (or both): Due to sampling error in the population dynamics (where fewer of the individuals with the trait actually reproduce, thus reducing their numbers and increasing the risk of extinction for that trait), and due to variational biases that cause the trait to be lost in the generation of offspring. (Even if a trait is selected for, the variational process may be strongly biased against it, thus causing offspring to lack the trait; such variational bias is the impetus behind the technique of *elitism* that is often used in evolutionary algorithms [85].) The disappearance of a trait becomes an instance of forgetting if the discarded trait is subsequently able to contribute *positively* to the fitness of an individual lacking that trait.

Such variability in a trait's contribution to fitness is possible in coevolution because the observed quality of an individual (and the traits it carries) may be highly contingent upon the context of its evaluation—the population of coevolving individuals. Thus, at one moment in evolutionary time, a trait may be highly undesirable (or of neutral worth but difficult to maintain) and purged, only to become of value at some later point in time. Simple zero-sum games with intransitivities illustrate how contingency can cause a coevolutionary system to learn, forget, and re-learn traits in a cyclic fashion; the familiar Rock-Paper-Scissors game is the canonical example of an intransitive cycle (Rock beats Scissors, which beats Paper, which in turn beats Rock), where one strategy exploits the next, leading to a perpetual alternation of selective pressure for and then against each strategy in turn.

The problem of designing a heuristic "memory mechanism" to prevent forgetting intrinsically entails the problem of deciding what it is that we are trying to "remember"—what is our *solution concept*? That is, what collection of traits constitutes the desired or "correct" set, and what properties does this collection have? A principled answer to this question is imperative when a domain forces mutual exclusivity between certain traits, or when an evolutionary representation (genome) cannot simultaneously encode all desired traits. Once we have our solution concept, what organizing principle do we use in the memory mechanism to obtain it? Thus, we view a memory mechanism as an accumulator of traits; a trait enters and remains in the memory only if it is "worth" remembering, according to our organizing principle.

In this chapter, we obtain our solution concept—Nash equilibrium—from game theory, and use game theory to construct an organizing principle for a new memory mechanism for coevolution. Using Watson and Pollack's *intransitive numbers game* [202], which is rife with intransitive cycles, we demonstrate the ability of our "Nash memory" mechanism to approximate a monotonic and asymptotic convergence to the Nash equilibrium of the game. Further, we show that the commonly used "best of generation" memory mechanisms (e.g., Rosin and Belew's *Hall of Fame* [175]), as well as Stanley and Miikkulainen's more recently proposed *Dominance Tournament* [187] mechanism, do not generally accumulate a collection of traits that corresponds to Nash equilibrium.

The chapter is organized as follows. Section 8.2 reviews key concepts from game theory. Section 8.3 continues with additional concepts such as *security* and *domination*. Section 8.4 then reviews the literature on methods to detect and prevent forgetting. Section 8.5 details the construction and operation of our Nash memory mechanism. Section 8.6 defines and discusses the properties of the intransitive numbers game. Section 8.7 presents our experimental setup and results. Section 8.8 offers concluding remarks. The work in this chapter is first published in Ficici and Pollack [65].

8.2 Game Theory Fundamentals

Here we present some fundamental points on game theory [83]. For simplicity, we restrict our discussion to symmetric zero-sum games (in strategic form) for two players; nevertheless, our Nash memory mechanism can be generalized to asymmetric constant-sum games for two players. (We note that variable-sum games easily complicate matters. For example, a symmetric coordination game has a single mixed-strategy Nash equilibrium. We can use this strategy with indifference to the other player's actions and get a guaranteed minimal payoff; in this sense, the mixed Nash is similar to one you might see in a zero-sum game, such as Rock-Paper-Scissors. Unlike RPS, however, each pure strategy of a coordination game is a pure Nash equilibrium strategy that *Pareto dominates* the mixed Nash strategy—that is, both players are better-off if they play the same pure strategy than they would be if one or both played the Nash mixture. To obtain these superior payoffs, the players must coordinate.)

Being symmetric, the game $\mathbf{G}$ defines a single set of *pure strategies*, $\mathcal{S}$, that is made available to both players. A player may adopt a pure strategy $s \in \mathcal{S}$ or may instead play a *mixed strategy*, which is specified by a probability distribution over $\mathcal{S}$. Pure strategies played with non-zero probability by a mixed strategy m are *in support* of (also known as "carriers" of) m. The function $\mathrm{C}(m)$ returns the set of pure strategies that support the mixture m; hence, $\mathrm{C}(m) = \{s \in \mathcal{S} : \Pr(s|m) > 0\}$. A pure strategy can be understood as a degenerate mixture.

For any pair of strategy choices made by the two players, the game $\mathbf{G}$ specifies the expected payoff earned by each player; by convention, $\mathrm{E}(\alpha, \beta)$ denotes the expected payoff earned by strategy α when played against β. Since the game $\mathbf{G}$ is zero-sum, $\mathrm{E}(\alpha, \beta) + \mathrm{E}(\beta, \alpha) = 0$. This implies that $\mathrm{E}(\alpha, \alpha) = 0$.

Any pure or mixed strategy s^* that is its own "best response" is a *Nash equilibrium* strategy. More precisely, s^* is a Nash iff, for all mixtures m, $\mathrm{E}(s^*, s^*) \geq \mathrm{E}(m, s^*)$. That is, if one player plays s^*, then the highest payoff obtainable by the other player is received by also playing s^*. Thus, a Nash strategy guarantees that, for all mixtures m, $\mathrm{E}(s^*, m) \geq 0$. Due to this property, Nash strategies provide the maximal *security level* that can be obtained in a zero-sum game—no other solution concept can guarantee a higher expected payoff without regard to the opponent's strategy choice. This security level is also known as the *value* of the game. Finally, all games with finite $\mathcal{S}$ have at least one Nash equilibrium.

8.3 Additional Concepts

The *security set* of a mixture m is the set of pure strategies against which m earns an expected payoff greater than or equal to zero; that is, $\mathrm{S}(m) = \{s \in \mathcal{S} : \mathrm{E}(m, s) \geq 0\}$. If the security set of a mixture contains the mixture's support set, then the mixture is *support-secure*; mixture m is support-secure iff $\mathrm{S}(m) \supseteq \mathrm{C}(m)$. All pure strategies are trivially support-secure. For any pure or mixed Nash strategy s^*, $\mathrm{S}(s^*) \supseteq \mathrm{C}(s^*)$ and $\mathrm{S}(s^*) = \mathcal{S}$. The *vulnerability set* of a mixture m is the complement (with respect to $\mathcal{S}$) of the security set: $\mathrm{V}(m) = \overline{\mathrm{S}(m)}$.

One strategy α *dominates* another strategy β iff $\forall s \in \mathcal{S} : \mathrm{E}(\alpha, s) \geq \mathrm{E}(\beta, s)$ and $\exists s \in \mathcal{S} : \mathrm{E}(\alpha, s) > \mathrm{E}(\beta, s)$. By definition, for any strategy s and any Nash strategy s^*, $\mathrm{E}(s^*, s) \geq 0$; therefore, any strategy that dominates a Nash must be Nash, as well. At the same time, Nash strategies need not dominate all (or any) non-Nash strategies. We examine this point further, below.

8.4 Memory Methods and Solution Concepts

Research to prevent forgetting includes a number of *memory mechanisms* that maintain a collection of "good" individuals (according to some organizing principle) discovered over evolutionary time; the memory thus encapsulates a wider range of phenotypes than is typically found in the coevolving population at any one time. The additional phenotypic variety afforded by the memory is used to augment the evaluation process, broaden selection pressure, and thereby reduce the likelihood of forgetting.

We review a number of memory methods used in the literature in Chapter 3.

As we state in Chapter 3, deciding how to organize a memory entails deciding what kind of solution we wish to obtain. If we view the memory as a repository of the most salient traits discovered so far, then the logical extension of this view implies that we should probe the *memory* for the result (solution) obtained through coevolution, not the coevolving population. In the standard coevolutionary algorithm, the coevolving population is expected both to perform search *and* represent the result of the coevolution. But, there is no reason to believe that these two tasks are orthogonal, let alone mutually supportive; indeed, our results on fitness sharing in Chapter 5 provide evidence that these tasks may interfere with each other.

A well-designed memory mechanism relieves the coevolving population from the burden (or, at least, the requirement) of representing the solution, allowing the population to concentrate on search (to improve the solution represented by the memory). The memory mechanism we introduce in this chapter uses game theory, particularly Nash equilibrium, as an organizing principle. A important feature of the Nash concept is that it allows a solution to be a *collective* of strategies. The appeal of this property is highlighted by results of Nolfi and Floreano [148]. In the presence of intransitivity, they emphasize that the real solution they obtain through coevolution is not a single, objectively best champion strategy—none exists to find—but, rather a set of locations in genotype space that are optimally poised for easy transformation into alternative strategies that have been effective in evolutionary history. That is, when a population cannot simultaneously and stably represent all the strategies in certain intransitive cycles, the best it can do is optimally traverse the cycle. This lack of a best pure strategy, coupled with a dynamic that focuses on a particular set of pure strategies, is highly suggestive of the *mixtures* that the Nash solution concept supports.

8.5 Construction and Operation of Nash Memory

8.5.1 General Framework and Instantiation

The Nash memory mechanism we examine in this chapter is a particular instantiation of a general framework, which consists of two mutually exclusive sets of pure strategies, $\mathcal{N}$ and $\mathcal{M}$. In addition to the memory mechanism, we assume the existence of some external search heuristic $\mathcal{H}$. The set $\mathcal{N}$ is unbounded in size and defined to be the support set for a mixture that is secure at least against the elements of $\mathcal{N}$ and $\mathcal{M}$; that is, $\mathrm{S}(\mathcal{N}) \supseteq \mathcal{N} \cup \mathcal{M}$. (We use $\mathcal{N}$ to denote either the mixture's support set or the mixed strategy itself, as convenient.) The objective of $\mathcal{N}$ is to represent a mixed strategy that is optimal (secure) with respect to what the search heuristic has discovered thus far; we wish the protection afforded by $\mathcal{N}$ (ideally) to increase monotonically as search progresses, thereby forming a better and better approximation of a Nash strategy for the game $\mathbf{G}$. The purpose of set $\mathcal{M}$ is to act as an accumulator or memory; it contains pure strategies that are not currently useful for $\mathcal{N}$ but were in the past and may be again in the future. The capacity of $\mathcal{M}$, however, is finite. In the present realization of the abstract memory model, we define $\mathcal{M}$ to be simply an unordered set of pure strategies with a cardinality no greater than some specified value c, which gives the *capacity* of the memory. We can imagine that c can vary over evolutionary time, in some adaptive fashion, but we instead assume a fixed value.

8.5.2 Initialization and First Update

We initialize $\mathcal{N}$ and $\mathcal{M}$ to be the empty set. Let $\mathcal{Q}$ be a set of strategies delivered by the search heuristic to the memory mechanism; we assume $|\mathcal{Q}| < c$. The first set $\mathcal{Q}$ to arrive updates $\mathcal{N}$ such that $\mathrm{C}(\mathcal{N}) \subseteq \mathcal{Q}$ and $\mathrm{S}(\mathcal{N}) \supseteq \mathcal{Q}$; the set $\mathcal{M}$ is updated to contain those elements of $\mathcal{Q}$ not in support of $\mathcal{N}$. For subsequent values of $\mathcal{Q}$, we must test the elements of $\mathcal{Q}$ against $\mathcal{N}$ to see if $\mathcal{N}$ and $\mathcal{M}$ require updating.

8.5.3 Testing $\mathcal{N}$

To verify that a mixed strategy is Nash, one need only check that the mixed strategy is secure against all pure strategies; testing against all possible mixtures (of which there is an uncountable infinity) is not required [192]. Consequently, if $\mathrm{E}(q, \mathcal{N}) > 0$ for any $q \in \mathcal{Q}$, then $\mathcal{N}$ is demonstrably not a Nash strategy and we attempt to improve our approximation; otherwise, we leave $\mathcal{N}$ and $\mathcal{M}$ undisturbed and wait for the search heuristic to deliver a new set of strategies. We compute the value $\mathrm{E}(q, \mathcal{N})$ by playing q against each strategy in support of $\mathcal{N}$ and taking a weighted average of outcomes, the weights being the probability distribution for the mixture: $\mathrm{E}(q, \mathcal{N}) = \sum_{n \in \mathrm{C}(\mathcal{N})} \mathrm{Pr}(n|\mathcal{N})\mathrm{E}(q, n)$.

8.5.4 Updating $\mathcal{N}$ and $\mathcal{M}$

We define the set $\mathcal{W} = \{q \in \mathcal{Q} : \mathrm{E}(q, \mathcal{N}) > 0\}$, that is, the "winners" from $\mathcal{Q}$. Given the pre-update values of $\mathcal{N}$ and $\mathcal{M}$, we define the post-update value $\mathcal{N}'$ such that $\mathrm{C}(\mathcal{N}') \subseteq (\mathcal{W} \cup \mathcal{N} \cup \mathcal{M})$ and $\mathrm{S}(\mathcal{N}') \supseteq (\mathcal{W} \cup \mathcal{N} \cup \mathcal{M})$; we obtain the value of $\mathcal{N}'$ with *linear programming*, which is the standard method for solving zero-sum games and for which polynomial-time algorithms exist [192]. Note that $\mathrm{S}(\mathcal{N}')$ is not necessarily a superset of $\mathrm{S}(\mathcal{N})$.

As Figure 8.1 illustrates, the post-update value $\mathcal{M}'$ contains zero or more items from each of three sources; some strategies in $\mathcal{W}$ may not be required, some strategies in $\mathcal{N}$ may be released, and some strategies may be retained from $\mathcal{M}$ (while others may be recalled from $\mathcal{M}$ to $\mathcal{N}'$). If the resultant $\mathcal{M}'$ has a cardinality $|\mathcal{M}'| > c$, then we discard items from $\mathcal{M}'$ until the capacity constraint is met. We can imagine a number of policies for removing items from $\mathcal{M}'$; we currently remove items at random first from those retained from $\mathcal{M}$, then those released from $\mathcal{N}$ (there is never the need to remove those from $\mathcal{W}$).

8.6 Intransitive Numbers Game

Watson and Pollack's *intransitive numbers game* [202] is a coevolutionary domain that is permeated by intransitive cycles. The game's geometric nature allows easy visualization of coevolutionary dynamics; we will use it to illustrate the operation of our Nash memory mechanism.

8.6.1 Definition

Each pure strategy of the intransitive numbers game is an n-dimensional vector, or point in n-dimensional space. For a pair of strategies α and β, the winning strategy is the one with higher magnitude in the dimension of least difference between the two strategies. More precisely, for two strategies α and β, the expected outcome $\mathrm{E}(\alpha, \beta)$ is:

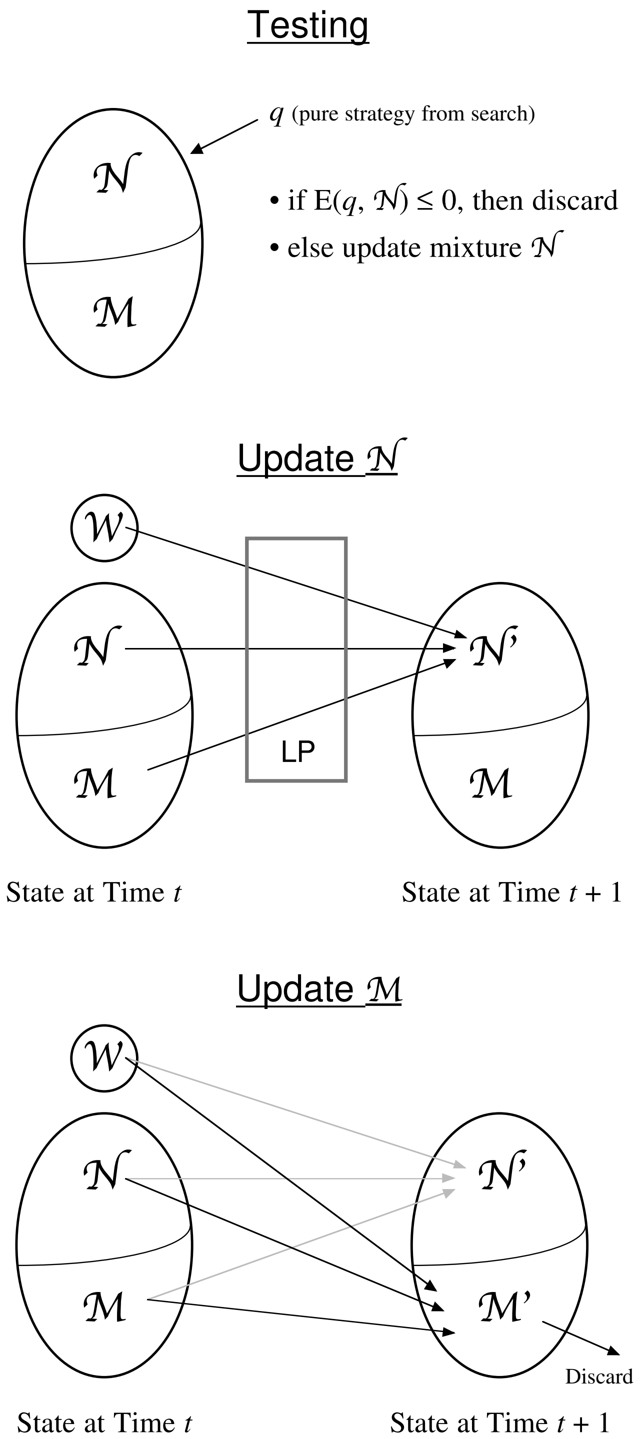


Figure 8.1: Test and update process of Nash memory.

$$\mathrm{E}(\alpha, \beta) = \begin{cases} 0 & \text{if } \min(h) = \infty \\ \mathrm{sign}(\sum_{i=1}^{n} g_i) & \text{otherwise} \end{cases} \tag{8.1}$$

$$g_i = \begin{cases} \alpha_i - \beta_i & \text{if } h_i = \min(h) \\ 0 & \text{otherwise} \end{cases} \tag{8.2}$$

$$h_i = \begin{cases} |\alpha_i - \beta_i| & \text{if } |\alpha_i - \beta_i| \geq \epsilon \\ \infty & \text{otherwise} \end{cases} \tag{8.3}$$

where ϵ is the smallest magnitude difference we wish to consider significant.

Equation 8.1 eliminates two asymmetries that exist in the original definition found in [202]: First, the game is now formally symmetric; second, no single dimension is the arbiter when the deltas in multiple dimensions are minimal. Note also that, while we use the term "expected outcome," the game is deterministic.

Figure 8.2 illustrates the intransitive numbers game in the open plane. On top, we calculate the winner of a pair-wise interaction between Strategies A and B. When we calculate the coordinate differences between the two strategies, we find the dimension of least difference to be the vertical dimension; in this dimension Strategy A has a higher magnitude than B, and so A beats B. In the middle, we show the regions in which an opponent against Strategy B can win and lose; Strategy A is clearly in a winning region. On the bottom, we add a third strategy C, which forms an intransitive cycle with Strategies A and B: A beats B, B beats C, and C beats A. Strategy C may be anywhere in the colored region and still create an intransitive cycle with Strategies A and B; this gives ample indication of how easily intransitivity is obtained in this game.

While Equation 8.1 defines the outcome between two specified strategies, it does not define the universe of strategies we are to consider, leaving the game's definition incomplete. Clearly, the strategy space may be finite or infinite, countable or uncountable; indeed, the intransitive numbers game can provide an impoverished form of open-endedness, if desired. For our purposes, we define the strategy space to be n-dimensional vectors of natural numbers, where each dimension spans the interval $[0, k]$; this yields $(k+1)^n$ distinct pure strategies.

8.6.2 Game Properties

The game defined above has a single Nash equilibrium strategy for all $\epsilon < k$ ($\epsilon = k$ implies that all strategies tie each other, making them all Nash strategies—a particularly uninteresting game). The Nash strategy is the pure strategy with value k in all n dimensions, i.e., $\langle \overbrace{k, \ldots, k}^{n} \rangle$. Of course, other definitions of the strategy space may yield multiple Nash as well as mixed Nash strategies.

As with all Nash equilibria in symmetric zero-sum games, the Nash strategy of our game does not lose to any other strategy. With $\epsilon = 1$, the Nash ties only itself and beats all others; thus, the Nash also dominates all other strategies. Here, the solution concepts of Nash equilibrium and domination agree.

