Abstract

Title of dissertation: Causally-Guided Evolutionary Computation for Design

Timur Chabuk, Doctor of Philosophy, 2013

Dissertation directed by: Professor James A. Reggia
Department of Computer Science

During recent years, evolutionary computation methods have been used successfully to discover solutions to problems involving design and invention in a wide variety of fields. However, for the evolutionary process to remain computationally tractable when applied to increasingly complex design problems, new extensions must be developed that increase the efficiency and effectiveness with which evolutionary systems produce optimal designs. To this end, the goal of the research presented here is to develop one such potential extension: causally-guided evolution. By this I mean evolutionary systems where the application of genetic operators to an individual are driven in part by observing that individual's performance characteristics and applying these operators based on explicit cause-effect relations in the domain. This differs from past evolutionary methods in which, after fitness-based selection, ge-

netic operators are applied to individuals blindly and randomly (i.e., without respect to the performance characteristics of the individuals).

In this context, this dissertation makes a number of significant contributions. A framework for causally-guided evolution is defined, including causally-guided genetic operators based on causal knowledge that is supplied by domain experts. The ability of these methods and causally-guided mutation to produce better solutions than conventional evolutionary processes is demonstrated on a neural network optimization task. These methods are then extended to include crossover, and the synergistic effects of causally-guided crossover and mutation are demonstrated when applied to a real-world antenna design task. Causally-guided mutation is extended further to influence both where and how mutation occurs, and the effectiveness of this approach is shown when applied to a constructive design task that creates synthetic social networks. Finally, a causally-guided evolutionary system that acquires causal knowledge through observation of the evolutionary process, rather than being given the knowledge a priori, is developed and successfully applied, demonstrating the applicability of causally-guided evolution to problems in which causal knowledge is not available. Collectively, this work clearly demonstrates for the first time the promise of causally-guided evolutionary computation in a variety of forms and when applied to a range of application problems.

Causally-Guided Evolutionary Computation for Design

by
Timur Chabuk

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:

Professor James A. Reggia	Committee Chair
Professor Michelle Girvan	Dean's Representative
Professor William Rand	Committee Member
Professor Dana Nau	Committee Member
Professor Atif Memon	Committee Member

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

During recent years, evolutionary computation methods have been used successfully to discover solutions to problems involving design and invention in a wide variety of fields. Examples include the evolutionary design of electronic circuits (Koza, 2003; Koza et al., 1997), antennas (Altshuler and Linden, 1997; Lohn et al., 2004, 2008), neural network architectures (Gruau and Quatramaran, 2001; Jung and Reggia, 2006; Stanley and Miikkulainen, 2002; Chen et al., 2012), music compositions (Biles, 2002), artistic endeavors (Rocke, 2002), control mechanisms for robots (Michel, 2001; Dupuis et al., 2013) and for cellular automata (Pan and Reggia, 2010), mechanical systems (Hu et al., 2008; Lipson, 2008; Rubrecht et al., 2011), architectural structures (Rosenman, 1997; Byrne et al., 2011), and quantum circuitry and algorithms (Spector and Klein, 2008; Stadelhofer et al., 2008). The design process in these situations is a human-machine collaboration in which a person defines the problem, search space, fitness function, etc., while the evolutionary process generates and evaluates a much larger number of alternatives than could be done manually. Sometimes the results of evolutionary systems are even qualitatively different from previous human-only solutions, such as unexpected animal-like forms or "biomorphs" (Dawkins, 1996), patentable elec-

tronic circuits (Koza, 2003), and novel irregularly shaped antennas (Hornby et al., 2006).

Evolutionary computation thus appears to be a very promising tool for supporting the creative design process. However, in order for the evolutionary process to remain computationally tractable when applied to increasingly complex design problems, new extensions must be developed that increase the efficiency and effectiveness with which evolutionary systems produce optimal designs.