With $\epsilon = 0$, however, the solution concepts point to very different outcomes. In this case, the Nash additionally ties all strategies that have the value k in any dimension. (The strategies that tie the Nash are those on the surface of the n-dimensional space. Thus, as n grows, the percentage of tieing strategies asymptotically approaches 100%, even though the number of losing (i.e., interior) strategies grows exponentially. Nevertheless, for $k = 100$ and $n = 10$, the percentage of tieing strategies is still $< 10\%$. Thus, concern over the asymptotic growth of tieing strategies must keep in mind the values of k and n.)

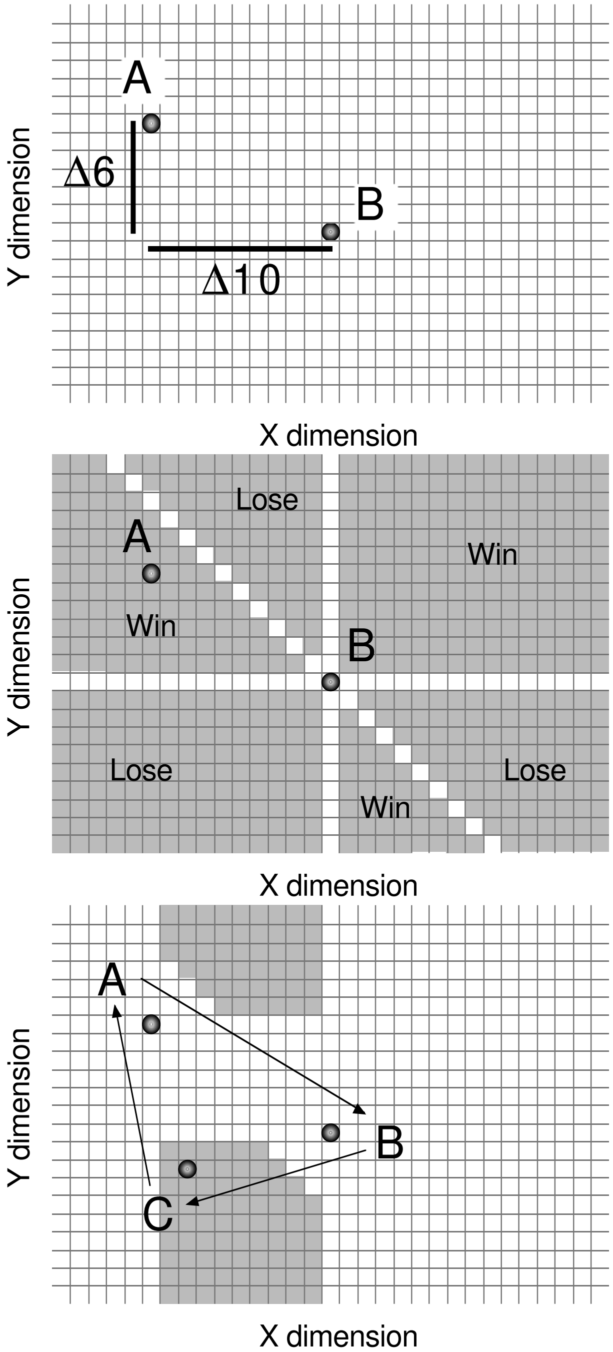


Figure 8.2: Intransitive numbers game on open plane. Top: Calculation of winning strategy in pair-wise interaction between Strategies A and B. Middle: Regions where an opponent will win and lose against Strategy B; Strategy A is in winning region. Bottom: Strategy C creates an intransitive cycle with Strategies A and B; Strategy C may be located anywhere in colored region and create this intransitivity.

Thus, setting $\epsilon = 0$ introduces additional ways in which a tie may occur between strategies. In particular, the Nash now ties strategies that others can beat; for example, in two dimensions, $\langle k, k \rangle$ (the Nash) ties both $\langle k, k-1 \rangle$ and $\langle k-2, k \rangle$, yet $\langle k-2, k \rangle$ beats $\langle k, k-1 \rangle$. *Therefore, for a non-Nash strategy to transform itself into the Nash, it may be required to discard certain skills—the ability to beat certain strategies; the goal of security is not always served by mere accumulation of prowess against opponents.* Due to these additional ties, virtually no strategy is dominated; those that are dominated are done so by the Nash. With $n = 2$, only four strategies are dominated, regardless of the value of k: $\langle 0, 0 \rangle$, $\langle k-1, k \rangle$, $\langle k, k-1 \rangle$, and $\langle k-1, k-1 \rangle$. With $n > 2$, the number of dominated strategies is only one (strategy $\langle 0, 0, \ldots, 0 \rangle$). Finally, with either value of epsilon, the Nash strategy is the pure strategy with the most wins (though, this is not a general feature of Nash equilibrium strategies).

8.7 Experiments

Rather than simply compare the performance of a coevolutionary algorithm with and without the aid of our memory mechanism, we wish instead to probe the ability of the memory mechanism to discover mixed strategies that provide greater and greater security. To accomplish this, we use $\mathcal{N}$ as a quasi-static evaluation function against which we repeatedly evolve a population of strategies.

8.7.1 Methods

The following procedure essentially transforms a *co*-evolutionary domain into an evolutionary domain. We use the game defined in Section 8.6.1 with $\epsilon = 0$, $k = 100$, and $n = 2$. We represent an individual pure strategy for this game with a bit-string b of length 200, such that the expressed strategy is $\langle \sum_{i=0}^{99} b_i, \sum_{i=100}^{199} b_i \rangle$. The only variation operator we apply is bit-wise mutation with a per-bit mutation probability of 0.01; recombinational operators are not used. We construct a random strategy by setting each of the 200 bits to one with a probability of 0.5; this yields an expected strategy of $\langle 50, 50\rangle$. The memory's capacity is $c = 100$.

The evolution proceeds in *epochs*, as follows:

0. Initialize memory and update with a random strategy.
1. Initialize population (of size 100) with random strategies.
2. Evolve population against $\mathcal{N}$ for 30 generations (one epoch).
3. If highest-scoring individual $\hat{s}$ in population beats $\mathcal{N}$, then update memory with $\hat{s}$.
4. Goto Step 1.

Each generation in Step 2 proceeds as follows:

a. Evaluate each individual (pure strategy) s in the population against $\mathcal{N}$ (the score w_i obtained by each individual i is in the range $[-1, 1]$).

b. The fitness of individual i is $f_i = w_i - \min(w) + 0.1$ (all fitness values are > 0).

c. Copy the 10 most-fit individuals to next generation (elitism).

d. Fill the remaining 90 positions with offspring using "fitness-proportionate" selection and asexual reproduction.

8.7.2 Results: Nash Memory

Figure 8.3 (Left) shows the mean and median scores obtained by the most fit individual $\hat{s}$ at the end of each epoch over 52 trials of our experiment. At the beginning of the experiment, the evolution step (Step 2, above) easily finds pure strategies that obtain the maximal score of 1.0 when played against the Nash memory's mixture $\mathcal{N}$. Over the next 50 epochs, however, the ability of evolution to discover pure strategies that score well against $\mathcal{N}$ is gradually neutralized as the memory mechanism integrates knowledge gained from previous epochs. Indeed, over subsequent epochs, the median score obtained by evolution asymptotically approaches zero, which is the *value* of the intransitive numbers game (and zero-sum games, in general—see Section 8.2, above). The mean score, however, actually becomes negative; thus, the distribution is not normal. This indicates that $\mathcal{N}$ tends to be somewhat superior to the strategies that evolution is able to discover. Over the course of an experiment, the number of pure strategies in support of $\mathcal{N}$ tends to grow. The mean size of C($\mathcal{N}$) at Epoch 500 is about 45, though the distribution is not normal (median is 50).

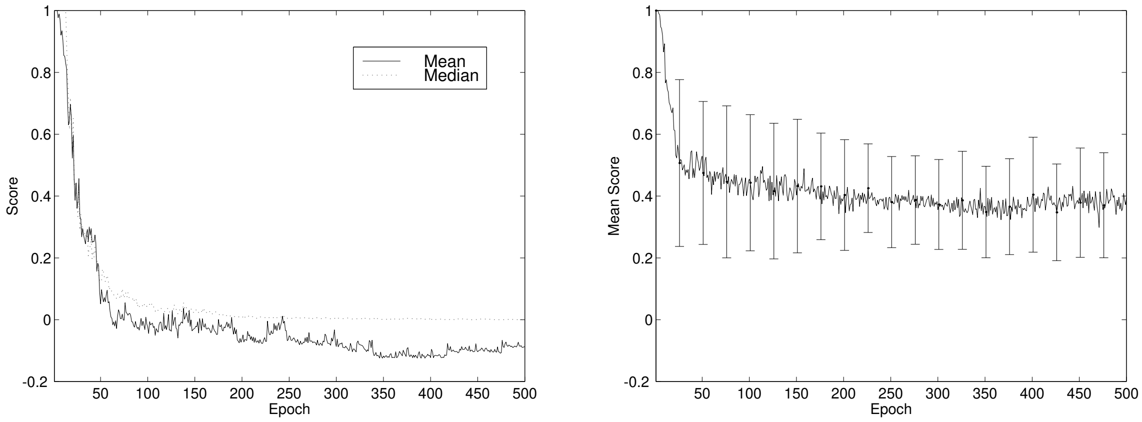


Figure 8.3: Performance of evolution over time against (Left) Nash memory, over 52 trials, and (Right) BOG memory ($m = \infty,\ l = 100$), over 54 trials.

Figures 8.4 and 8.5 shows the behavior of a typical trial. (For convenience, $\mathcal{N}^i$ will denote the value of $\mathcal{N}$ at the beginning of the i-th epoch, similarly for $\mathcal{M}^i$.) The memory's strategy $\mathcal{N}^1$ is initialized to the pure strategy $\langle 53, 41\rangle$ (indicated by the square in Figure 8.5). The evolution step (Step 2, above) easily finds a pure strategy

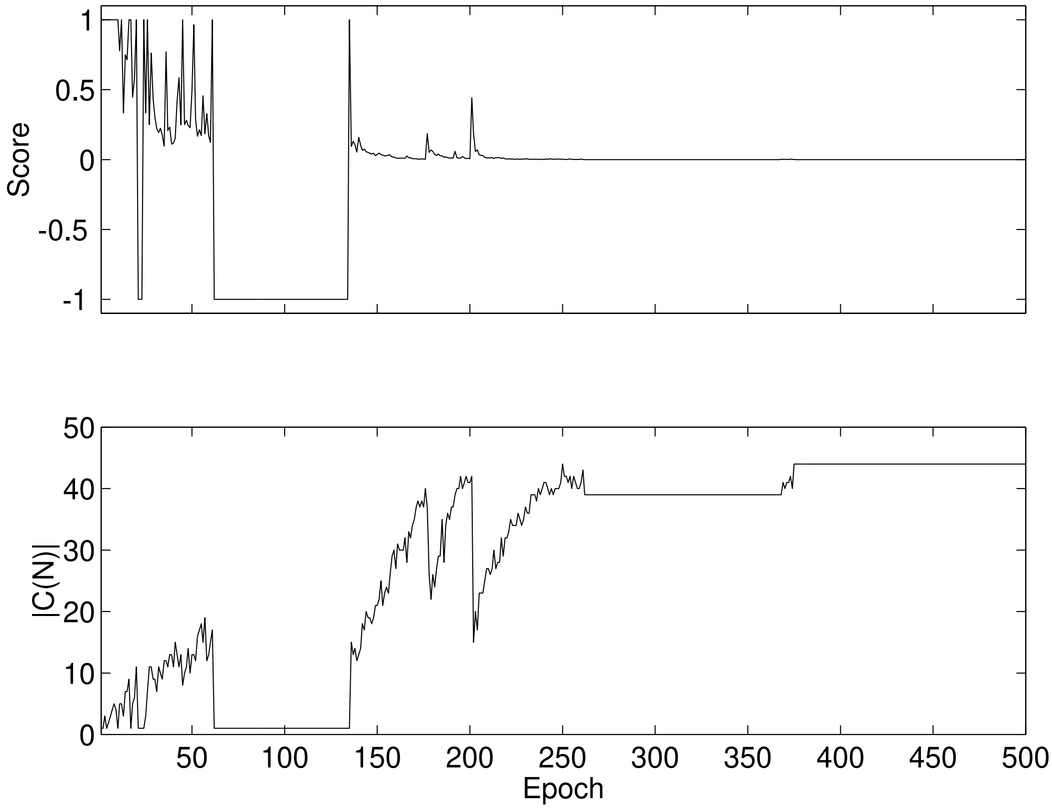


Figure 8.4: Example run (#31) of evolution against Nash memory over time. Upper graph gives maximal score achieved by evolution against $\mathcal{N}$ at the end of each epoch; lower graph shows size of support set $\mathrm{C}(\mathcal{N})$ at the beginning of each epoch.

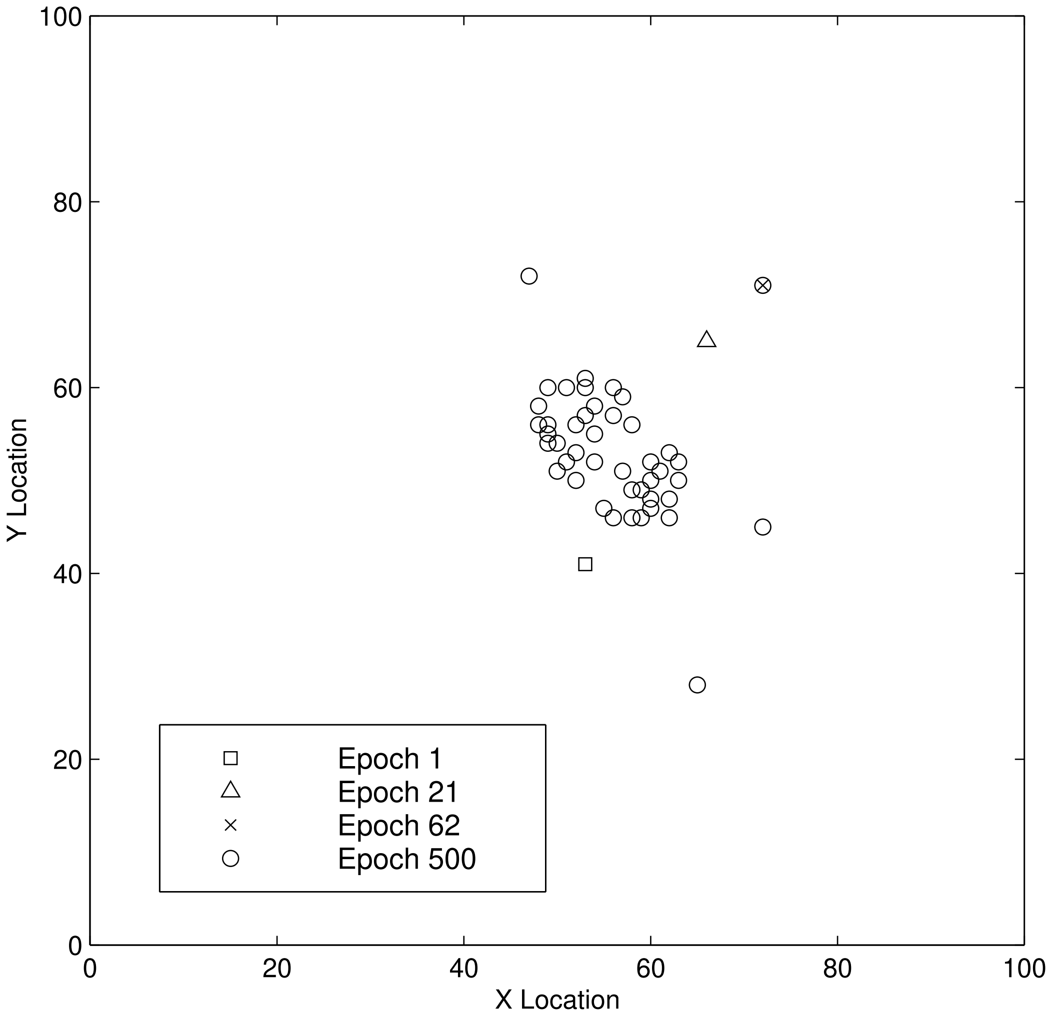


Figure 8.5: Example run of evolution against Nash memory over time. Pure strategies in support of $\mathcal{N}$ at the beginning of Epochs 1, 21, 62, and 500.

$\langle 55, 47\rangle$ that beats $\mathcal{N}^1$. Accordingly, we update the memory such that $\mathcal{N}^2$ is $\langle 55, 47\rangle$, and move $\langle 53, 41\rangle$ to the memory set $\mathcal{M}^2$.

In Epoch 2, the evolution step discovers the pure strategy $\langle 48, 52\rangle$, which beats $\mathcal{N}^2$. Since $\mathcal{M}^2$ contains something, we must evaluate the performance of $\langle 48, 52\rangle$ against the contents of $\mathcal{M}^2$ before we can determine $\mathcal{N}^3$. Though $\langle 48, 52\rangle$ beats $\langle 55, 47\rangle$, we find that it loses to $\langle 53, 41\rangle$ (the strategy in $\mathcal{M}^2$). Thus, we find an intransitive cycle; indeed, this intransitivity is symmetric (as in the Rock-Scissors-Paper game), so no one pure strategy can be argued to be any better than another (in any sense). This is not to say that the three strategies are interchangeable, however, for each one has unique strengths and weaknesses with respect to the other two. Since none of these pure strategies has an empty vulnerability set (see Section 8.3, above), $\mathcal{N}^3$ must be a mixture of some kind. The solution (found with linear programming) happens to be a mixed strategy where each of the pure strategies is played with equal probability; $\mathcal{N}^3$ has all three strategies in support and $\mathcal{M}^3$ is empty.

By the time we arrive at $\mathcal{N}^{20}$, eleven strategies are in support. In this epoch, evolution discovers pure strategy $\langle 66, 65\rangle$ (triangle in Figure 8.5), which beats each of the eleven support strategies. Further, because $\mathcal{N}^{21} = \{\langle 66, 65\rangle\}$, we can infer that $\langle 66, 65\rangle$ is also secure against the contents of $\mathcal{M}^{20}$. The strategy $\mathcal{N}^{21}$ is sufficiently good that the evolution step obtains the worst possible score (-1) over the next few epochs. Later, we encounter a similar, but more pronounced, situation where $\mathcal{N}^{62}$ is the pure strategy $\langle 72, 71\rangle$ ('x' in Figure 8.5). This strategy remains until Epoch 136, where it is replaced by a mixture. The mixture $\mathcal{N}^{500}$ (circles in Figure 8.5) retains $\langle 72, 71\rangle$ in support.

Several other sample runs are shown in Figure 8.6.