To this end, the goal of the research presented here is to develop one such potential extension: "causally-guided evolution." By this I mean evolutionary systems where the applications of genetic operators to an individual in an evolving population are driven in part by observing that individual's performance characteristics and performing causal reasoning about those characteristics based on explicit cause-effect relations in the domain. This differs from past evolutionary methods in which genetic operators are applied to individuals blindly and randomly (i.e., without respect to the performance characteristics of the individuals). In my approach, causal knowledge/inference is used to bias (but not control) the application of mutation and crossover operators while leaving the fitness-based evolutionary search process otherwise unchanged. To make this idea clearer, it is useful to view the creation of each generation's population in a typical evolutionary process as consisting of two distinct steps:

1. selection of designs/individuals to carry forward into the next generation;

2. modification of designs/individuals via genetic operations.

There is a fundamental difference in the way evolutionary methods handle these two steps. The selection of individual designs to carry forward (aspect 1 above) is guided by the evaluated fitness of those designs. In contrast, the modification of designs via genetic operations such as crossover and mutation (aspect 2 above) is not influenced by any individual design's performance. In other words, the modification step where new problem solutions are generated using genetic operations is traditionally largely blind and random.

In the work presented in this dissertation, I introduce the use of explicit cause-effect relations in the exploration/variation aspect of evolutionary computation (aspect 2 above). While there are undoubtedly numerous cause-effect relations at play in an evolutionary process, in this study I explore the use of two such relations, which I term diagnostic causal relations and mechanistic causal relations. *Diagnostic causal relations* describe a correspondence between or a rule about the relationship between some part of the individual's genetic representation and some part of the same indvidual's phenotypic performance. *Mechanistic causal relations* describe the expected effect of modifying individuals in particular ways, and are described in greater

detail in Chapter 5. Each explicit causal relation is specified prior to the beginning of an evolutionary process, but is used during the evolutionary run to influence the application of a selected genetic/variation operator (mutation, crossover, etc.) to an individual that is a parent for the next generation. For example, in applying a mutation operator to an individual parent to produce a modified offspring, analysis of the parent using diagnostic causal relations is performed in order to bias the mutation so that the phenotypic problems of the individual are addressed with higher probability than they would be were the usual blind operator used.

1.1 Goals and Specific Aims

The central **goal** of this research is to develop and evaluate methods for causally-guided evolutionary computation, as described above. My **hypotheses** are that ultimately this will make evolutionary systems more effective by allowing them to explore a much larger number of good designs while still exploring novel solutions that initially appear unpromising, and more computationally efficient by decreasing the number of poorly fit individuals that do not contribute useful information to the evolutionary search process. I hypothesize that the benefits of causally-guided evolutionary systems will be most pronounced when applied to design problems in which domain expertise is present but insufficient for solving problems in closed form. Causally-guided evolutionary methods are intended to preserve the

limited dependence on domain knowledge found in traditional evolutionary computation, while leveraging whatever cause-effect knowledge is available. There is no a priori guarantee that any of these benefits will hold, and in fact it is entirely plausible that just the opposite would be true, i.e., adding causal influences could produce evolutionary search that is less effective and less computationally efficient. The purpose of the research described in this dissertation is to examine this issue.

In this context, the following are the **specific objectives** that guided this research:

1. Design and evaluate a framework for using diagnostic causal relations to guide the evolutionary search process through the biasing of *where* mutation operators are applied to individuals. Explore the feasibility of these methods by evaluating their performance when used to solve parameter optimization problems in which causal knowledge is available and well-understood. The intent of this first objective was to provide an initial proof-of-concept.

2. Extend the methods developed under the previous objective by designing and implementing a second genetic operator that is guided by diagnostic causal knowledge: causally-guided crossover. As in causally-guided mutation, diagnostic relations are used to bias where crossover is applied to individuals, in the sense that it influences what parts of individuals are inherited by offspring. Explore the feasibility of these

methods by evaluating their performance when used to solve parameter optimization problems in which diagnostic casual knowledge is present but incomplete or unclear. In particular, examine the combined effects of causally-guided mutation and causally-guided crossover to evaluate whether they can be applied synergistically.