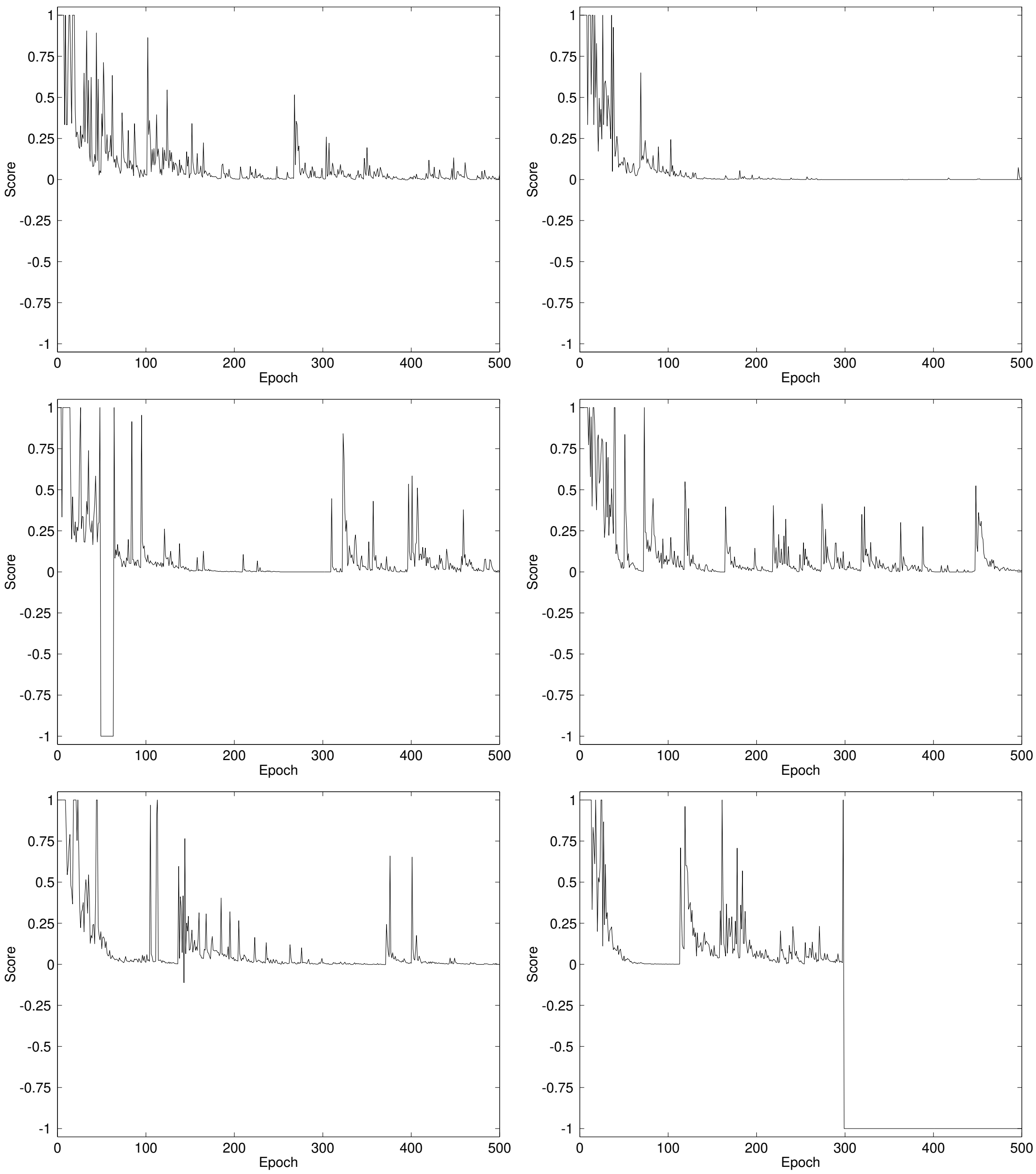


Figure 8.6: Sample runs using Nash memory mechanism. Runs 3, 5, 7, 12, 21, and 22.

8.7.3 Results: Best of Generation

We also run our experiment using the best-of-generation memory mechanism with $m = \infty,\ l = 100$ (see Section 8.4, above). During evaluation, an evolving individual's score against the memory is the average score obtained against the strategies sampled from the memory; the highest-scoring individual at the end of an epoch is added to the memory if its score is greater than 0. We run 54 trials of this experiment, six of which are shown in Figure 8.7; these graphs show the score obtained by the most fit individual $\hat{s}$ at the end of each epoch. We clearly see that random samples of memory contents do not provide a sufficiently organized collection of opponents to effectively broaden selection pressure.

Figure 8.3 (Right) shows the mean scores (with standard deviation) obtained by $\hat{s}$ over the 54 trials of this experiment; these data are normally distributed. The mean score obtained by $\hat{s}$ between epochs 250 and 500 is ≈ 0.375 ($\sigma \approx 0.160$). In other experiments where $m = 100,\ l = 100$ (not shown), the expected score of $\hat{s}$ at steady-state is ≈ 0.758 ($\sigma \approx 0.148$). Thus, the organizing principle behind BOG provides considerably less effective learning.

8.7.4 Results: Dominance Tournament

We next use the Dominance Tournament as our memory mechanism. The score given to an evolving individual is the number of consecutive pure strategies, from the most to least recently placed in the memory, that the individual beats. At the end of an epoch, if any individuals exist that beat the entire contents of the memory, we randomly select one such individual and place it in the memory. The DT method quickly converges onto a pure strategy that cannot be dominated by any other *with respect to the memory's contents*. Because we only consider those strategies in the memory

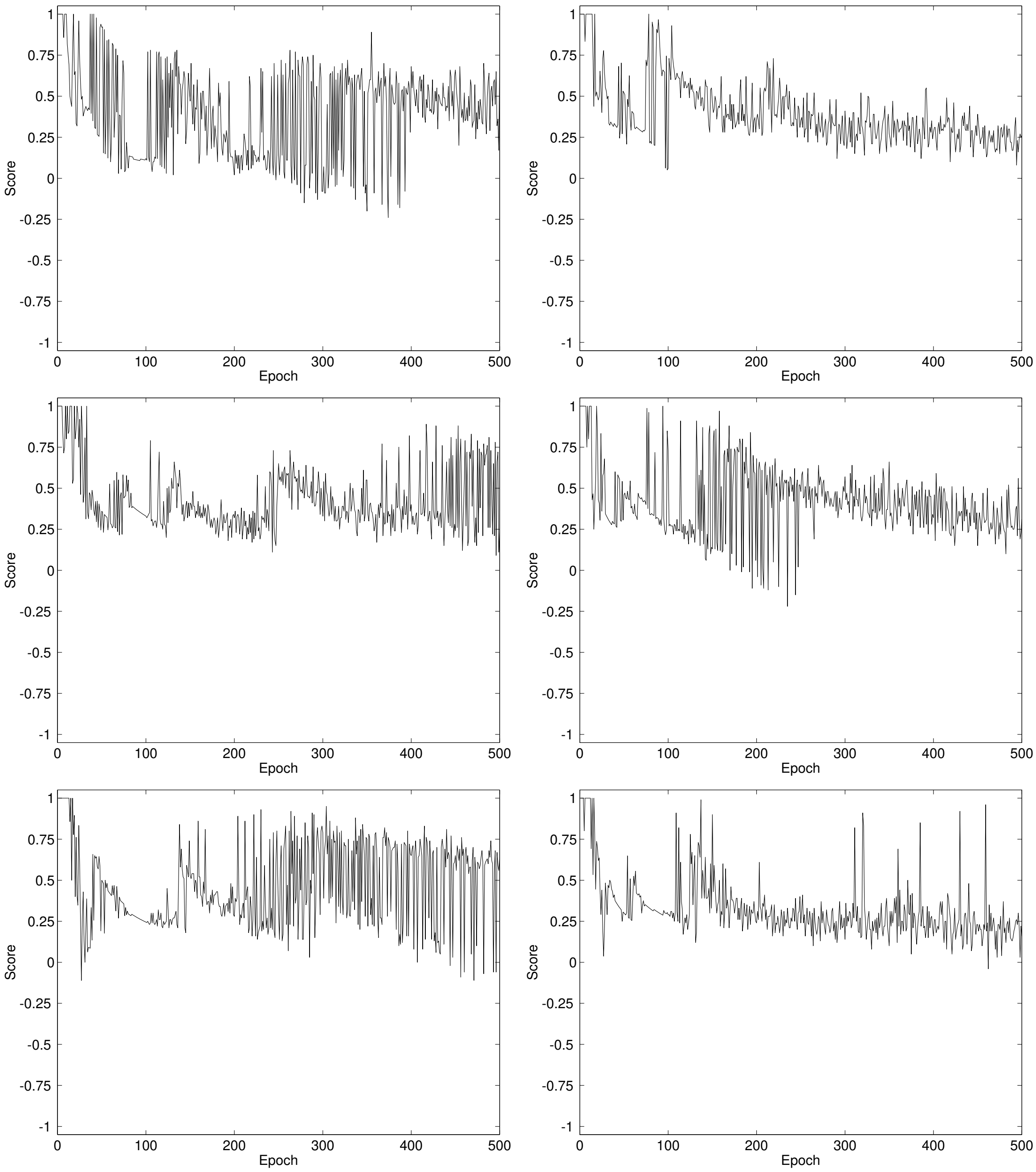


Figure 8.7: Sample runs using Hall of Fame memory mechanism. Runs 21, 22, 24, 36, 31, and 25.

to determine dominance, and not the entire universe of strategies, false positives may occur; indeed, all but four strategies in our game are actually *non*-dominated. Thus, the strategy onto which we converge is highly path-dependent; particularly, we may converge onto the dominated strategy $\langle 100, 99 \rangle$ (or $\langle 99, 100 \rangle$—see Section 8.6), after which the memory will reject the Nash $\langle 100, 100 \rangle$, which dominates $\langle 100, 99 \rangle$ but does not beat it. This situation is illustrated in Figure 8.8 (top). The small solid circles represent a hypothetical set of strategies that are added sequentially to the memory (moving diagonally to the top); each strategy in this sequence beats all other strategies that come before it in the sequence. The sequence terminates with strategy $\langle n-1, n \rangle$ (represented by the hollow circle). This strategy is actually dominated by the Nash strategy $\langle n, n \rangle$ (represented by the large shaded circle), yet we cannot add the Nash strategy to the memory because it ties the dominated strategy rather than beat it. Figure 8.8 (top and bottom) illustrates that the dominated strategy is done so by the Nash with respect to the shaded regions of the n by n square. With respect to these regions, the Nash does strictly better than the dominated strategy; the two strategies obtain identical outcomes with respect to all other strategies in the square. No strategy in the n by n square is capable of beating *all* of the strategies in our sequence; thus, we converge onto a dominated strategy. Further, of all the pure strategies onto which we may converge, only the Nash equilibrium strategy has an empty vulnerability set; all the others are beatable.

8.7.5 Results: Bootstrapping with the Nash Memory

The results we report in Section 8.7.2 only show that the Nash memory is able to learn a mixed strategy $\mathcal{N}$ that is secure against what the evolution step is likely to discover, given the limitations we place on the evolution, such as the random initial

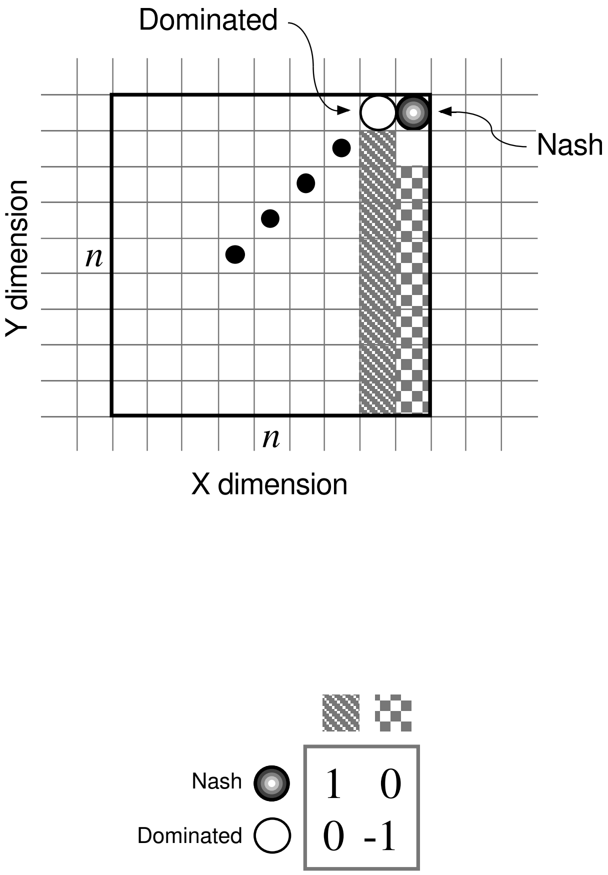


Figure 8.8: Dominance tournament may converge onto a dominated strategy. Top: Small solid circles indicate hypothetical set of strategies in memory, added sequentially from left to right; sequence ends with larger hollow circle. Bottom: Payoffs obtained by Nash and dominated strategies against strategies in shaded regions; the Nash is strictly better against these strategies.

population. In particular, the mixed strategies $\mathcal{N}$ obtained above are *not* the Nash equilibrium for the numbers game.

Therefore, we create a new outer loop around the method described in Section 8.7.1 to explore the ability of the memory to constructively contribute genetic material to the search process. When the evolution step is unable to score higher than 0.04 against $\mathcal{N}$ for 50 consecutive epochs, we change the procedure used to initialize the population: A snapshot of $\mathcal{N}$ is taken and used to initialize populations in all subsequent epochs (until our outer-loop criterion is met once again). Specifically, the initial population contains the strategies in support of $\mathcal{N}$ in proportion to their probabilities in the mixture distribution.

This new initialization process allows the evolution step to further challenge the Nash memory. While we do not converge onto the precise Nash strategy for our game (which is $\langle 100, 100\rangle$), we do obtain a mixture $\mathcal{N}$ (with 123 pure strategies in support) that virtually eliminates vulnerability to all strategies ($\mathcal{N}$'s worst score ≈ -0.05) except the true Nash ($\mathcal{N}$'s score ≈ -0.323); the memory mechanism is poised to accept the true Nash strategy, if search can find it. Thus, the solution $\mathcal{N}$ is an excellent approximation to the true Nash strategy *in behavior*, though superficially they appear nothing alike.

8.8 Conclusion

We examine the performance of three distinct memory mechanisms using Watson and Pollack's intransitive numbers game: Nash memory, Best-of-Generation, and Dominance Tournament. We do not intend to argue that the BOG and DT mechanisms cannot improve the performance of a coevolutionary algorithm; indeed, the BOG approach is known to help, e.g. [175, 148]. We are more interested in the role of coevolutionary memory and its organization.

We show that the Nash memory improves its collection of traits, expressed as a mixed strategy, as search exposes the memory to new traits. Over time, the mixed strategy asymptotically approaches the performance of the Nash equilibrium strategy for the numbers game, thus providing 1) an excellent approximation of the (game-theoretic) solution, and 2) a gradually increasing challenge for evolving strategies. In contrast, the BOG method exhibits only limited ability, when faced with pervasive intransitivity, to increasingly challenge evolution; lacking a strong organizing principle, BOG does not converge to the Nash equilibrium. The DT method implements the principle of domination, but only with respect to local knowledge; without the global knowledge required to properly determine domination, the numbers game easily leads the DT method astray.

Chapter 9

Monotonic Solution Concepts

9.1 Introduction

In this chapter, we investigate how a coevolutionary algorithm may behave if it never discards any information learned during its operation—all strategies discovered during search are retained and utilized. As a general principle, the expected quality of a solution returned by a search method should improve as the search method is given more computational resources in time and/or space. More desirable still, if we repeatedly query a search method (at "appropriate" points during its execution) on any single run, then the quality of the solution returned by the method should improve monotonically—that is, the quality of the solution at time t should be no worse than the quality at time $t-1$.

Given a coevolutionary algorithm that does not discard information during execution, can we expect it to meet the *desideratum* of monotonicity? On the one hand, we may expect that it must, since the algorithm's knowledge of the domain is ever increasing; the accumulation of strategies makes the evaluation process increasingly comprehensive and provides growing genetic diversity for the variation operators. On the other hand, we may remind ourselves that many of the problems associated with coevolution arise because evaluation is never fully comprehensive, and therefore can be misled when the domain contains intransitivities; new information may drastically shift our perspective—how, then, can we guarantee that the solution will improve monotonically with time?

This chapter shows that, when information is not discarded, a coevolutionary algorithm may or may not behave monotonically, depending upon the *solution concept* used by the algorithm. We develop a formal framework to discuss solution concepts and monotonicity and show that certain solution concepts (Nash equilibrium) guarantee monotonicity and others (including that most commonly used in coevolution) do not. The groundwork for this chapter is built in Chapter 2. Section 9.2 develops the notion of the *preference relation*, which result from a solution concept; Section 9.3 presents the proofs that certain solution concepts are monotonic with respect to the preference relation, while others are not; Sections 9.4 through 9.6 provide additional results that compare and contrast different solution concepts; finally, Section 9.7 discusses the relevance of the monotonicity results to open-ended domains.

9.2 Properties of Preference Relation

In Chapter 2, we define a preference relation (Definition 1) in which we prefer one configuration $\mathcal{K}_\alpha$ over another configuration $\mathcal{K}_\beta$ if an only if every game for which $\mathcal{K}_\beta$ is a solution is a subgame of a game for which $\mathcal{K}_\alpha$ is a solution. For convenience, we reproduce Definition 1 here.

Definition 7 (Preference Relation) $\mathrm{Pref}(\mathcal{K}_\alpha, \mathcal{K}_\beta) = 1$ *iff for all* $\mathbf{T}_\beta$*, where* $\mathbf{T}_\beta \subseteq \mathbf{T}$ *and* $\mathcal{K}_\beta \in \mathrm{S}^*(\mathbf{T}_\beta, \mathcal{O})$*, there exists* $\mathbf{T}_\alpha$*, where* $\mathbf{T}_\alpha \subseteq \mathbf{T}$ *and* $\mathcal{K}_\alpha \in \mathrm{S}^*(\mathbf{T}_\alpha, \mathcal{O})$*, such that* $\mathbf{T}_\alpha \supset \mathbf{T}_\beta$*.*

9.2.1 Reflexivity, Symmetry, and Transitivity

The preference relation is not reflexive—we will never prefer a configuration to itself: $\text{Pref}(\mathcal{K}_\alpha, \mathcal{K}_\alpha) = 0$. Non-reflexivity follows from the definition of the preference relation; given the set of games $\mathcal{G}_\alpha$ in which $\mathcal{K}_\alpha$ is a solution, there must exist at least one game $\mathbf{G} \in \mathcal{G}_\alpha$ that does not have a superset in $\mathcal{G}_\alpha$.

The preference relation is not symmetric—if we prefer one configuration to another, then the converse is not true: $\text{Pref}(\mathcal{K}_\alpha, \mathcal{K}_\beta) = 1 \implies \text{Pref}(\mathcal{K}_\beta, \mathcal{K}_\alpha) = 0$. Non-symmetry also follows from the definition of the preference relation. Let $\mathcal{G}_\alpha$ be the set of games where configuration $\mathcal{K}_\alpha$ is a solution and $\mathcal{G}_\beta$ be the set of games where $\mathcal{K}_\beta$ is a solution. Since we prefer $\mathcal{K}_\alpha$ to $\mathcal{K}_\beta$, each game in $\mathcal{G}_\beta$ is a subgame of some game in $\mathcal{G}_\alpha$. If we simultaneously prefer $\mathcal{K}_\beta$ to $\mathcal{K}_\alpha$, then each game in $\mathcal{G}_\alpha$ must also be a subgame of some game in $\mathcal{G}_\beta$; but, for this to be true, we must allow for an intransitivity in the subset relation.

While the preference relation is neither reflexive nor symmetric, it is transitive—if we prefer configuration $\mathcal{K}_\alpha$ to $\mathcal{K}_\beta$, and prefer $\mathcal{K}_\beta$ to $\mathcal{K}_\gamma$, then we prefer $\mathcal{K}_\alpha$ to $\mathcal{K}_\gamma$. The transitivity of the preference relation follows directly from the transitivity of the subset relation, as Figure 9.1 illustrates. Since each game in $\mathcal{G}_\gamma$ is a subset of some game in $\mathcal{G}_\beta$, and each game in $\mathcal{G}_\beta$ is a subset of some game in $\mathcal{G}_\alpha$, then each game in $\mathcal{G}_\gamma$ must also be a subset of some game in $\mathcal{G}_\alpha$.

Thus, the preference relation yields a partial ordering over the space of configurations.

9.2.2 Monotonicity

Let $\mathcal{G}_x$ be the set of games for which $\mathcal{K}_x$ is a solution and $\mathcal{G}_y$ the set of games for which $\mathcal{K}_y$ is a solution. The preference relation given by Definition 7 says that we prefer

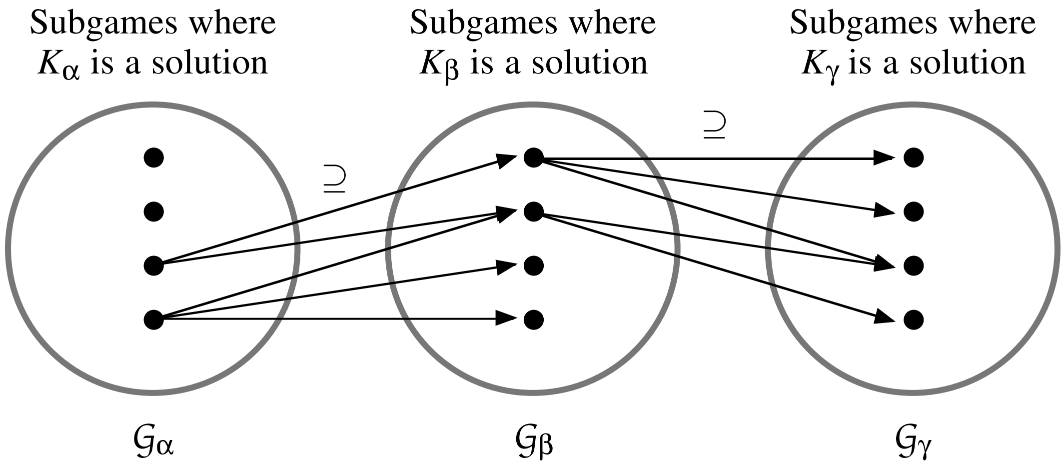


Figure 9.1: Transitivity of preference relation follows from transitivity of superset relation. Directed edge indicates superset relation between two games.