3. Develop above framework of objectives 1 and 2 for using causal relations to guide the evolutionary search process when applied to constructive design problems. In particular, extend the framework to include the use of mechanistic causal relations to guide how mutation is applied to indivduals. Explore the feasibility of these methods by evaluating their performance when used to solve constructive design problems (in contrast to the parameter optimization problems considered as part of the first two objectives). In particular, examine the combined effects of the two types of causal guidance (*where* guidance based on diagnostic knowledge and *how* guidance based on mechanistic knowledge) to evaluate whether they act antagonistically or synergistically.

4. Design and evaluate a framework for applying causally-guided evolution to application problems in which causal knowledge is not available a priori and must instead be acquired and applied during the execution of the evolutionary process. Explore the feasibility of these methods by evaluating their performance when used to solve design problems without the use of a priori causal knowledge.

1.2 Executive Summary

The rest of this dissertation is organized as follows. Chapter 2 discusses previous work that is directly relevant to the research presented in this dissertation, including research in the fields of knowledge incorporation in evolutionary computation, adaptive and self-adaptive parameter control in evolutionary computation, evolutionary computation for design, and causal reasoning.

Chapter 3 presents an initial exploration of causally-guided evolutionary computation using a single causally-guided genetic operator. A general framework for causally-guided evolution is presented, including the formalism by which domain experts may describe diagnostic cause-effect relationships in their domain, the high-level form of causally-guided genetic operators, and the general (i.e., application independent) form of one such operator: causally-guided mutation. To evaluate these ideas, a causally-guided evolutionary system was developed and applied to the task of optimizing a set of connection weights in a fixed-architecture neural network in order to produce a network that recognizes mirror symmetry of input patterns. As discussed in Chapter 3, this task was selected for an initial study in part because there is clear and known cause-effect relationships in the domain and because historically it has been a standard difficult test problem in neural network research. This diagnostic cause-effect knowledge, which serves as the basis for causal guidance in this study, is described in detail along with the application-

specific causally-guided mutation operator that is used. Evolutionary systems with and without causal guidance were developed and used to solve the neural network design problem. Analysis of these experimental results is presented, revealing a clear increase in the effectiveness and efficiency of causally-guided evolutionary computation when compared to control methods, and establishing for the first time the feasibility of causally-guided evolution.

Chapter 4 presents a number of important extensions to the work described in Chapter 3, both in terms of the causally-guided evolutionary methods themselves as well as the types of application problems to which causally-guided evolution is applied and evaluated. While Chapter 3 explored the use of a single causally-guided operator (mutation), the work presented in Chapter 4 introduces a second (crossover), the general (i.e., application independent) form of which is presented. These methods are evaluated by applying them to the real-world task of designing antenna arrays that meet prespecified performance criteria. In contrast to the neural network optimization task from the previous chapter, the causal knowledge that is available for this antenna design task is more limited and less complete, and a central goal of this chapter is to evaluate the feasibility of causally-guided evolutionary computation when applied to such real-world problems. The specific diagnostic causal knowledge, application-specific causally-guided mutation, and application-specific causally-guided crossover operators that are used in this study are presented, followed by an experimental evaluation in which

evolutionary systems with and without causal guidance are used to solve the antenna design problem. Analysis of these experiments is presented, revealing distinct benefits of causally-guided mutation and crossover, as well as the synergistic effects of applying both causally-guided operators together. Further analysis examines the types of antenna designs that are produced, revealing that causally-guided evolution avoids producing particular types of antenna designs that are directly related to the causal knowledge that is used. Finally, antenna designs were systematically varied and their changes in performance examined in order to learn about cause-effect relations in this domain and more generally.

Chapter 5 explores the feasibility of applying causally-guided evolutionary computation methods to design construction problems, rather than design optimization problems as in Chapters 3 and 4. To this end, a second type of causal relation is introduced. Mechanistic causal relations describe the cause-effect relationship between the application of mutation operators to design components in an individual, and the resulting change in performance of those individuals. A new causally-guided mutation operator is defined in which "where" guidance based on diagnostic relations and "how" guidance based on mechanistic relations are both used. These methods are evaluated by applying them to the task of designing synthetic social networks with characteristics that match real-world data sets. The specific causal knowledge that is employed, as well as the application-specific forms of causally-guiding where and how mutation are applied are described. Evo-

lutionary systems that employ and those that do not employ these forms of causal guidance are used to design synthetic social networks. Analysis of their performance is presented, revealing a dramatic increase in performance when causal guidance is used to influence both *where* and *how* mutation is applied, and suggesting some important lessons regarding the proper way to design causally-guided genetic operators.

Chapter 6 explores the feasibility of applying causally-guided evolutionary computation methods to application domains in which causal knowledge is not available a priori, and instead must be acquired and applied during the evolutionary process. The evolutionary process presents a wealth of data from which causal relations may be inferred. Chapter 6 presents one such way in which mechanistic causal relations may be acquired through observation of the evolutionary process. These methods are evaluated by applying them to the same synthetic social network design task as in Chapter 5. However, in this chapter the evolutionary methods are applied without any a prior causal knowledge. Analysis of these experiments is presented, revealing that the learned causally-guided evolutionary systems clearly outperform the control systems which do not employ any causal guidance. Furthermore, in many instances the evolutionary systems in which causal knowledge was acquired through observing the evolutionary process performed just as well as those for which causal knowledge was supplied a priori. This clearly supports the feasibility of applying causally-guided evolutionary computation to application domains in which causal knowledge is not available a prior,

and suggests that causally-guided evolution may be useful in a wide range of application domains. Furthermore, because these methods reduce the "start-up" costs associated with causally-guided evolution (i.e., defining causal relations, causal guidance, etc.,) they may be attractive even in application domains where a priori knowledge is available.

Chapter 7 presents conclusions, limitations of the work presented here, a summary of contributions made, as well as various directions for future research.

Chapter 2

Background

2.1 Introduction

In this section, existing studies that are relevant to this dissertation research into causally-guided evolutionary computation for design are presented. First, a general outline of evolutionary computation algorithms is presented, followed by a description of the canonical forms of a few major variants. While causally-guided evolutionary computation has not been previously studied, some research exists that explores the dynamic guidance of evolutionary processes through other means. These methods, commonly referred to as *adaptive* and *self-adaptive parameter control*, are presented and differences from the current research are discussed. The methods for causally-guided evolutionary computation presented here can also be seen as an instance of knowledge-incorporation. While no previous studies have explored the use of causal knowledge to bias genetic operators, past research exploring various other means of leveraging domain knowledge in evolutionary computation are discussed. There are two major types of design problems which evolutionary computation can be used to solve: design optimization and design construction. The research presented in this dissertation investigates the

feasibility of using causally-guided evolutionary computation to solve both types of problems. The differences between these types of problems is discussed, and previous studies involving the use of evolutionary computation methods to solve these problems are presented. Lastly, an overview of the use of causal-based diagnostic problem solving is provided. While these techniques are not used directly used in this dissertation research, they help to shape the cause-effect formalism and the causal guidance that is used and are particularly relevant to future direction of the research presented here.

2.2 Evolutionary Computation

The term *evolutionary computation* refers to a set of general-purpose search algorithms that are inspired by principles of biological evolution. While there are a substantial number of variations between different approaches to evolutionary computation, such as genetic algorithms (Holland, 1975; Mitchell, 1996; Goldberg, 1989; De Jong, 2006) evolution strategies (Rechenberg, 1973; Schwefel, 1981, 1995; Rudolph, 2000), genetic programming (Banzhaf et al., 1998; Koza, 1992), and evolutionary programming (Fogel et al., 1966; Fogel, 1991, 1995; Porto, 2000), most methods work at a top level as illustrated in Figure 2.1. These algorithms function by creating successive populations, or generations, of candidate problem solutions. In each generation, the fitness of each individual in the population is evaluated. Successive generations are created by selecting the more fit individuals from

the current generation, and applying genetic operations such as crossover and mutation to them. The fitness of each individual in this new generation is calculated, and selection methods and genetic operations are applied again, creating yet another generation. This continues until some predetermined termination criterion is met. While genetic algorithms, evolutionary strategies, evolutionary programming and genetic programming follow this generic outline, the canonical form of each of these algorithms uses distinct combinations of various encoding methods, selection methods, and genetic operations.