$\mathcal{K}_x$ to another configuration $\mathcal{K}_y$ if and only if every game in $\mathcal{G}_y$ is a subgame of some game in $\mathcal{G}_x$. Of course, not every game in $\mathcal{G}_x$ must be a supergame; in particular, the preference relation allows the existence of a game in $\mathcal{G}_x$ that is a subgame of a game in $\mathcal{G}_y$, as shown in Figure 9.2. Here, we prefer $\mathcal{K}_x$ to $\mathcal{K}_y$, yet game $E \in \mathcal{G}_X$ is a subgame of game $D \in \mathcal{G}_y$.

When a configuration $\mathcal{K}$ is a solution for games α and β, where $\alpha \supset \beta$, but not for some game γ, where $\alpha \supset \gamma \supset \beta$, then we call the solution concept $\mathcal{O}$ *non-monotonic*. We define a *monotonic* solution concept to be one such that every configuration $\mathcal{K}$ that is a solution to games α and β, where $\alpha \supseteq \beta$, will also be a solution to any game γ, where $\alpha \supseteq \gamma \supseteq \beta$. The solution concept in Figure 9.2 is therefore non-monotonic and the relationship between game-sets $\mathcal{G}_x$ and $\mathcal{G}_y$ is an *instance* of non-monotonicity.

9.2.3 State of Knowledge

The game strategies that 1) have been discovered by a search heuristic and 2) are still in the computer's memory, such that they can be utilized by the heuristic, are defined as the heuristic's *state of knowledge*. We denote the state of knowledge at

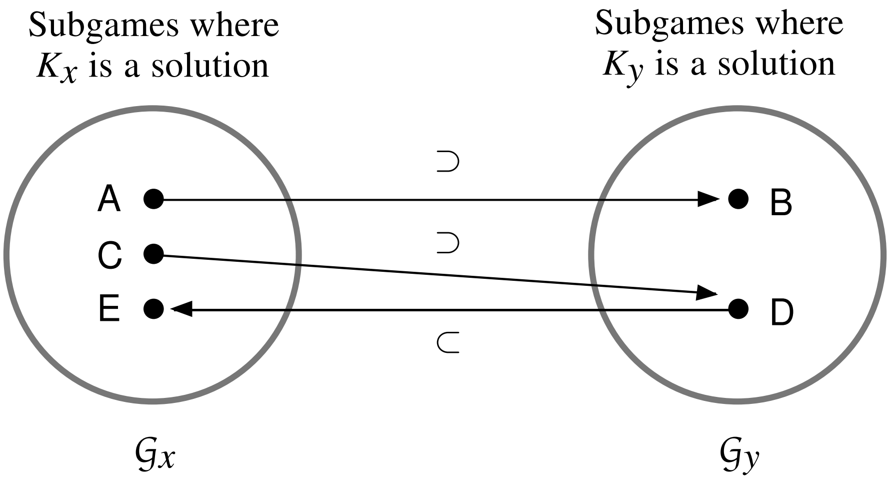


Figure 9.2: Non-monotonic solution concept.

time t as $\mathcal{W}^t$. At any time-step t that is appropriate to query the search heuristic, the configuration $\mathcal{K}_w$ returned by the heuristic should be a solution to the game defined by $\mathcal{W}^t$. Otherwise, the heuristic does not implement the solution concept $\mathcal{O}$. Note that all strategies included in configuration $\mathcal{K}_w$ must also be in $\mathcal{W}^t$—we cannot use strategies of which we lack knowledge.

9.2.4 Dynamics of Non-Monotonic Solution Concepts

For any solution concept, the preference relation provides a global partial ordering of configurations. If the solution concept is non-monotonic, however, the process of local search—even when it does not discard any information discovered during search—may be unable to conform to the preference relation and will contradict it. For example, let us return to Figure 9.2. If the game $E \in \mathcal{G}_x$ is our state of knowledge at time t (i.e., $\mathcal{W}^t = E$), then the solution returned by the search heuristic when queried is $\mathcal{K}_x$. Let us say that at time $t+1$, the heuristic discovers new strategies and our state of knowledge is expanded to include these additional strategies such that our new state of knowledge is game D. Since $D \notin \mathcal{G}_x$ and $D \in \mathcal{G}_y$, our solution is now $\mathcal{K}_y$. We thus appear to have run counter to our preference relation, which says that we prefer $\mathcal{K}_x$ to $\mathcal{K}_y$.

9.2.5 Dynamics of Monotonic Solution Concepts

The definition of a monotonic solution concept guarantees that local search, when it does not discard information, will always conform to the preference relation. To show this must be the case, let us imagine two states of knowledge, $\mathcal{W}^t$ and $\mathcal{W}^{t+k}$, where $\mathcal{W}^t \subset \mathcal{W}^{t+k}$ and $k \geq 1$. Further, let us say that the solution for $\mathcal{W}^t$ is $\mathcal{K}_x$ and for $\mathcal{W}^{t+k}$ is $\mathcal{K}_y$. Now, if the solution concept if monotonic, then we cannot prefer $\mathcal{K}_x$ to $\mathcal{K}_y$. If we prefer $\mathcal{K}_x$ to $\mathcal{K}_y$, then there must exist some game $\mathbf{G}$ that is a superset of $\mathcal{W}^{t+k}$ for which $\mathcal{K}_x$ is a solution. But, if $\mathcal{K}_x$ is a solution of both games $\mathcal{W}^t$ and $\mathbf{G}$, and $\mathcal{W}^t \subset \mathcal{W}^{t+k} \subset \mathbf{G}$, then either $\mathcal{K}_x$ must also be a solution to $\mathcal{W}^{t+k}$ (in which case we do not transition to $\mathcal{K}_y$) or the solution concept is not monotonic.

We can construct a directed graph of the preference relation where each vertex is a configuration and for each pair of configurations $\mathcal{K}_x$ and $\mathcal{K}_y$, where we prefer $\mathcal{K}_x$ to $\mathcal{K}_y$, there is a directed edge from $\mathcal{K}_x$ to $\mathcal{K}_y$. If local search does not discard information, then the property of monotonicity means that once we visit a vertex on the graph, we will never follow an edge leading back to that vertex or to any of its descendants in the graph. Thus, a monotonic solution concept means that the quality of the result returned by a search heuristic (assuming that it does not discard information) will also increase monotonically over time; this is not guaranteed to be true for a non-monotonic solution concept, even when information is never discarded.

9.3 Solution Concepts

Having discussed the properties of monotonic and non-monotonic solution concepts in general, we now investigate some specific solution concepts. We show that Nash equilibrium is monotonic, non-domination is monotonic if it is implemented in a specific way, and the "best scoring" solution concept commonly used in coevolutionary algorithms is not monotonic.

9.3.1 Nash Equilibrium

Nash equilibrium is a monotonic solution concept in zero-sum games. Our proof is by contradiction. Let us assume that the Nash equilibrium concept is not monotonic. Figure 9.3 shows our canonical non-monotonic solution concept (reproduced from Figure 9.2 for convenience). We prefer configuration $\mathcal{K}_x$ to configuration $\mathcal{K}_y$. From the definition of Nash equilibrium, we know that if $\mathcal{K}_x$ is secure against the strategies in game A, then it must also be secure against any subset of A; in particular, $\mathcal{K}_x$ must be secure against game D. But, for $\mathcal{K}_x$ to be a *solution* for state of knowledge D, the contents of $\mathcal{K}_x$ must be included in D—no solution can include strategies outside of our state of knowledge. Because the strategies in $\mathcal{K}_x$ are in game E, they must also be present in D (and C). Therefore, configuration $\mathcal{K}_x$ must be a solution for game D, as well, in contradiction to what Figure 9.3 indicates. Thus, the Nash solution concept must be monotonic. (To extend this proof to symmetric variable-sum games, we merely substitute the criterion of security with best-response; that is, a strategy is Nash with respect to some set of strategies if no strategy in the set is a better response to the strategy than the strategy itself.) $\square$

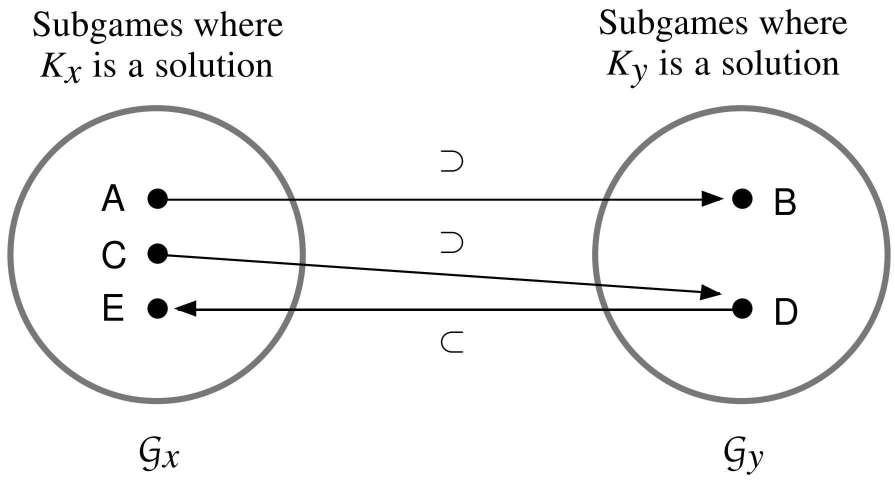


Figure 9.3: A non-monotonic solution concept.

9.3.2 Non-Dominated Front

Another solution concept is that of the non-dominated front. Whether this concept is monotonic or not depends upon a particular aspect of its implementation; specifically, if we allow a newly-discovered strategy to join the non-dominated front, when it appears identical to strategies already on the front, then non-monotonicities may result. Therefore, disallowing strategies that appear identical to those on the front is sufficient to guarantee that the solution concept is monotonic. The proof below is rather more involved than that for Nash equilibrium above, so we first review some properties of the non-domination solution concept and establish some notation.

Properties Let us first recall some properties of the non-domination concept. To establish that two strategies α and β mutually non-dominate each other, we need only two dimensions of comparison, one in which is α is superior to β and another in which β is superior to α. Nevertheless, we might not have knowledge of these dimensions that prove mutual non-dominance between the strategies. If α and β in fact mutually non-dominate, but we lack knowledge of all dimensions that distinguish their performance, then α and β will appear identical; if we have dimensions in which β out-performs α, but lack knowledge of dimensions in which α outperforms β, then α will appear dominated by β.

To definitively establish that one strategy α dominates another β, we need to demonstrate that α is no worse than β in *all* dimensions of comparison, and strictly better in at least one. If α dominates β, but we lack knowledge of any dimension in which α is superior to β, then the two strategies will appear identical, and therefore (trivially) mutually non-dominating. In general, as more dimensions of behavior are used to compare two strategies, the apparent relation between the strategies, if it changes at all, can only progress from *identical* to *dominated* or from *dominated* to *mutually non-dominating.* Figure 9.4 illustrates this process as a deterministic three-state automaton.

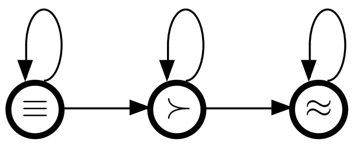


Figure 9.4: Allowable state transitions in relation between two strategies as new dimensions of behavior are discovered.

Notation Table 9.1 summarizes our notation for the non-domination solution concept. All of the relations shown below are implicitly understood to exist with respect to some set of dimensions of comparison. For example, it may be that α dominates β with respect to the set of dimensions $\mathcal{R}$ (denoted $\alpha \overset{\mathcal{R}}{\succ} \beta$), but mutually non-dominates with respect to a different set of dimensions $\mathcal{S}$ (denoted $\alpha \overset{\mathcal{S}}{\approx} \beta$). As another example, α may appear identical to β given a set of dimensions $\mathcal{R}$ (denoted $\alpha \overset{\mathcal{R}}{\equiv} \beta$), but appear dominated with respect to another set of dimensions $\mathcal{S}$. Finally, if α and β appear identical with respect to the entire universe of strategies $\mathcal{U}$, then they in fact have equal payoff profiles; that is, $\alpha \overset{\mathcal{U}}{\equiv} \beta \Longleftrightarrow \alpha = \beta$. (Of course, in an open-ended domain, we can never firmly establish equality between two strategies; we may yet discover a dimension of behavior that distinguishes them.)

Notation	Meaning
$\alpha = \beta$	α and β in fact have identical payoff profiles
$\alpha \overset{\mathcal{R}}{\equiv} \beta$	α and β appear identical, given the dimensions used for comparison
$\alpha \overset{\mathcal{R}}{\succ} \beta$	α dominates β with respect to $\mathcal{R}$
$\alpha \overset{\mathcal{R}}{\approx} \beta$	α and β mutually non-dominate each other with respect to $\mathcal{R}$ (which implies $\alpha \overset{\mathcal{S}}{\approx} \beta$ for any $\mathcal{S} \supset \mathcal{R}$)

Table 9.1: Notation for non-domination solution concept.

Theorem and Proof We are now ready to state and prove the following theorem.

Theorem 1 (Non-Monotonicity of Non-Domination) *Let us prefer configuration $\mathcal{K}_x$ to $\mathcal{K}_y$. A non-monotonic transition from configuration $\mathcal{K}_x$ to $\mathcal{K}_y$ implies that at least one of the configurations contains a strategy that appears identical to another strategy in the configuration, given its state of knowledge.*

Figure 9.5 shows a non-monotonic transition from $\mathcal{K}_x$ to $\mathcal{K}_y$. Configuration $\mathcal{K}_x$ is a solution to games A and C, but not B, for which configuration $\mathcal{K}_y$ is a solution. Further, A is a superset of B, which is a superset of C. Since all games in $\mathcal{K}_y$ are subsets of some game in $\mathcal{K}_x$, we prefer $\mathcal{K}_x$ to $\mathcal{K}_y$. Our preliminary observations are as follows: Games A and C must contain $\mathcal{K}_x$, since $\mathcal{K}_x$ is a solution to those games; similarly, game B must contain $\mathcal{K}_y$. Because of the superset relationship between the games, we also know that B must contain $\mathcal{K}_x$, and A must contain $\mathcal{K}_y$. We can thus re-write the games as follows:

$$C = X \cup \alpha \tag{9.1}$$

$$B = X \cup \alpha \cup Y \cup \beta \tag{9.2}$$

$$A = X \cup \alpha \cup Y \cup \beta \cup \gamma \tag{9.3}$$

where X and Y are the strategies in $\mathcal{K}_x$ and $\mathcal{K}_y$, respectively, and α, β, and γ are additional strategies that may be in the games but are not part of any solution (and are therefore dominated by one or both of $\mathcal{K}_x$ and $\mathcal{K}_y$).

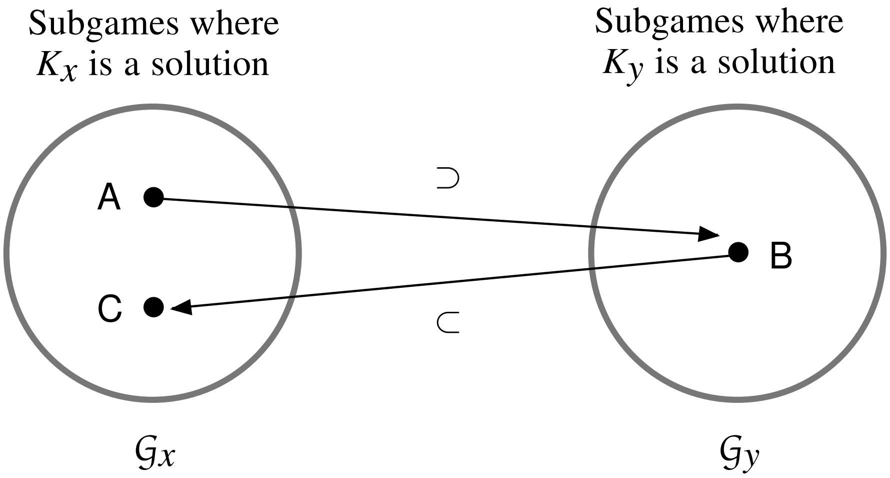


Figure 9.5: Monotonicity of Non-Domination.

As a next step, we must account for the fact that we do not know the relationship between the two configurations—for example, do they have an empty intersection or not? Is one a subset of the other? Therefore, we re-write X and Y to be $X = x \cup z$ and $Y = y \cup z$, respectively; the set z represents the intersection (if any) between X and Y, and x and y represent those elements (if any) that are unique to X and Y, respectively. Our equations are now:

Solution	Game	
$\mathcal{K}_x = x \cup z$	$C = x \cup z \cup \alpha$	(9.4)
$\mathcal{K}_y = y \cup z$	$B = x \cup z \cup \alpha \cup y \cup \beta$	(9.5)
$\mathcal{K}_x = x \cup z$	$A = x \cup z \cup \alpha \cup y \cup \beta \cup \gamma$	(9.6)

No more that one of x, y, and z may be an empty set. If $x = \emptyset$ and $y = \emptyset$, then the two configurations $\mathcal{K}_x$ and $\mathcal{K}_y$ must be identical, which by definition they cannot be. If $x = \emptyset$ and $z = \emptyset$, then $\mathcal{K}_x$ contains no strategies, which also cannot be the case (similarly for $\mathcal{K}_y$). Four possible cases remain, as summarized in Table 9.2.

Case	Property	Notes
1	$x \neq \emptyset, z \neq \emptyset, y \neq \emptyset$	Non-empty intersection and each configuration contains unique strategies
2	$x = \emptyset$	$\mathcal{K}_x$ is a proper subset of $\mathcal{K}_y$
3	$y = \emptyset$	$\mathcal{K}_x$ is a proper superset of $\mathcal{K}_y$
4	$z = \emptyset$	Empty intersection and each configuration contains unique strategies

Table 9.2: Four possible relationships between $\mathcal{K}_x$ and $\mathcal{K}_y$.

Case 1 The reasoning we follow to prove the theorem in Case 1 is shown in Figure 9.6. We assume that x, y, and z are non-empty. In game B, x is dominated. Therefore, for each element $x_i \in x$ there exists an element in one or both of z and y that dominates x_i. We are free to assume either that $y_j \in y$ dominates x_i in game B or that y_j and x_i are mutually non-dominating; that is, either $y_j \overset{B}{\succ} x_i$ or $y_j \overset{B}{\approx} x_i$. (We note that y_j cannot appear identical to x_i in game B because y_j cannot simultaneously be non-dominated and appear identical to a dominated strategy.) Regardless of the relationships between the elements of y and x in game B, the only possible relationship between y and x in game $A = B \cup \gamma$ is one of mutual non-domination, since x is non-dominated in game A. That is, $y \overset{B\cup\gamma}{\approx} x$. Therefore, the only way for y to be dominated in game A is to be dominated by z; that is, $y \overset{B\cup\gamma}{\prec} z$. This implies that every element in y must appear identical to some element in z in game B—there is no other way for the elements of y to avoid domination in game B yet become dominated when we introduce the additional strategies of γ in game A. Thus, we prove the theorem for Case 1: Configuration $\mathcal{K}_y$ contains strategies that appear identical to each other in the state of knowledge B, to which we make a non-monotonic transition.

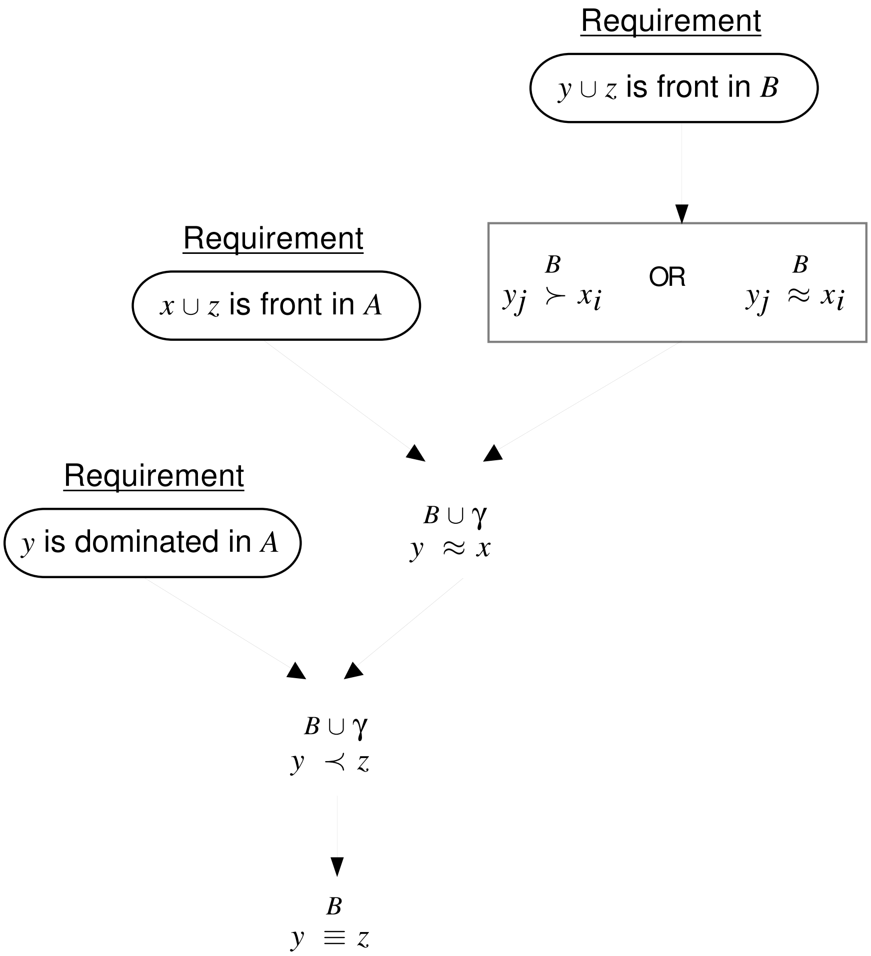


Figure 9.6: Proof of theorem for Case 1, where $x \neq \emptyset, z \neq \emptyset, y \neq \emptyset$.

Case 2 Here we assume that $x = \emptyset$; this implies that configuration $\mathcal{K}_x$ is a proper subset of $\mathcal{K}_y$. If the elements in y are not dominated in game B, but become dominated in game $A = B \cup \gamma$, where only z is non-dominated, then the following must be true. In game A, for each element in y there must exist an element in z that dominates it, that is, $y \overset{A}{\prec} z$. This implies that each element of y must appear identical to some element in z in game B; as we state above, there is no other way for the elements of y to avoid domination in game B yet become dominated when we introduce the additional strategies of γ in game A. Thus, we prove the theorem for Case 2: Configuration $\mathcal{K}_y$ again contains strategies that appear identical to each other in the state of knowledge B, to which we make a non-monotonic transition.

Case 3 Here we assume that $y = \emptyset$; this implies that configuration $\mathcal{K}_x$ is a proper superset of $\mathcal{K}_y$. Our argument is essential identical to that presented for Case 2. If the elements in x are not dominated in game C, but become dominated in game B, then the following must be true. In game B, for each element in x there must exist an element in z that dominates it, that is, $z \overset{B}{\succ} x$. This implies that each element of x must appear identical to some element in z in game C; there is no other way for the elements of x to avoid domination in game C yet become dominated when we introduce the additional strategies of β in game B. Thus, we prove the theorem for Case 3: This time, configuration $\mathcal{K}_x$ contains strategies that appear identical to each other in the state of knowledge A, from which we make a non-monotonic transition.

Case 4 We assume that z is empty. Thus, in game B, for each element in x there must exist some element in y that dominates it: $y \overset{B}{\succ} x$. But, when the dimensions γ are added in game A, the set x is to become non-dominated and y is to become dominated. Since z is empty, the only non-dominated strategies that remain to dominate y are in x; but, the strategies in x can at best mutually non-dominate those in y, for $y \overset{B}{\succ} x \Longrightarrow \neg(x \overset{B\cup\gamma}{\succ} y)$. Thus, z cannot be empty and we can discard Case 4. $\square$

Corollary Theorem 1 states that, for the non-domination solution concept, non-monotonicity implies a configuration that contains strategies that appear identical to each other. The following corollary follows immediately from the theorem.

Corollary 1 (Guarantee of Monotonicity) *Assume a state of knowledge $\mathcal{W}^t$ for which configuration $\mathcal{K}$ is a solution; further, assume that no strategy in $\mathcal{K}$ appears identical to any other strategy in $\mathcal{K}$ with respect to $\mathcal{W}^t$. Given a new state of knowledge $\mathcal{W}^{t+1} = \mathcal{W}^t \cup \mathcal{Q}$, the prohibition of non-dominated strategies in $\mathcal{Q}$ that appear identical to strategies in $\mathcal{K}$ that are non-dominated with respect to $\mathcal{W}^{t+1}$ is necessary and sufficient to guarantee the monotonicity of the non-domination solution concept.*

This corollary is an actionable result for coevolutionary algorithm design if the non-domination concept is to be used.

Digression: An Additional Proof In game B, x is dominated. Assuming that x, y, and z are non-empty, we can prove that for every element in x there must exist an element in z that dominates it. The reasoning of this proof is outlined in Figure 9.7. Our proof is by contradiction, so we assume that there exists some element $x_i \in x$ that is not dominated by any member of z; specifically, x_i must mutually non-dominate every element in z (x_i cannot appear identical to any element in z because x is dominated whereas z is not). The implication of $z \overset{B}{\approx} x_i$ is that some element $y_j \in y$ must dominate x_i in game B. Since both y and z are on the front in game B, the strategy y_j must either mutually non-dominate each strategy in z ($y_j \overset{B}{\approx} z$) or appear identical to some strategy z_h ($y_j \overset{B}{\equiv} z_h$). If there exists some z_h that appears identical to y_j, then we have a contradiction because y_j cannot simultaneously dominate x_i and appear identical to a strategy in z that we have already assumed must mutually non-dominate x_i. Therefore, y_j must mutually non-dominate each element in z.

When we move to game $A = B \cup \gamma$, the non-dominated front is now $x \cup z$. In particular, the strategies in y are now dominated. Because y_j dominates x_i in game B, the only possible relationship between y_j and x_i in game A is mutual non-domination (x_i cannot dominate y_j). Further, since y_j is mutually non-dominated with each strategy in z in game B, strategy y_j must remain mutually non-dominated with z in game A. Since no strategy in z dominates y_j, nor does x_i, then y_j must be dominated by some other strategy $x_k \in x$. For y_j to be dominated by x_k in game A, however, strategy y_j must appear identical to x_k in game B (y_j can neither dominate nor mutually non-dominate x_k in game B). If $y_j \overset{B}{\equiv} x_k$, then we have a contradiction because y_j cannot simultaneously be on the non-dominated front of game B and appear identical to another strategy that must be dominated in game B. Therefore, there cannot exist a strategy x_i that mutually non-dominates every element in z, and so for every element in x there must exist some element in z that dominates it. Further, for any z_h that dominates x_i in game B, strategy z_h must mutually non-dominate x_i in game A. $\square$

9.3.3 Best-Scoring Strategy

We can imagine another solution concept that simply picks the strategy $\hat{s}$ from a set of unique strategies $\mathcal{S}$ that obtains the highest average score from interaction with each member of $\mathcal{S}$. This solution concept is similar to that typically used in coevolution—namely, pick the individual that obtains the highest score after interaction with the entire population. The essential difference is that we assume $\mathcal{S}$ not to

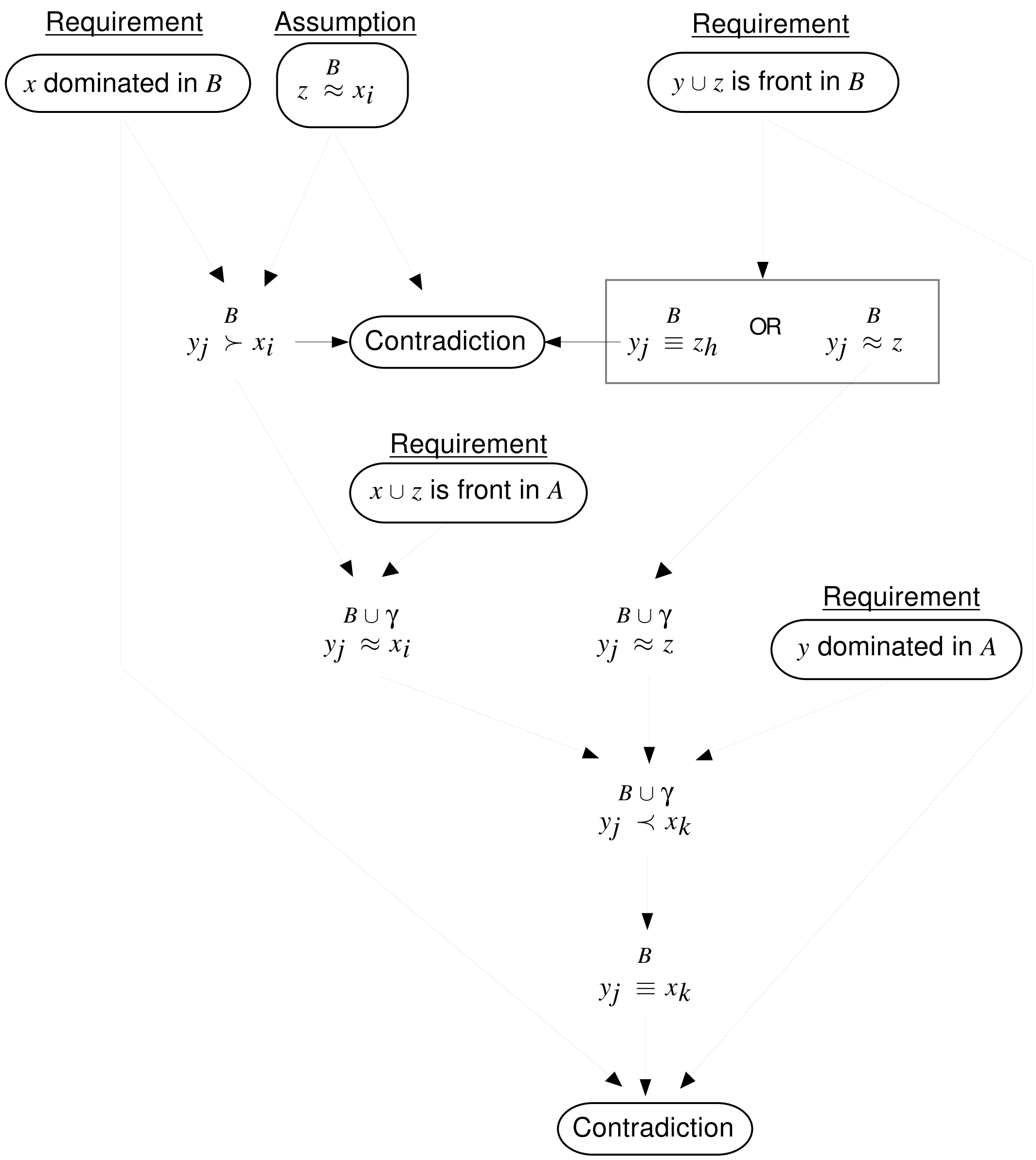


Figure 9.7: Proof by contradiction that each element in x is dominated by some element in z.

contain duplicates, and therefore the choice of $\hat{s}$ does not depend upon the frequency with which difference strategies might appear; in standard coevolution, of course, frequency-dependent effects exist.

We will call our alternative solution concept *best-scoring strategy* (BSS). Here we show that BSS is a non-monotonic solution concept. Our proof is by contradiction, so let us assume that BSS is monotonic. If BSS is monotonic, then by definition a configuration $\mathcal{K}_x$ that is a solution to games A and C, where $A \supseteq C$, must also be a solution to any game B, where $A \supseteq B \supseteq C$. Let us consider a simple symmetric zero-sum game **G**. We can imagine the following sub-games of **G**:

$$C = c \cup \{x, y\} \tag{9.7}$$

$$B = C \cup b \tag{9.8}$$

$$A = B \cup a \tag{9.9}$$

where x and y are individual strategies, and a, b, c, A, B, C are sets. Figure 9.8 illustrates our example as a payoff matrix ('*' indicates that any sub-matrix can be used).

Let us say that x and y tie each other; further, x beats everyone in c and a, but loses to everyone in b, whereas y beats everyone in b, but loses to everyone in c and a. Also, let us say that members of c tie against members of b and a, and members of b tie against members of a. Certainly, x is the solution to game C, and for x to be the solution to game A we need merely make $|a| + |c| > |b|$. Now, if BSS is a monotonic solution concept, then from the preceding should immediately follow that x is also a solution to game B. But, if the cardinality of b is greater than that of c, then x cannot be the solution to game B; instead, we find that y is the solution. Thus, our

	x	y	c	b	a
x	0	0	1	-1	1
y	0	0	-1	1	-1
c	-1	1	*	0	0
b	1	-1	0	*	0
a	-1	1	0	0	*

Figure 9.8: Game of Equations 9.7–9.8 as a payoff matrix.

assumption that BSS is monotonic is contradicted. □

The BSS solution concept is free of frequency-dependent effects, yet is still non-monotonic; thus, we can easily see that the standard coevolutionary solution concept of "best in the ecology" (BITE)—which is frequency-dependent—is also non-monotonic. The non-monotonicity of BSS and BITE is certainly consistent with the red-queen effect [197] and with the common sentiment that objective metrics of goodness are difficult to obtain for coevolutionary domains. In an effort to grasp the source of this difficulty, Luke and Wiegand [131] approach this issue from the point of view of the game, discussing properties that give a game an objective measure. In contrast, the work we present here shifts the focus of attention away from the game and to the solution concept; "monotonicity," which must exist in one sense or another if we are to have an objective metric of goodness, reveals that difficulty in obtaining an objective metric of performance can be due, in no small part, to the solution concept itself.

9.4 Empirical Results

Here we compare three solution concepts, Nash equilibrium, non-domination (non-monotonic version), and best-scoring strategy, by empirical results. Equation 9.10 shows a five-strategy, symmetric zero-sum game. Figure 9.9 shows the graph of the preference relation generated by this game using Nash equilibrium as the solution concept. Each "vertex" shows a configuration in bold and below it a list of games (each line is a separate game) for which the configuration is a solution. A directed edge that connects two vertices indicates the preference relation. For example, we prefer configuration $\{bde\}$ to configuration $\{ce\}$—each game for which $\{ce\}$ is a solution is a sub-game of some game for which $\{bde\}$ is a solution. Since the preference relation is transitive, we also prefer $\{bde\}$ to $\{e\}$, for example. The *depth* of a preference graph is the length of the longest (directed) path between any two vertices. Since the preference relation is transitive, this path must be between a vertex with in-degree of zero and another vertex with out-degree zero.

Figure 9.10 shows the preference relation graph generated by the same game except now we use non-domination as the solution concept. There are several points to notice. Most noticeably, Figure 9.10 has many more configurations than Figure 9.9—17 instead of 8. Also, the depth of the graph in Figure 9.10 is greater—four instead of three. The version of non-domination we use for this graph is non-monotonic, and indeed we see non-monotonic transitions from configuration $\{ad\}$ to configuration $\{d\}$; specifically, game $\{ad\}$ (for which $\{ad\}$ is the solution) is a supergame of $\{d\}$ (for which $\{d\}$ is the solution), and game $\{acd\}$ is a supergame of games $\{d\}$ and $\{cd\}$.

$$\mathbf{G} = \begin{array}{c|ccccc} & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} \\ \hline \text{a} & 0 & 0 & 1 & 0 & -1 \\ \text{b} & 0 & 0 & -1 & -1 & 1 \\ \text{c} & -1 & 1 & 0 & -1 & 0 \\ \text{d} & 0 & 1 & 1 & 0 & -1 \\ \text{e} & 1 & -1 & 0 & 1 & 0 \end{array} \tag{9.10}$$

Statistical Properties Above, we examine how two solution concepts, Nash equilibrium and non-dominated front (non-monotonic version), yield different preference relation graphs for the same game. Here, we generate 100 random seven-strategy and nine-strategy games each, and measure the number of configurations and the depth of the preference graph for each game for each of the three solution concepts we discuss above (Nash equilibrium, non-dominated front (non-monotonic version), and best-scoring strategy). Our random games are symmetric zero-sum games where each entry in the upper triangle of the payoff matrix is a 0, 1, or -1 with probability 1/3. Tables 9.3 and 9.4 summarize our results for the seven-strategy and nine-strategy games, respectively.

For both game sizes, the average number of configurations and average graph depth generated by a game is smallest when using the Nash solution concept and largest when using the non-dominated front, with best-scoring strategy in between; the data are normally distributed and all differences in average value are significant (0.01 significance level using paired Student's t-test). Further, the gaps between average values is much larger for nine-strategy games than for seven-strategy games, which suggests very large gaps for realistic games; of course, the number of possible subgames grows exponentially in the number of strategies, and so we should not be

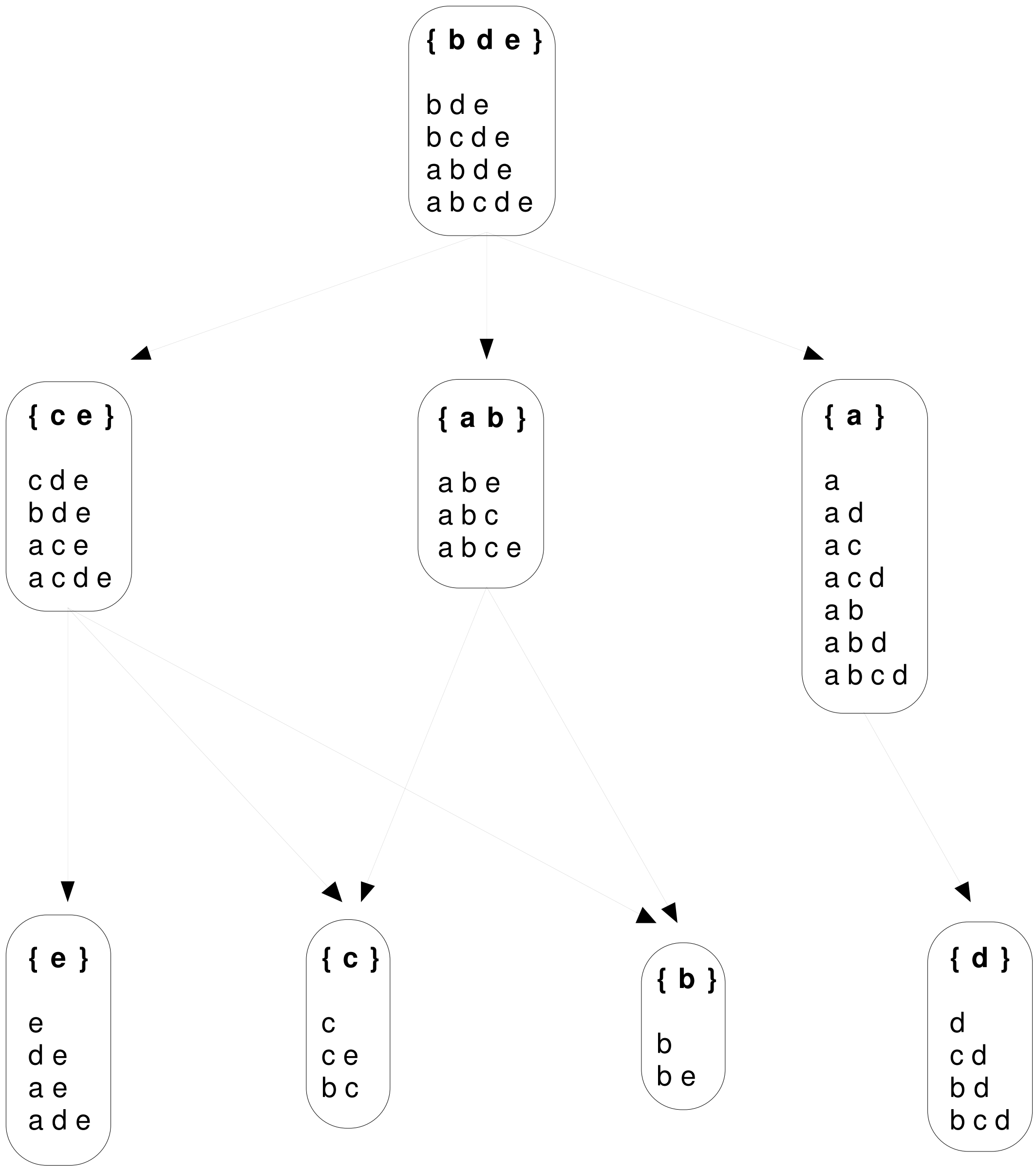


Figure 9.9: Preference diagram for game using Nash equilibrium.