Genetic algorithms traditionally employ a fixed-length binary string to encode solutions to a problem. Stochastic fitness-proportionate selection is used, in which the probability that a solution will survive to the next generation is equal to its fitness divided by the sum of the fitness of all individuals. Mutation operators are used sparingly, and usually consist of simply flipping a random bit in the binary string. The crossover operator, in which the genetic materials from two individuals are recombined to create two new individuals, is usually the dominant genetic operator in genetic algorithms (Mitchell, 1996; Banzhaf et al., 1998; De Jong, 2006).

In contrast, *evolutionary strategies* employ real-valued vectors to represent prospective solutions. Each individual consists of a target vector, which encodes the actual solution being evolved, and a vector of strategy parameters, which defines how genetic operations are applied to the individ-

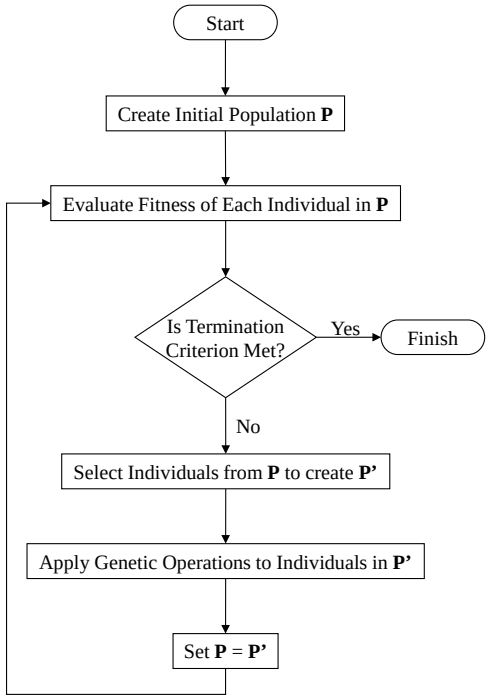

Figure 2.1: Schematic overview of a generic evolutionary computation algorithm.

ual. For example, during mutation small normally distributed random values are added or subtracted from the target vector elements. The variance of these normal distributions is defined by the strategy parameters. Strategy parameters and target vectors are handled separately during mutation and recombination. While there has been some work involving recombination in evolutionary strategies, the primary genetic operator is mutation. As in all evolutionary computation methods, selection is fitness based, but is deterministic rather than stochastic (Rudolph, 2000; Banzhaf et al., 1998; De Jong, 2006).

The goal of *evolutionary programming* has historically been to design

small computer programs in the form of finite state machines. In evolutionary programming, the actual finite state machine consisting of nodes and connections is used to encode the solution. In this sense, in contrast to genetic algorithms and evolutionary strategies, there is no separation between the genotype and phenotype of an individual in evolutionary programming. Mutation operators such as adding a node, removing a node, etc., are used to make changes to an individual finite state machine. Deterministic fitness-based selection is used. Crossover operators traditionally are not used (Porto, 2000; Banzhaf et al., 1998; De Jong, 2006).

Genetic programming is used to evolve computer programs that solve problems, but otherwise is quite similar to genetic algorithms. Genetic programming most commonly uses a tree-based representation, in which each node represents either a function or terminal. The tree may be parsed and interpreted as a computer program. The dominant genetic operator in genetic programming is crossover, in which two sub-tress from two individuals are swapped, resulting in two novel programs. Mutation operators are also used in which a sub-tree of an individual is selected and replaced with a new randomly generated sub-tree. Genetic programming variants have used deterministic selection methods similar to those used by evolutionary strategies, and stochastic selection methods similar to those used by genetic algorithms (Banzhaf et al., 1998; De Jong, 2006).

While the canonical form of these evolutionary computation methods are presented above, it is worth noting that in recent years these fields have

borrowed extensively from each other, and the distinction between them continues to blur. For example, it is common for modern genetic algorithm approaches to employ real-valued chromosomes, and place more emphasis on mutation operators than is done in the canonical form of the algorithm.

Similarly, the evolutionary techniques described in this dissertation do not entirely match any of the canonical forms of evolutionary computation described above and instead borrow elements from each. In the symmetry neural network design and antenna design studies (presented in Chapters 3 and 4), real-valued vectors of numbers are used to represent prospective solutions, as is done in evolutionary strategies. In the synthetic social network design study (presented in Chapters 5 and 6), the genetic representation used is the social network itself, i.e., there is no separation between the genotype and the phenotype of the individual, as is done in evolutionary programming. In all of the studies presented in this dissertation, stochastic fitness-based selection is used, as is done in genetic algorithms and genetic programming. Most importantly, in each of these evolutionary computation methods the application of genetic operators to individual prospective solutions is carried out without regard to the fitness characteristics of those individuals. In contrast, the research presented here focuses on the use of causal relations to influence the application of genetic operators.

2.3 Knowledge Incorporation in Evolutionary Computation

There have been a significant number of previous studies in which application-specific domain knowledge has been used to influence how evolutionary search is conducted. Past work on *knowledge incorporation* has long recognized that implicitly/explicitly incorporating domain knowledge in an evolutionary process can be useful in choosing an effective representation for the individuals in a population, in creating a non-random initial population, in designing fitness functions, and in composing genetic/variation operators (De Jong, 1988; Jin, 2004; Du and Rada, 2012).

For example, if one represents solutions to a traveling salesperson problem as permutations of the cities to be visited, then conventional mutation and crossover operators will generally produce illegal offspring solutions. Using this knowledge in advance can lead one to use genetic operators that instead permute/invert some of the cities of a parent in creating offspring and thereby avoid generating illegal solutions (Fogel, 2000; Whitley, 2000). As another example, in evolving rule sets for a pattern classification task, one may use domain knowledge about rule structure to restrict where crossover points can be done in the second parent chromosome relative to their locations in the first to avoid illegal offspring (De Jong et al., 1993). Further, in attempting to evolve solutions to job shop scheduling problems, it is possible to use specialized mutation operators based on knowledge from previous

studies about the occurrence of idle capacity to increase the effectiveness of the evolutionary process (Becerra and Coello, 2005).

My approach relates more to past studies in *fitness approximation* (Jin, 2005), where an approximate model of the fitness function is used to efficiently estimate the fitness of individuals. In some of this past work, the fitness approximations of individuals have been used to guide genetic operators (Abboud and Schoenauer, 2002; Rasheed et al., 2005). However, in causally-guided evolution there is no fitness approximation (the actual fitness is calculated) and instead the causal knowledge is used to guide genetic operators.

Finally, numerous past studies have investigated the combination of local search with evolutionary computation methods. Inspired in part by natural systems that combine evolutionary adaptation of a population of individuals with learning within the lifetime of each of its members, theses algorithms are sometimes referred to as hybrid evolutionary algorithms, Baldwinian evolutionary algorithms, Lamarckian evolutionary algorithms, cultural algorithms, genetic local search, or memetic algorithms (Whitley et al., 1994; De Jong, 2006; Mitchell, 1996; Krasnogor and Smith, 2005; Knowles and Corne, 2005; Moscato and Cotta, 2010; Neri and Moscato, 2011). In all of these approaches, local search is applied to each offspring that is produced by standard genetic operators (Knowles and Corne, 2005). For example, Lamarckian evolution (genetic transmission of traits acquired during the life

of an individual to its offspring (Whitley et al., 1994)), although widely viewed as biologically implausible, has been used with mixed results in past EC applications (summarized in (De Jong, 2006; Mitchell, 1996)). In this way, the global search capabilities of evolutionary computation are combined with the fine-tuned optimization capabilities of local search (Moscato and Cotta, 2010). These algorithms have been successfully applied to a variety of problems, including flow-shop scheduling (Ishibuchi et al., 2002), drug design (Tse et al., 2007), telecommunication routing (He and Mort, 2000), and the design of control systems for simulated agents (Jung and Reggia, 2009).