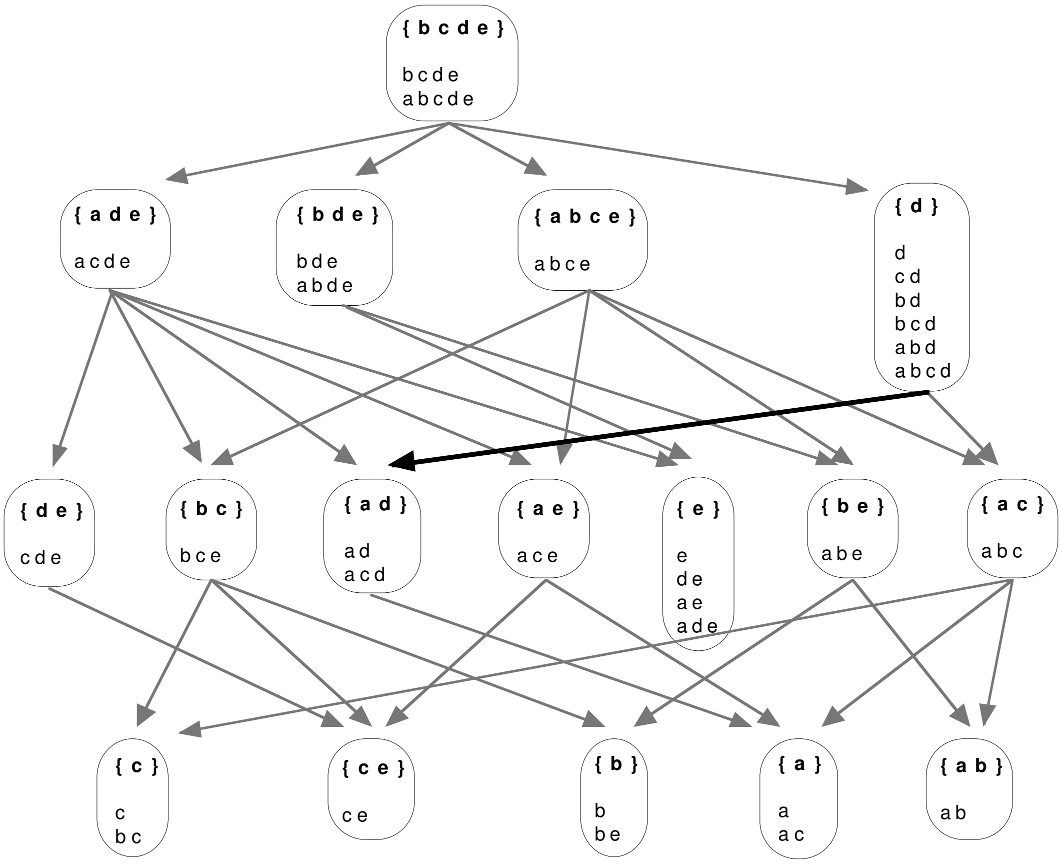


Figure 9.10: Preference diagram for game using non-domination.

	#Configurations		Correlation			Paired t-test value		
Concept	mean	std	nash	score	front	nash	score	front
Nash	16.25	2.64	1.00	0.46	0.63	n/a	18.56	27.67
Score	23.40	4.25	0.46	1.00	0.65	18.56	n/a	19.95
Front	37.60	9.10	0.63	0.65	1.00	27.67	19.95	n/a

	Graph Depth		Correlation			Paired t-test value		
Concept	mean	std	nash	score	front	nash	score	front
Nash	4.93	0.61	1.00	-0.04	-0.19	n/a	13.74	16.68
Score	6.19	0.66	-0.04	1.00	0.33	13.74	n/a	3.00
Front	6.40	0.53	-0.19	0.33	1.00	16.68	3.00	n/a

Table 9.3: Summary statistics for random games of seven strategies. Top: Number of configurations. Bottom: Preference graph depth.

	#Configurations		Correlation			Paired t-test value		
Concept	mean	std	nash	score	front	nash	score	front
Nash	33.12	6.35	1.00	0.77	0.79	n/a	28.61	29.84
Score	53.33	10.69	0.77	1.00	0.82	28.61	n/a	25.95
Front	122.69	34.75	0.79	0.82	1.00	29.84	25.95	n/a

	Graph Depth		Correlation			Paired t-test value		
Concept	mean	std	nash	score	front	nash	score	front
Nash	5.89	0.63	1.00	0.09	-0.06	n/a	17.77	28.00
Score	7.55	0.74	0.09	1.00	0.30	17.77	n/a	8.59
Front	8.20	0.49	-0.06	0.30	1.00	28.00	8.59	n/a

Table 9.4: Summary statistics for random games of nine strategies. Top: Number of configurations. Bottom: Preference graph depth.

surprised to find rapid growth in the number of configurations. Nevertheless, the Nash solution concept yields considerably fewer configurations to cover the same number of subgames, and so we may regard Nash configurations as more robust in this sense.

Figures 9.11 and 9.12 show the individual data points that are summarized in Tables 9.3 and 9.4, respectively. We sort the 100 games according to the number of configurations obtained from the Nash solution concept; each vertical triple of data points corresponds to data obtained from the different solution concepts for the same game. In this way, we can visualize the degree of correlation between the Nash concept and the other two. The figures clearly indicate the consistency with which the Nash concept generates the fewest configurations (and the non-dominated front the most) for each game; indeed, of the 200 games in these figures, there is only one instance—game 64 in Figure 9.11—where the non-dominated front does not generate the most configurations (the best-scoring concept does). We find more variability in the graph-depth data; in game 34 of Figure 9.11 the best-scoring concept creates the most shallow graph (Nash and non-dominated front tie for deepest graph).

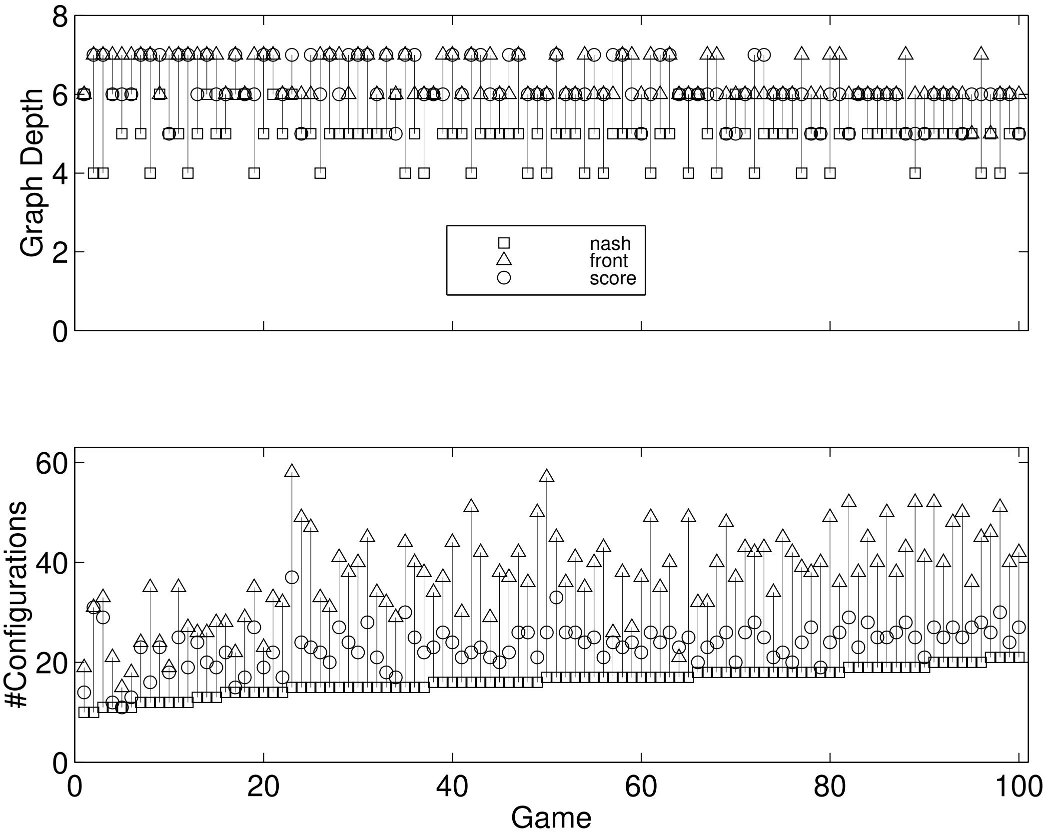


Figure 9.11: Graph depth and number of configurations of 100 random seven-strategy games.

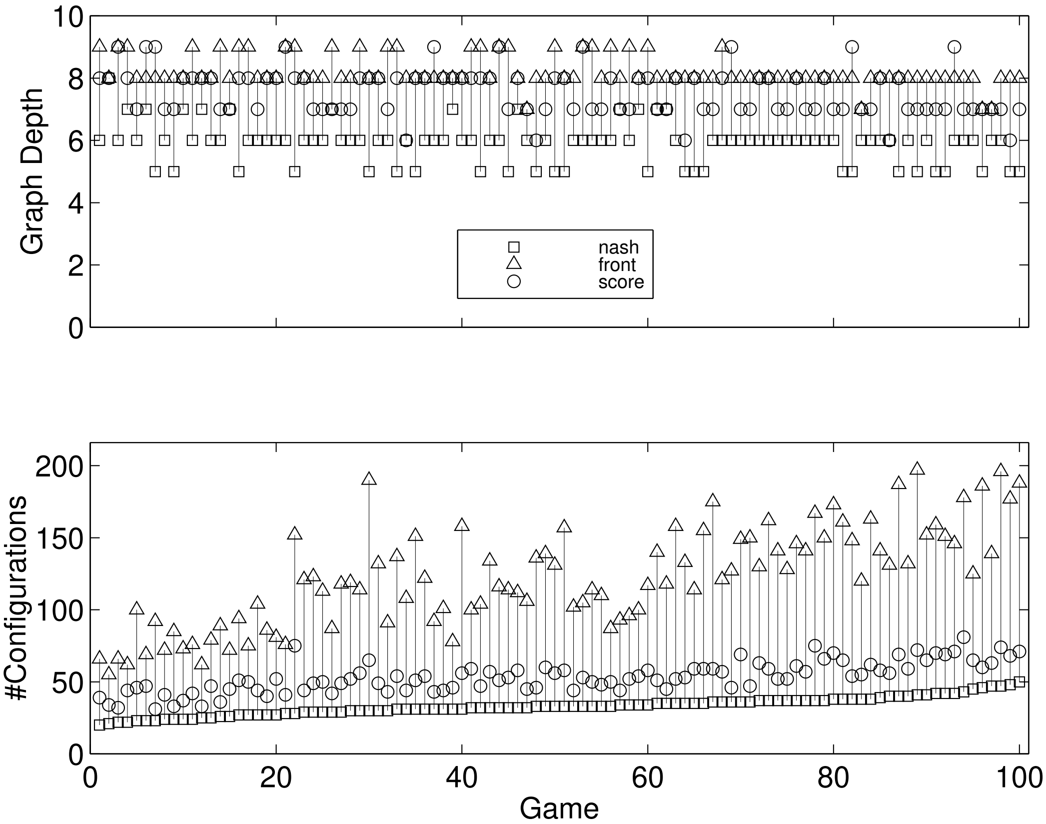


Figure 9.12: Graph depth and number of configurations of 100 random nine-strategy games.

9.5 Pathological Games

We show above that the non-domination solution concept is non-monotonic when identical-appearing strategies are allowed on the front. In this section, we show one way to construct games that highlight the non-monotonicity obtained with non-domination. Figure 9.13 illustrates the construction of such a game. We build a symmetric zero-sum game as that has $2k$ strategies. We divide the game into four sub-games, corresponding to the four sub-matrices indicated in the figure. Strategies s_1 through s_k are defined to tie each other, thus the entries of the upper left sub-matrix are all zero. The sub-game concerning strategies s_{k+1} through s_{2k} (lower right sub-matrix) may be any arbitrary zero-sum game, hence the "*" notation. The upper right sub-matrix records the scores obtained by strategies s_1 through s_k against strategies s_{k+1} through s_{2k}. For each strategy $s_{i=1\ldots k}$, we assign a unique combination of wins against strategies s_{k+1} through s_{2k}. For example, we can pick k of the $\begin{pmatrix} k \\ 2 \end{pmatrix}$ (k choose 2) two-win combinations and assign them to the k strategies s_1 through s_k. Finally, the lower left sub-matrix is the negative of the transpose of the upper right sub-matrix, which is required to make $\mathbf{G}$ a proper symmetric zero-sum game.

Given any two strategies s_i and s_j, where $i, j = 1 \ldots k$, we will be unable to distinguish them without one of the four strategies (of s_{k+1} through s_{2k}) that they beat being in our state of knowledge $\mathcal{W}$. Indeed, the game we construct above has many such pairs, which is what allows the non-monotonic property of non-domination to become acute. Equation 9.11 shows an example ten-strategy game built according to the method shown above. This game has 133 configurations, 65 of which—nearly half—have non-monotonic transitions. Further, many of these non-monotonic transitions lead to configurations that also have non-monotonic transitions. Figure 9.14

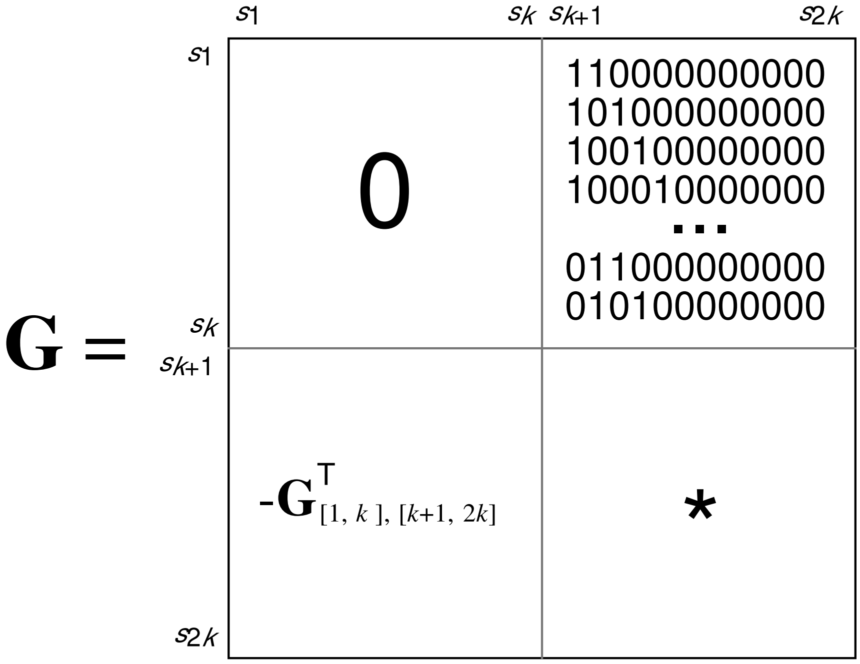


Figure 9.13: Construction of a game that highlights non-monotonicity of non-domination solution concept.

shows how many consecutive non-monotonic transitions are possible from each configuration. For our example game, there are two configurations (numbers 16 and 120) from which 23 consecutive non-monotonic transitions can be made. Thus, there exist sequences of states of knowledge, where each subsequent state is a proper superset of the previous one, that will nevertheless lead us to worse solutions over time.

$$
\mathbf{G} = \begin{array}{c|cccccccccc}
 & a & b & c & d & e & f & g & h & i & j \\
\hline
a & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
b & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
c & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
d & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
e & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
f & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 1 & -1 & 0 \\
g & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & -1 \\
h & 0 & -1 & 0 & 0 & -1 & -1 & -1 & 0 & -1 & -1 \\
i & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
j & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0
\end{array} \tag{9.11}
$$

9.6 Granularity of Gradient

A configuration $\mathcal{K}$ can be used as a static evaluation function for the evolution of strategies; all the strategies in the evolving population interact with $\mathcal{K}$, and the outcomes combined with the solution concept will provide some number of levels of fitness differential, or gradient, upon which selection can act. The question we investigate here concerns the amount of fitness differential that a solution concept can provide; the more levels of fitness that can be expressed by a solution concept, the more fine-grain the gradient. While evolution cannot operate without gradient, the "best" fitness granularity is likely to be a problem and substrate-dependent quantity. We content ourselves here with simply demonstrating that different solution concepts can have different granularity of gradient.

Our examination proceeds in the context of zero-sum games and in the absence

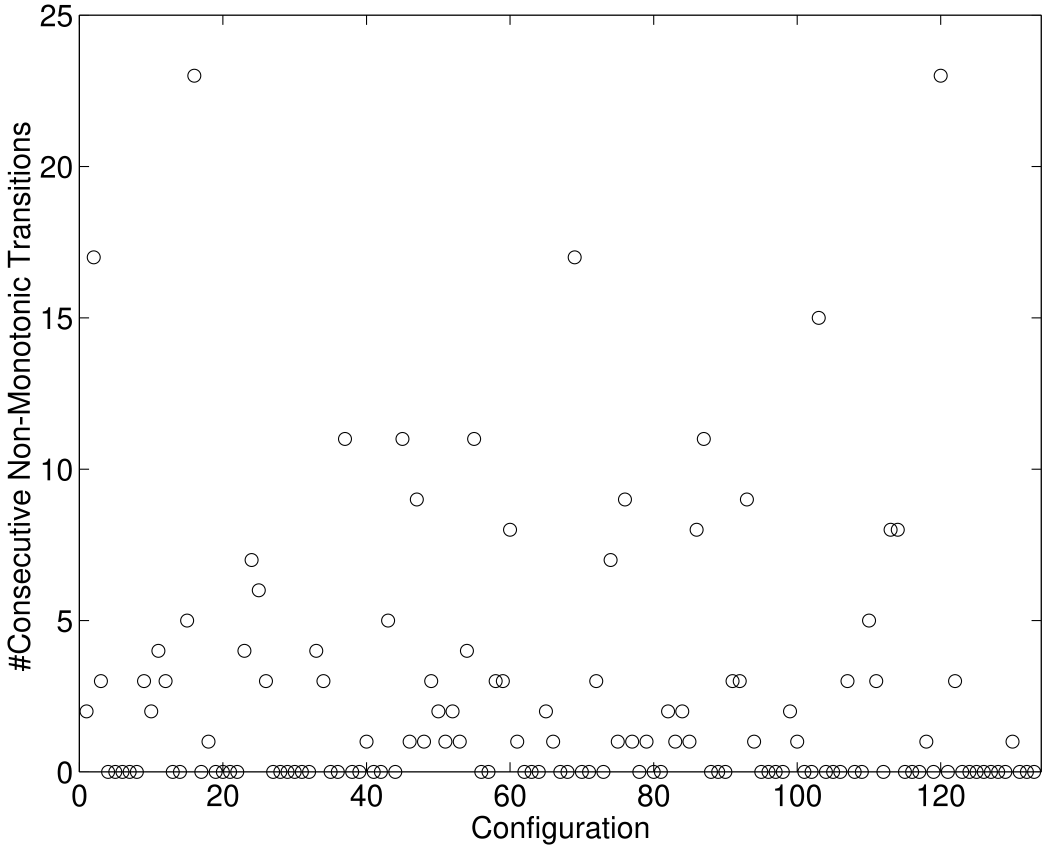


Figure 9.14: Number of consecutive non-monotonic transitions reachable from each configuration.

of fitness sharing, for simplicity. Let us assume that the configuration $\mathcal{K}$ used for evaluation is composed of n distinct strategies. For the best-scoring solution concept, we are interested to count the number of configuration strategies each member of the evolving population is able to defeat. Since there are n strategies in $\mathcal{K}$, a population member can achieve a score no less than zero and no more than n; thus, we have at most $n + 1$ distinct fitness levels.

If our solution concept is non-domination, then we have at least two ways we can score members of the evolving population. One way to score members is according to the Pareto layer in which they belong; the individuals on the front $\mathcal{F}^0$ receive the highest score, those only dominated by members of $\mathcal{F}^0$ are in $\mathcal{F}^1$ and receive the next highest score, and so on. Thus, the number of distinct fitness values depends upon on the number of Pareto layers we can have; given n dimensions of comparison, we can have at most n layers. A second scoring method is to assign an individual a score based upon the number of other individuals in the population that it dominates. In this case, the number of fitness levels clearly depends upon the population size.