As with memetic algorithms, causally-guided evolution involves augmenting conventional evolutionary search with additional capabilities; in this case: causal reasoning. However, it should be noted that the changes made to individuals by causally-guided genetic operators can be quite large, and therefore cannot accurately be described as being "local." Furthermore, in memetic algorithms the local search occurs during the fitness evaluation ("lifetime") of the individual whereas in causally-guided evolution, causal guidance is applied during reproduction.

2.4 Adaptive and Self-Adaptive Evolutionary Computation

In most evolutionary computation variants, there are numerous parameters that govern the execution of the evolutionary process. The values of parameters such as population size, mutation rate, crossover rate, etc., have

a strong influence on the overall performance of an evolutionary process. However, determining a good or optimal set of parameter values a priori can be very difficult, if not impossible. Additionally, parameter settings that are optimal early in an evolutionary process may not be optimal later in the process (Meyer-Nieberg and Beyer, 2007). The fields of *adaptive parameter control* and *self-adaptive parameter control* attempt to address this problem by dynamically adjusting parameter settings during an evolutionary run in response to the performance of the evolutionary process (Leung et al., 2012; Aleti et al., 2012).

There are numerous criteria by which to classify the various parameter control methods. Angeline (Angeline, 1995) proposes classifying parameter control methods according to two criteria: the types of rules that govern the adaptation of parameters and the level at which the adaptation occurs. There are two types of rules that may be used: absolute update rules and empirical update rules. Absolute update rules, more commonly known as *adaptive parameter control* methods, compute statistics about an evolutionary process as it is running and use predetermined heuristics to adjust parameter values in response to the observed statistics. In this sense parameter changes are made dynamically, but according to fixed rules. In contrast, empirical update rules, more commonly known as *self-adaptive parameter control* methods, encode parameter values into the individual chromosomes in a population and allow the evolutionary process to change them through standard evolutionary methods. The parameter values encoded into a chromosome

play no roll in the fitness of that individual, and thus cannot be directly optimized by the evolutionary process. However, if a parameter value encoded in a particular individual results in beneficial modifications being made to that individual's offspring, it will spread through the population along with the offspring. Detrimental parameter values will result in damaged offspring which are less likely to survive, and thus the detrimental parameter values themselves are less likely to be persist in the population. In this manner, the evolutionary process optimizes the encoded parameter values as well as the actual individual solutions.

The second criteria by which Angeline proposes parameter control methods should be classified is the level at which parameter adaptation occurs. *Population level* adaptation results in global changes that affect the entire population of individuals. *Individual level* adaptation modifies parameters that are associated with particular individuals, independent from other individuals. *Component level* adaptation changes parameters that are associated with individual components of an individual, independent from other components in that individual. The various levels at which parameter control may take place involve important trade-offs. Population level methods are easier to implement than individual or component level methods, as they involve fewer parameters and are usually conceptually simpler. However, population level methods do not have as much potential as individual and component level methods to efficiently control the evolutionary process. This is because the types of parameter adjustments that would be ideal for each individual

will most likely not be the same as those that are ideal for the population on average. Put differently, lower level methods offer greater control in that parameters associated with each individual (or each component) may be independently adjusted. However, the large number of parameters involved in such methods makes it difficult to adjust them efficiently. For adaptive parameter control methods, it is difficult to develop heuristics that efficiently and effectively adjust this large set of parameters (Angeline, 1995).

In addition to these two criteria, Eiben (Eiben et al., 1999) proposes using what evolutionary characteristic is being adapted, and the information about the evolutionary search that is used to guide adaptation, as two additional classification criteria. In this classification scheme, the primary criteria for classification are whether the method is adaptive or self-adaptive (as discussed above) and what evolutionary characteristic is being adapted. The types of evolutionary characteristics that may be adapted include: representation of individuals, evaluation functions, variation operators (crossover, mutation, etc.,), selection operators, replacement operators, and population. The secondary criteria for classification are the level/scope at which adaptation occurs (as discussed above) and the information that is used to guide the adaptation.