If our solution concept is Nash equilibrium, then the maximal number of fitness levels we can obtain depends upon the probability distribution over the n strategies in configuration $\mathcal{K}$. If the distribution is uniform, then we can have no more than n fitness values. On the other hand, if no probability value in the distribution can be obtained through the summation and subtraction of the other probability values in the distribution, then the maximal number of distinct fitness values is 2^n. That is, there are 2^n distinct win/loss profiles that we can obtain when we play the n strategies in $\mathcal{K}$, and each one of these win/loss profiles generates a unique fitness value.

9.7 Monotonicity and Open-Ended Domains

Much research in coevolution concerns how a coevolutionary dynamic may generate an *open-ended* system, where novel and increasingly complex behaviors are continually innovated. Examples of such research include the evolution of language [90, 60], the Iterated Prisoner's Dilemma [128, 98], competitive games in virtual worlds [185, 96], artificial chemistries [77], and complex A-Life ecologies [168, 107, 125, 189]. Nevertheless, a precise articulation of open-endedness is made difficult by, among other things, the lack of a universally agreed-upon notion of complexity [2, 40, 114]. From the point of view of the framework developed in this chapter, we can contribute the following thoughts.

The game, as we formally define it here, is merely a function that maps n-tuples of strategies to vectors of payoffs; in particular, the game represents nothing about the *behavior* of a strategy, other than the payoffs it earns—a highly abstracted view of behavior, indeed. Thus, with respect to the theme of open-endedness, we can infer only certain things from the game. For example, if the game is finite, then clearly the domain (or, more accurately, that part of the domain exposed by the evolving substrate) cannot be open-ended—there are only a finite number of strategies defined. Tic-tac-toe is an example of a closed-ended domain. Other games, such as Watson and Pollack's numbers games [202], are open-ended at least in the sense that the domain allows an infinity of strategies; nevertheless, for any finite set of numbers-game strategies, we can always find another strategy that will beat all of them, yet this winning strategy will appear no more complex than the others.

A more compelling form of open-endedness entails the generation of novel and unexpected solutions to the problems posed by a coevolving population—open-endedness is a sequence of set-breaking tasks. Innovative solutions not only have intrinsic interest, but also reveal the domain to which they belong to have more nuance than previously imagined. We can thus make a simple taxonomy of domains: Closed-ended domains have a finite space of behaviors (strategies); *impoverished* open-ended domains have an infinity of behaviors, but these behaviors can be systematized in finite space—i.e., the "rules of the game" are finite; *rich* open-ended domains not only have an infinity of behaviors, but also the systematization of the domain requires infinite space.

The idea of rich open-endedness appears to leave little room for objective metrics of goodness, since every new innovation changes the prism through which we understand the domain. For example, each innovation may uncover another intransitive structure in the domain; a strategy previously understood as "poor" may regain respectability because we realize that an entire aspect of its behavior was overlooked. But, we should not conflate behavioral complexity with adaptive utility. Whatever our perception of a strategy's behavior, the game of our formalism does not distinguish between impoverished and rich open-endedness, nor for that matter between closed-endedness and open-endedness.

Since our formalizations of solution concept and monotonicity are built on the game, and not the domain's behaviors, our results concerning monotonicity apply equally to all domains, closed-ended and open-ended, impoverished and rich. If we do not discard information, then the solutions we obtain at time t are guaranteed to be no worse, in an objective (i.e., global) sense, than solutions obtained earlier; and, this is true *regardless of the fact that we have only local knowledge of the domain and no knowledge of what strategies we might discover in the future.* If our solution concept is non-monotonic, then no such guarantee can be made, even for closed-ended domains.

Chapter 10

Conclusions

This dissertation demonstrates the importance of solution concepts in the design, operation, and understanding of coevolutionary algorithms. We show in Chapter 1 that solution concepts interface optimization problems with optimization methods; favorable outcomes result when the solution concept implemented by the optimization method is consistent with that required by the optimization problem. While this point may seem obvious, years of coevolutionary practice indicate otherwise.

Being the algorithmic progeny of "ordinary" evolutionary methods (which optimize fixed objective functions), *co*-evolutionary algorithms inherit not only a variety of procedural traits but also (and more importantly) *a custom of use*. In particular, we typically view coevolutionary algorithms as population-based search methods to find an *individual*; solutions are generally not conceived to be collectives. Many of the search problems that we attempt to solve with coevolution reveal a variety of difficulties; this dissertation takes the position that these pathologies (which we review in Chapter 3) result directly from incompatibilities between the solution concepts required by coevolutionary search problems and those implemented by methodologies we inherit from earlier evolutionary optimization practice.

From the point of view of compatibility with polymorphic Nash equilibria, Chapters 4 through 6 examine the behaviors of various mechanisms that are frequently used in coevolutionary algorithms. Chapter 4 shows that several selection methods, commonly used in "ordinary" evolution, do not transfer with transparency to the coevolutionary setting. Using an evolutionary game-theoretic framework, we show that the selection methods we test prevent an evolving population from attaining polymorphic Nash equilibria. Instead, a variety of other behaviors emerge, including oscillation and even chaos. Along the lines of Chapter 4, Chapter 5 shows that fitness sharing methods, used to enhance genotypic and phenotypic population diversity, distort polymorphic Nash equilibria. Chapter 6 shows that even a standard finite-population replicator can distort polymorphic Nash equilibria due to the interaction of sampling effects with asymmetric dynamics of convergence; an alternative replicator (Baker's [20] SUS) is shown to dramatically reduce the effect. Thus, each of these chapters provides a specific actionable result. More generally, a conclusion that emerges from these chapters as a whole is that techniques used to improve the population's ability to perform search may interfere with its ability to represent the solution.

We assert that pathologies in coevolutionary optimization arise when algorithms fail to implement the required (or desired) solution concepts. Accordingly, in Chapters 7 and 8, we design two novel heuristics around particular solution concepts in an effort to address certain pathological outcomes. Chapter 7 introduces the use of Pareto optimality as a solution concept for coevolutionary algorithms. We propose a matched-pair of solution concepts: Pareto optimality for our primary search effort, and our orthogonal concept of *distinctions* for our secondary search effort. Our approach, known as *Pareto coevolution* addresses the issues of gradient creation and maintenance. Using this method, we discover an effective cellular automaton rule for a density classification task. Chapter 8 uses the concept of Nash equilibrium to build a new type of memory mechanism that addresses the issue of evolutionary forgetting. Using Watson and Pollack's [202] intransitive numbers game, we demonstrate the ability of our Nash memory to handle intransitive structures, systematically broaden selection pressure, and provide an effective elitism scheme that also cleanly represents our ultimate solution.

Building on the framework of Chapter 2, Chapter 9 takes a novel view at solution concepts with respect to a property we call *monotonicity.* Assume an arbitrary search algorithm that implements some solution concept $\mathcal{O}$ and has infinite memory. We hope that, as it operates and accumulates knowledge of the search space, the algorithm forms monotonically better answers to our search problem over time. We define a preference relation into which any solution concept $\mathcal{O}$ may be embedded; this preference relation defines a partial order over the space of possible answers to a search problem. We then show that, as our algorithm operates and refines its answer in response to new information, the sequence of refinements generated by our algorithm may or may not contradict our partial ordering *dependent solely upon properties of the solution concept itself.* We prove that some solution concepts have this monotonic property (e.g., Nash equilibrium), while others do only subject to particular implementation details (e.g., Pareto optimality), and yet others lack the property (e.g., the solution concept used in conventional coevolutionary algorithms). Finally, our results hold even if the domain in question is open-ended.

To summarize, in Chapters 1 through 3 we discuss the importance of solution concepts to our understanding of coevolutionary algorithm operation and design. We demonstrate in Chapters 4 through 6 how solution concepts help explain the divergence between expected and actual algorithm outcomes, bringing our expectations of coevolution more in line with algorithmic reality. In Chapters 7 and 8 we demonstrate how solution concepts can guide the design of algorithms, and thereby bring algorithm operation more in line with our expectations and goals. Finally, in Chapter 9 we show that solution concepts have intrinsic properties that cause (or prevent) the dynamics of search to conform to a static, global partial-order over the space of possible results.

Bibliography

[1] C. Adami. On modelling life. *Artificial Life*, 1:429–438, 1994. 1.5.8

[2] C. Adami. What is complexity? *BioEssays*, 24(12):1085–1094, 2002. 9.7

[3] C. Adami and C. T. Brown. Evolutionary learning in the 2d artificial life system "Avida". In Brooks and Maes [27], pages 377–381. 1.5.8

[4] J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and S. D., editors. *Artificial Evolution (AE 95)*. Springer-Verlag, 1995. 17, 137

[5] D. Andre et al. Evolution of intricate long-distance communication signals in cellular automata using genetic programming. In C. G. Langton and K. Shimohara, editors, *Artificial Life V*, pages 513–520. MIT Press, 1996. 3.2.1, 7.3.1, 7.5.3

[6] D. Andre, F. H. B. III, and J. R. Koza. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 3–11. MIT Press, 1996. 3.2.1, 7.1, 7.3.1, 7.5.3

[7] P. Angeline, G. Saunders, and J. Pollack. An evolutionary algorithm that constructs recurrent networks. *IEEE Transactions on Neural Networks*, 5:54–65, 1994. 2.7

[8] P. J. Angeline. Competitive fitness. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C4.3. Institute of Physics Publishing, Bristol, UK, 1997. 3.1

[9] P. J. Angeline and J. B. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, 1993. 1.3, 1.4.1, 2.7, 3.2.2, 3.2.2, 3.2.2, 3.2.4

[10] W. B. Arthur. Inductive reasoning and bounded rationality. *American Economics Association Papers Proceedings*, 84:406–411, 1994. 1.5.8

[11] W. B. Arthur. Complexity and the economy. *Science*, 284:107–109, April 1999. 1.5.8

[12] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In A. Borodin and P. Raghavan, editors, *36th Annual Symposium on Foundations of Computer Science (FOCS '95)*, pages 322–331. IEEE Press, 1995. 1.5.5

[13] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984. 4.10

[14] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 32–41. Morgan Kaufmann, 1987. 1.4.1, 1.4.2, 4.10

[15] R. Axtell. The emergence of firms in a population of agents: Local increasing returns, unstable nash equilibria, and power law size distributions. Technical Report Working Paper No. 3, Center on Social and Economic Dynamics, The Brookings Institution, June 1999. 1.5.8

[16] R. Axtell. The emergence of institutions of self-governance on the commons. In *Conference on Computational Political Economy*. University of Michigan, 2003. 1.5.8

[17] T. Bäck. Evolution strategies: An alternative evolutionary algorithm. In Alliot et al. [4], pages 3–20. 4.5

[18] T. Bäck et al., editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, 2000. 1.4.1

[19] T. Bäck et al., editors. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing, 2000. 1.4.1

[20] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21. Lawrence Earlbaum Associates, 1987. 1.6, 6.1, 6.7, 6.4, 7.4, 10

[21] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming: An Introduction On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann, 1997. 1.4.1

[22] L. Barnett. *Evolutionary Search on Fitness Landscapes with Neutral Networks*. PhD thesis, University of Sussex, 2003. 3.4.1

[23] A. Blum and M. Kearns, editors. *Proceedings of the Ninth Annual ACM Conference on Computational Learning Theory (COLT 1996)*. ACM Press, 1996. 79, 174

[24] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002. 1.5.6

[25] R. Boyd. Mistakes allow evolutionary stability in the repeated prisoner's dilemma game. *Journal of Theoretical Biology*, 136:47–56, 1989. 3.6.2

[26] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2001. 2.10.6

[27] R. A. Brooks and P. Maes, editors. *Artificial Life IV*. MIT Press, 1994. 3, 170, 185

[28] G. W. Brown. Iterative solutions of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, 1951. 1.5.5

[29] A. Bucci and J. B. Pollack. Order-theoretic analysis of coevolution problems: Coevolutionary statics. In A. M. Barry, editor, *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 229–235, 2002. 3.4.2, 7.2.3, 7.6

[30] A. Bucci and J. B. Pollack. A mathematical framework for the study of coevolution. In De Jong et al. [49], pages 221–235. 3.4.2, 3.8

[31] S. Bullock. Co-evolutionary design: Implications for evolutionary robotics. Technical Report CSRP 384, School of Cognitive and Computing Sciences, University of Sussex, 1995. Presented as a poster at the Third European Conference on Artificial Life (ECAL 1995). 3.5.2, 3.8

[32] Cantú-Paz et al., editors. *2003 Genetic and Evolutionary Computation Conference*. Springer, 2003. 45, 65, 182

[33] J. Cartlidge and S. Bullock. Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. In Fogel et al. [72], pages 1420–1425. 3.4.2

[34] K. Chellapilla and D. Fogel. Evolving neural networks to play checkers without expert knowledge. *IEEE Transactions on Neural Networks*, 10(6):1382–1391, 1999. 3.2.2, 3.2.4

[35] K. Chellapilla and D. B. Fogel. Anaconda defeats Hoyle 6-0: A case study competing an evolved checkers program against commercially available software. In Zalzala et al. [209], pages 857–863. 3.2.2, 3.2.4

[36] D. Cliff and G. F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In Morán et al. [144], pages 200–218. 3.3, 3.5.1, 3.6.1, 3.6.2, 7.1

[37] D. Cliff and G. F. Miller. Co-evolution of pursuit and evasion II: Simulation methods and results. In Maes et al. [132], pages 506–515. 3.3, 3.6.2

[38] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In T. Fawcett and N. Mishra, editors, *Machine Learning: Proceedings of the Twentieth International Conference (ICML 2003)*, pages 83–90. AAAI Press, 2003. 1.5.5

[39] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In J. J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms (1985 Carnegie-Mellon University)*, pages 183–187. Lawrence Earlbaum Associates, 1988. 2.7

[40] J. P. Crutchfield. The calculi of emergence: Computation, dynamics and induction. *Physica D*, 75(1–3):11–54, 1994. 9.7

[41] P. Darwen. *Co-evolutionary Learning by Automatic Modularisation with Speciation.* PhD thesis, University of New South Wales, 1996. 1.4.6

[42] P. Darwen and X. Yao. Automatic modularization by speciation. In T. Fukuda, T. Furuhashi, T. Bäck, H. Kitano, and Z. Michalewicz, editors, *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 88–93. IEEE Press, 1996. 1.4.6

[43] R. Das et al. A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor et al., editors, *Parallel Problem Solving from Nature 3*, pages 344–353. Springer, 1994. 3.2.1

[44] H. Dawid. *Adaptive Learning by Genetic Algorithms: Analytical Results and Applications to Economic Models.* Springer, second edition, 1999. First Edition 1996. 3.7

[45] E. D. de Jong and J. B. Pollack. Learning the ideal evaluation function. In Cantú-Paz et al. [32], pages 277–288. 3.4.2, 3.8, 7.2.3, 7.6

[46] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Doctoral thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975. 1.4.5

[47] K. A. De Jong. Are genetic algorithms function optimizers? In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 3–13. Elsevier, 1992. 1.4.5

[48] K. A. De Jong. Genetic algorithms are NOT function optimizers. In L. D. Whitely, editor, *Foundations of Genetic Algorithms 2 (FOGA)*, pages 5–18. Morgan Kaufmann, 1993. 1.4.5

[49] K. A. De Jong, R. Poli, and J. E. Rowe, editors. *Proceedings of the Foundations of Genetic Algorithms 2003 Workshop (FOGA 7)*. Morgan Kaufmann Publishers, 2003. 30, 207

[50] E. A. Di Paolo. *On the Evolutionary and Behavioral Dynamics of Social Coordination: Models and Theoretical Aspects*. PhD thesis, University of Sussex, 1999. 1.5.8

[51] E. A. Di Paolo. Behavioral coordination, structural congruence and entrainment in a simulation of acoustically coupled agents. *Adaptive Behavior*, 8(1):27–48, 2000. 1.5.8

[52] U. Dieckmann and M. Doebeli. On the origin of species by sympatric speciation. *Nature*, 400:354 –357, July 1999. 1.5.8

[53] B. Dolin, F. H. B. III, and E. G. Rieffel. Co-evolving an effective fitness sample: Experiments in symbolic regression and distributed robot control. In G. B. Lamont, H. Haddad, G. Papadopoulos, and B. Panda, editors, *Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 553–559. ACM Press, 2002. 7.6, 7.2, 7.3

[54] R. W. Easton. *Geometric methods for discrete dynamical systems*. Oxford University Press, 1998. 4.9

[55] S. L. Epstein. Toward an ideal trainer. *Machine Learning*, 15(3):251–277, 1994. 3.8

[56] L. J. Eshelman, editor. *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995. 99, 173

[57] F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems*. Springer, 2003. 1.5.2

[58] S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic investigation of selection methods used in evolutionary algorithms. In Zalzala et al. [209], pages 880–887. 1.4.2, 1.6, 4.1

[59] S. G. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In C. Adami et al., editors, *Proceedings of the Sixth Conference on Artificial Life*, pages 238–247. MIT Press, 1998. 3.3, 3.4.1, 3.5.1

[60] S. G. Ficici and J. B. Pollack. Coevolving communicative behavior in a linear pursuer-evader game. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pages 505–514. MIT Press, 1998. 1.5.8, 3.3, 3.4.1, 3.5.1, 9.7

[61] S. G. Ficici and J. B. Pollack. Effects of finite populations on evolutionary stable strategies. In L. D. Whitley et al., editors, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, pages 927–934. Morgan-Kaufmann, 2000. 1.6, 6.1

[62] S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In Schoenauer et al. [184], pages 467–476. 1.4.2, 3.7, 7.1

[63] S. G. Ficici and J. B. Pollack. Game theory and the simple coevolutionary algorithm: Some results on fitness sharing. In R. Heckendorn, editor, *2001 Genetic and Evolutionary Computation Conference Workshop Program*, pages 2–7, 2001. 1.6, 5.1

[64] S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. In J. Kelemen and P. Sosík, editors, *Sixth European Conference on Artificial Life (ECAL 2001)*, pages 316–325. Springer, 2001. 1.5.7, 1.6, 3.4.2, 7.1

[65] S. G. Ficici and J. B. Pollack. A game-theoretic memory mechanism for coevolution. In Cantú-Paz et al. [32], pages 286–297. 1.6, 3.6.2, 8.1

[66] R. Fisher. *The genetical theory of natural selection*. Clarendon Press, Oxford, 1930. 1.5.8

[67] D. Floreano and S. Nolfi. Adaptive behavior in competing co-evolving species. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life*, pages 378–387. MIT Press, 1997. 3.3, 3.6.2

[68] D. Floreano and S. Nolfi. God save the Red Queen! Competition in coevolutionary robotics. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Proceedings of the Second Conference on Genetic Programming*, pages 398–406. Morgan Kaufmann, 1997. 3.3, 3.6.1, 3.6.2

[69] D. Floreano, S. Nolfi, and F. Mondada. Competitive co-evolutionary robotics: From theory to practice. In R. Pfeifer et al., editors, *From Animals to Animats V*, pages 515–524. MIT Press, 1998. 3.6.2

[70] D. Floreano, S. Nolfi, and F. Mondada. Co-evolution and ontogenetic change in competing robots. In M. Patel, V. Honavar, and K. Balakrishnan, editors, *Advances in the Evolutionary Synthesis of Intelligent Agents*, chapter 10. MIT Press, 2001. 3.6.2

[71] D. B. Fogel. An overview of evolutionary programming. In L. D. Davis, K. De Jong, M. D. Vose, and L. D. Whitley, editors, *Evolutionary Algorithms*, pages 89–109. Springer, 1997. 4.4

[72] D. B. Fogel, M. A. El-Sharkawi, and X. Yao, editors. *Proceedings of the 2002 Congress on Evolutionary Computation*. IEEE Press, 2002. 33, 206

[73] D. B. Fogel and G. B. Fogel. Evolutionary stable strategies are not always stable under evolutionary dynamics. In *Evolutionary Programming IV*, pages 565–577, 1995. 1.6, 4.10, 6.1, 6.2, 6.3, 6.3.1

[74] D. B. Fogel, G. B. Fogel, and P. C. Andrews. On the instability of evolutionary stable states. *BioSystems*, 44:135–152, 1997. 1.6, 4.10, 6.1, 6.2, 6.3, 6.3.1, 6.3.2, 6.3.2

[75] G. B. Fogel, P. C. Andrews, and D. B. Fogel. On the instability of evolutionary stable strategies in small populations. *Ecological Modelling*, 109:283–294, 1998. 1.6, 4.10, 6.1, 6.2, 6.3, 6.3.1

[76] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995. 1.4.6, 7.2.2

[77] W. Fontana. Algorithmic chemistry. In Langton et al. [120], pages 159–209. 9.7

[78] H. Freund and R. Wolter. Evolution of bit strings: Some preliminary results. *Complex Systems*, 5:279–298, 1991. 1.4.5

[79] Y. Freund and R. E. Schapire. Game theory, on-line prediction and boosting. In Blum and Kearns [23], pages 325–332. 1.5.5

[80] Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999. 1.5.5

[81] D. Friedman. On economic applications of evolutionary game theory. *Journal of Evolutionary Economics*, 8:15–43, 1998. 1.5.4

[82] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, 1998. 1.4.2, 1.5.4, 1.5.5, 3.2.1, 3.4.1, 3.5

[83] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1998. 1.1, 1.5.1, 2.6.5, 4.2.2, 8.2

[84] H. Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2000. 1.5.4

[85] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989. 1.4.1, 1.4.5, 2.7, 3.7.2, 4.6, 5.1, 5.4, 8.1

[86] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. Rawlins, editor, *Foundations of Genetic Algorithms (FOGA 1)*, pages 69–93, 1991. 4.1

[87] P. Hammerstein. Darwinian adaptation, population genetics and the streetcar theory of evolution. *Journal of Mathematical Biology*, 34:511–532, 1996. 3.7.2

[88] P. J. Hancock. An empirical comparison of selection methods in evolutionary algorithms. In T. C. Fogarty, editor, *Evolutionary Computing (AISB '94)*, pages 80–94, 1994. 4.1

[89] I. Harvey. Cognition is not computation: Evolution is not optimisation. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN97)*, pages 685–690. Springer-Verlag, 1997. 1.4.5

[90] T. Hashimoto and T. Ikegami. Emergence of net-grammar in communicating agents. *BioSystems*, 38(1):1–14, 1996. 9.7

[91] D. Hillis. Co-evolving parasites improves simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990. Reprinted in [92]. 1.1, 1.4.1, 3.2.1, 3.2.2, 3.2.2, 3.8

[92] D. Hillis. Co-evolving parasites improves simulated evolution as an optimization procedure. In Langton et al. [120], pages 313–324. 2.2.2, 2.7, 3.2.1, 3.2.2, 3.4.1, 7.1, 91

[93] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998. 1.5.4, 3.5, 4.1, 4.2

[94] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, reprint edition, 1992. Original edition University of Michigan Press, 1975. 1.4.5

[95] J. H. Holland. Echoing emergence: Objectives, rough definitions, and speculations for echo-class models. In G. A. Cowan, D. Pines, and D. Meltzer, editors, *Complexity: Metaphors, models and reality*, Volume XIX of Santa Fe Institute Studies in the Sciences of Complexity, pages 309–342. Addison-Wesley, 1994. 1.5.8

[96] G. Hornby and B. Mirtich. Diffuse versus true coevolution in a physics-based world. In Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, and Smith, editors, *Proceedings of 1999 Genetic and Evolutionary Computation Conference*, pages 1305–1312. Morgan Kaufmann, 1999. 3.5.2, 3.6.2, 3.8, 9.7

[97] P. T. Hraber, T. Jones, and S. Forrest. The ecology of echo. *Artificial Life*, 3(3):165–190, 1997. 1.5.8

[98] T. Ikegami. From genetic evolution to emergence of game strategies. *Physica D*, 75(1–3):310–327, 1994. 9.7

[99] H. Juillé. Evolution of non-deterministic incremental algorithms as a new approach for search in state spaces. In Eshelman [56], pages 351–358. 2.7, 3.2.1

[100] H. Juillé. Incremental co-evolution of organisms: A new approach for optimization and discovery of strategies. In Morán et al. [144], pages 246–260. 2.2.2

[101] H. Juillé. *Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm*. PhD thesis, Brandeis University, 1999. 7.2.1, 7.5.3

[102] H. Juillé and J. Pollack. Co-evolving intertwined spirals. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 461–468. MIT Press, 1996. 2.10.4

[103] H. Juillé and J. Pollack. Dynamics of co-evolutionary learning. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 526–534. MIT Press, 1996. 3.8, 7.2.2

[104] H. Juillé and J. B. Pollack. Coevolutionary learning: A case study. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 251–259. Morgan Kaufmann, 1998. 1.6, 2.2.2, 2.7, 3.2.1, 3.4.1, 3.4.2, 3.5.1, 7.1, 7.3.1, 7.3.2, 7.3.3, 7.5.3

[105] H. Juillé and J. B. Pollack. Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. In J. R. Koza et al., editors, *Proceedings of the Third Annual Genetic Programming Conference*, pages 519–527. Morgan Kaufmann, 1998. 1.6, 3.2.1, 3.4.1, 3.4.2, 3.5.1, 3.8, 7.1, 7.3.1, 7.3.2, 7.3.3, 7.5.3

[106] H. Juillé and J. B. Pollack. Coevolutionary learning and the design of complex systems. *Advances in Complex Systems*, 2(4):371–393, 2000. 1.6, 3.2.1, 3.2.2, 3.4.1, 3.4.2, 3.5.1, 3.8, 7.3.1

[107] K. Kaneko and I. Tsuda. Constructive complexity and artificial reality: An introduction. *Physica D*, 75(1–3):1–10, 1994. 9.7

[108] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993. 1.4.2

[109] M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In J. Breese, D. Koller, and M. Goldszmidt, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UIA-2001)*, pages 253–260. Morgan Kaufmann, 2001. 1.5.3

[110] M. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994. 1.5.7

[111] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. 1.5.7

[112] M. Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1983. 3.4.1

[113] D. Knuth. *Sorting and Searching*. The Art of Computer Programming, vol. 3. Addison-Wesley, 2nd edition, 1998. 3.2.1

[114] J. F. Kolen and J. B. Pollack. The observer's paradox: Apparent computational complexity in physical systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:253–277, 1995. 9.7

[115] D. Koller, N. Megiddot, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In F. T. Leighton and M. Goodrich, editors, *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC 1994)*, pages 750–759. ACM Press, 1994. 1.5.3

[116] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 1027–1034. Morgan Kaufmann, 2001. 1.5.3

[117] A. S. Kondrashov and M. Shpak. On the origin of species by means of assortative mating. *Proceedings of the Royal Society of London B*, 265:2278–2273, 1998. 1.5.8

[118] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992. 2.7

[119] M. Land and R. K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74(25):1548–1550, 1995. 2.2.2, 2.7, 3.2.1, 7.1, 7.3.1, 7.3.2

[120] C. Langton, C. Taylor, J. Farmer, and S. Rasmussen, editors. *Artificial Life II: Proceedings of the 1990 Workshop on Artificial Life*, SFI Studies in the Sciences of Complexity, vol. X. Addison-Wesley, 1992. 77, 92, 128, 205

[121] B. LeBaron. A builder's guide to agent-based financial markets. *Quantitative Finance*, 1:254–261, 2001. 1.5.8

[122] B. LeBaron. Financial market efficiency in a coevolutionary environment. In *Proceedings of the Workshop on Simulation of Social Agents: Architectures and Institutions*, pages 33–51. Argonne National Laboratory and The University of Chicago, 2001. 1.5.8

[123] B. LeBaron, W. B. Arthur, and R. Palmer. The time series properties of an artificial stock market. *Journal of Economic Dynamics and Control*, 23:1487–1516, 1999. 1.5.8

[124] C. E. Lemke and J. J. T. Howson. Equilibrium points in bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12:413–423, 1964. 1.5.2

[125] R. E. Lenski, C. Ofria, T. C. Collier, and C. Adami. Genome complexity, robustness and genetic interactions in digital organisms. *Nature*, 400(6745):661–664, August 1999. 9.7

[126] S. Lessard. Evolutionary stability: One concept, several meanings. *Theoretical Population Biology*, 37:159–170, 1990. 1.5.4, 4.2

[127] A. Lindenmayer. Mathematical models for cellular interaction in development. Parts I and II. *Journal of Theoretical Biology*, 18:280–299 and 300–315, 1968. 2.7

[128] K. Lindgren. Evolutionary phenomena in simple dynamics. In Langton et al. [120], pages 295–312. 1.4.2, 9.7

[129] H. Lipson and P. J. B. Automatic design and manufacture of artificial lifeforms. *Nature*, 406:974–978, 2000. 3.2.4

[130] M. L. Littman. Markov games as a framework for multiagent reinforcement learning. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 157–163. Morgan Kaufmann, 1994. 1.5.6

[131] S. Luke and R. P. Wiegand. When coevolutionary algorithms exhibit evolutionary dynamics. In A. Barry, editor, *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 236–241, 2002. 9.3.3

[132] P. Maes et al., editors. *From Animals to Animats IV*. MIT Press, 1996. 37, 181

[133] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory.* Oxford University Press, 1995. 1.5.6

[134] J. Maynard-Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982. 1.1, 1.4.1, 1.4.6, 1.5.4, 1.6, 3.7.2, 4.1, 4.2, 4.3, 6.1, 6.2

[135] J. Maynard-Smith and G. R. Price. The logic of animal conflict. *Nature*, 246:15–18, 1973. 1.5.4

[136] R. D. McKelvey and A. McLennan. Computation of equilibria in finite games. In H. M. Amman, D. A. Kendrick, and J. Rust, editors, *Handbook of Computational Economics*, volume 1, chapter 2, pages 87–142. Elsevier, 1996. 1.5.2

[137] N. Meuleau and C. Lattaud. The artificial evolution of cooperation. In Alliot et al. [4], pages 159–180. 4.10

[138] G. Miller and P. Todd. Evolutionary wanderlust: Sexual selection with directional mate preferences. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From Animals to Animats II (1992)*, pages 21–30. MIT Press, 1993. 1.5.8

[139] G. Miller and P. Todd. The role of mate choice in biocomputation: Sexual selection as a process of search, optimization, and diversification. In W. Banzhaf and F. Eeckman, editors, *Evolution and biocomputation: Computational models of evolution*, pages 169–204. Springer, 1995. 1.5.8

[140] G. F. Miller and D. Cliff. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats III*, pages 411–420. MIT Press, 1994. 3.3

[141] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996. 1.4.1, 4.1, 4.7

[142] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391, 1994. 2.2.2, 2.7, 3.2.1, 7.1, 7.3.1, 7.3.2

[143] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993. 3.2.1

[144] F. Morán et al., editors. *Proceedings of the Third European Conference on Artificial Life*. Springer, 1995. 36, 100

[145] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951. Second Series. 1.5.1, 4.2.2

[146] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001. 1.5.6

[147] J. Noble and R. A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In L. Spector et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 493–500. Morgan Kaufmann, 2001. 7.1

[148] S. Nolfi and D. Floreano. Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998. 1.4.6, 3.5.1, 3.5.2, 3.6.2, 8.4, 8.8

[149] S. Nolfi and D. Floreano. How co-evolution can enhance the adaptation power of artificial evolution: Implications for evolutionary robotics. In P. Husbands and J.-A. Meyer, editors, *Proceedings of the First European Workshop on Evolutionary Robotics (EvoRobot98)*, pages 22–38. Springer, 1998. 3.6.2

[150] M. Nowak. An evolutionary stable strategy may be inaccessible. *Journal of Theoretical Biology*, 142:237–241, 1990. 1.5.4

[151] M. Oliphant and J. Batali. Learning and the emergence of coordinated communication. *Center for Research in Language Newsletter*, 11(1), 1997. 1.5.8

[152] G. M. B. Oliveira et al. Evolving solutions of the density classification task in 1d cellular automata, guided by parameters that estimate their dynamic behaviour. In M. A. Bedau et al., editors, *Artificial Life VII*, pages 428–436. MIT Press, 2000. 2.2.2, 2.7, 7.1, 7.3.1

[153] B. Olsson. *NK*-landscapes as test functions for evaluation of host-parasite algorithms. In Schoenauer et al. [184], pages 487–496. 3.4.2, 3.8, 7.2.1

[154] L. E. Ortiz and M. Kearns. Nash propagation for loopy graphical games. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 793–800. MIT Press, 2003. 1.5.3

[155] L. Pagie and P. Hogeweg. Evolutionary consequences of coevolving targets. *Evolutionary Computation*, 5(4):401–418, 1997. 3.2.2

[156] L. Pagie and P. Hogeweg. Information integration and red queen dynamics in coevolutionary optimization. In Zalzala et al. [209], pages 1260–1267. 1.4.1, 3.8

[157] L. Panait and S. Luke. A comparative study of two competitive fitness functions. In W. B. Langdon et al., editors, *Proceedings of the 2002 Genetic and Evolutionary Computation Conference (GECCO)*, pages 503–511. Morgan Kaufmann, 2002. 1.4.1

[158] J.-S. Pang, R. E. Stone, and R. W. Cottle. *The Linear Complementarity Problem*. Computer Science and Scientific Computing. Academic Press, 1992. 1.5.2

[159] J. Paredis. Coevolutionary computation. *Artificial Life*, 2(4):355–375, 1995. 3.6.2

[160] J. Paredis. Coevolutionary algorithms. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C7.4. Institute of Physics Publishing, Bristol, UK, 1997. 3.1

[161] J. Paredis. Coevolving cellular automata: Be aware of the red queen! In T. Bäck, editor, *Proceedings of the 7th Int. Conference on Genetic Algorithms (ICGA 97)*, pages 393–400. Morgan Kaufmann, 1997. 3.2.1, 3.5.1

[162] J. Paredis. Towards balanced coevolution. In Schoenauer et al. [184], pages 497–506. 3.4.2, 3.8, 7.2.1

[163] H.-O. Peitgen, H. Jürgens, and D. Saupe. *Chaos and Fractals: New Frontiers of Science*. Springer-Verlag, 1992. 6.5

[164] J. B. Pollack and A. D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(3):225–240, 1998. 3.2.2, 3.6.2, 7.1

[165] M. A. Potter. *The Design and Analysis of a Computational Model of Cooperative Coevolution*. PhD thesis, George Mason University Department of Computer Science, 1997. 1.4.2

[166] M. A. Potter and K. A. D. Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor and H.-P. Schwefel, editors, *Proceedings of the Third Conference on Parallel Problems Solving from Nature (PPSN 3)*, pages 249–257. Springer-Verlag, 1994. 3.6.2

[167] M. A. Potter and K. A. D. Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000. 1.4.2, 1.4.6

[168] T. S. Ray. Evolution, complexity, entropy and artificial reality. *Physica D*, 75(1–3):239–263, 1994. 1.5.8, 9.7

[169] T. S. Ray. An evolutionary approach to synthetic biology: Zen and the art of creating life. *Artificial Life*, 1(1/2):195–226, 1994. 1.5.8

[170] C. W. Reynolds. Competition, coevolution and the game of tag. In Brooks and Maes [27], pages 59–69. 3.3

[171] J. Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54:296–301, 1951. 1.5.5

[172] C. D. Rosin. *Coevolutionary Search Among Adversaries*. PhD thesis, University of California, San Diego, 1997. 1.5.7, 3.2.1, 3.3, 3.4.1, 3.4.2, 3.5.2, 3.6.1, 3.6.2, 5.1, 5.3, 5.3.1, 5.5, 5.4, 7.2.1, 7.2.2, 7.2.3, 7.5.3

[173] C. D. Rosin and R. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In Eshelman [56], pages 373–381. 3.5.2

[174] C. D. Rosin and R. Belew. A competitive approach to game learning. In Blum and Kearns [23], pages 292–302. 1.4.6, 1.5.7, 2.10.7

[175] C. D. Rosin and R. Belew. New methods for competitive co-evolution. *Evolutionary Computation*, 5(1):1–29, 1997. 3.3, 3.6.1, 3.6.2, 3.8, 5.5, 8.1, 8.8

[176] G. W. Rowe, I. F. Harvey, and S. F. Hubbard. The essential properties of evolutionary stability. *Journal of Theoretical Biology*, 115:269–285, 1985. 1.5.4, 4.2, 4.3.1

[177] M. J. K. Sally A. Goldman. On the complexity of teaching. *Journal of Computer and Systems Sciences*, 50(1):20–31, 1995. 1.5.7, 2.10.7, 3.5.2

[178] A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):211–229, 1959. 3.2.2, 3.2.4

[179] A. Samuel. Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal of Research and Development*, 11(6):601–617, 1967. 3.2.4

[180] L. Samuelson. *Evolutionary Games and Equilibrium Selection*. MIT Press, 1997. 1.5.1, 1.5.4, 2.6.5

[181] G. M. Saunders and J. B. Pollack. The evolution of communication schemes over continuous channels. In Maes et al. [132], pages 580–589. 1.5.8

[182] L. M. Schmitt. Coevolutionary convergence to a global optima. In Cantú-Paz et al. [32], pages 373–374. 1.4.6

[183] L. M. Schmitt. Theory of coevolutionary genetic algorithms. In M. Guo and L. T. Yang, editors, *International Symposium on Parallel and Distributed Processing and Applications (ISPA)*, volume 2745 of *Lecture Notes in Computer Science*, pages 285–293. Springer, 2003. 1.4.6

[184] M. Schoenauer et al., editors. *Parallel Problem Solving from Nature VI*. Springer-Verlag, 2000. 62, 153, 162, 201

[185] K. Sims. Evolving 3d morphology and behavior by competition. In Brooks and Maes [27], pages 28–39. 1.1, 1.4.1, 3.2.3, 3.6.2, 9.7

[186] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 15–22. ACM, 1994. 2.4.2

[187] K. O. Stanley and R. Miikkulainen. The dominance tournament method of monitoring progress in coevolution. In A. Barry, editor, *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 242–248, 2002. 3.3, 3.6.2, 8.1

[188] S. H. Strogatz. *Nonlinear Dynamics and Chaos*. Addison-Wesley, 1994. 4.2, 6.5

[189] T. Taylor. Creativity in evolution: Individuals, interactions and environments. In P. J. Bentley and D. W. Corne, editors, *Creative Evolutionary Systems*, pages 79–108. Morgan Kaufmann, 2001. 1.5.8, 9.7

[190] T. J. Taylor. *From Artificial Evolution to Artificial Life*. PhD thesis, Division of Informatics, University of Edinburgh, 1999. 1.5.8

[191] G. Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68, 1995. 3.2.2

[192] P. R. Thie. *An Introduction to Linear Programming and Game Theory*. John Wiley and Sons, 1988. 1.5.2, 8.5.3, 8.5.4

[193] P. Todd and G. Miller. On the sympatric origin of species: Mercurial mating in the Quicksilver model. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 547–554. Morgan Kaufmann, 1991. 1.5.8, 3.8

[194] P. Todd and G. Miller. Biodiversity through sexual selection. In C. G. Langton and K. Shimohara, editors, *Artificial Life V (1996)*, pages 289–299. MIT Press, 1997. 1.5.8

[195] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In H. Kautz and B. Porter, editors, *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, 2000. 1.5.6

[196] L. Valiant. A theory of the learnable. *Communications of the A.C.M.*, 27(11):1134–1142, 1984. 1.5.7

[197] L. van Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, 1973. 3.3, 3.5, 9.3.3

[198] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic, second edition, 2001. 1.5.2

[199] X. Wang and T. Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1571–1578. MIT Press, 2003. 1.5.6

[200] R. Watson and J. Pollack. Hierarchically-consistent test problems for genetic algorithms. In P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the 1999 Congress on Evolutionary Computation (CEC)*, pages 1406–1413. IEEE Press, 1999. 1.4.2

[201] R. A. Watson and J. B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In Schoenauer et al. [184], pages 425–434. 7.1

[202] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector et al., editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 702–709, 2001. 3.4.1, 3.6, 3.6.1, 8.1, 8.6, 8.6.1, 9.7, 10

[203] J. Weibull. *Evolutionary Game Theory*. MIT Press, 1995. 1.5.4, 3.5

[204] J. Werfel, M. Mitchell, and J. P. Crutchfield. Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, 4(4):388–393, 2000. 3.2.1, 7.1, 7.3.1

[205] G. M. Werner and M. G. Dyer. Evolution of communication in artificial organisms. In Langton et al. [120], pages 659–687. 1.5.8

[206] R. P. Wiegand, W. C. Liles, and K. A. De Jong. Analyzing cooperative coevolution with evolutionary game theory. In Fogel et al. [72], pages 1600–1605. 1.4.2

[207] R. P. Wiegand, W. C. Liles, and K. A. De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In De Jong et al. [49], pages 203–220. 1.4.2

[208] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001. 1.5.6

[209] A. Zalzala et al., editors. *Proceedings of the 2000 Congress on Evolutionary Computation*. IEEE Press, 2000. 35, 58, 156