Numerous adaptive and self-adaptive parameter control methods have been developed for evolutionary strategies, evolutionary programming, genetic algorithms, and genetic programming (surveyed in (Angeline, 1995;

Eiben et al., 1999; Meyer-Nieberg and Beyer, 2007)). While adaptive and self-adaptive methods are common to contemporary evolutionary strategies and evolutionary programming methods, they are more rarely used in genetic algorithms and genetic programming (Meyer-Nieberg and Beyer, 2007). Adaptive and self-adaptive methods have been used to adjust representation interpretations (Shaefer, 1987; Schraudolph and Belew, 1992; Whitley et al., 1991), mutation operators and their probabilities (Back, 1992; Julstrom, 1995; Lis, 1996; Smith and Fogarty, 1996), crossover operators and their probabilities (Spears, 1995; Angeline, 1996; Julstrom, 1995; Lis, 1996), and evaluation functions (Eiben and Ruttkay, 1996; Smith and Tate, 1993). Self-adaptive algorithms have been applied on a component (Angeline and Pollack, 1993; Angeline, 1996), individual (Schaffer and Morishima, 1987; Spears, 1995), and population level (Spears, 1995; Teller, 1996). In contrast, the vast majority of adaptive algorithms have been applied on a population level (Rechenberg, 1973; Shaefer, 1987; Schraudolph and Belew, 1992; Whitley et al., 1991; Smith and Tate, 1993; Julstrom, 1995; Lis, 1996; Eiben and Ruttkay, 1996), and very little (Iba and de Garis, 1996; White and Oppacher, 1994) has been done on an individual or component level. In each of these studies, it was found that evolutionary computation systems employing adaptive and/or self-adaptive parameter control outperformed comparable evolutionary computation systems employing static parameters.

The research presented in this dissertation can be viewed as an instance of adaptive and self-adaptive parameter control, in which causal relations are

used as the basis for adapting parameters associated with genetic operators on an individual level. For example, in a typical evolutionary approach, each gene in an individual can be viewed as having a parameter associated with it that specifies the likelihood that the gene will be modified during mutation. In many instances, this parameter may not be explicitly specified but is still implicitly defined (e.g., equal probability is assumed). In the research presented here, causal relations are used as the basis for examining the performance characteristics of the individual in question and adapting these parameter values. Interestingly, there is very little past work (Iba and de Garis, 1996; White and Oppacher, 1994) involving the use of adaptive parameter control methods on an individual or component level, as the research presented in this dissertation does. To my knowledge, no previous studies use explicit cause-effect relations as the basis for parameter adaptation.

2.5 Evolutionary Design

Evolutionary computation methods may be used to solve two distinct types of design problems: *design optimization* and *design construction*. Typically, in optimization problems, a great deal is known about the particular form that a solution will take, and the design process consists of optimizing a set of parameters associated with that form. It is common to represent solutions to optimization problems as a fixed-length list of parameter values. Such a representation is said to be knowledge-rich, as the representation does

not describe the actual form of a solution. Instead, domain knowledge allows the form of the solution to be assumed by the representation, and only parameters of that solution must be evolved (Bentley and Corne, 2002).

In contrast, constructive design problems are more open-ended, in that the structures of solutions are not known a priori, and to some extent must themselves be designed by the evolutionary process. Typically, in constructive design problems, rather than optimizing a fixed set of parameters, the evolutionary process must arrange and re-arrange design components to create new solutions. For this reason, component-based representations are typically used. These types of representations are knowledge-lean in that they do not rely on assumptions about the structure of solutions, and instead describe the structure of a solution themselves. For example, Lego-block based representations have been used to evolve build-able structures such as cranes and bridges (Funes and Pollack, 1999). Multi-layer neural networks have been evolved by arranging and re-arranging layers of neurons together to form complete network structures (Chen et al., 2012). Component-based representations have been used to evolve open-ended designs such as crooked-wire monopole antennas, in which each component can be viewed as a length of wire, and these components are arranged and re-arranged together to form tree-like structures (Hornby et al., 2006; Lohn et al., 2004). In each of these instances, the evolutionary process is best viewed as "exploring" the various ways in which components may be arranged together, rather than "optimizing" parameter values (Bentley and Corne, 2002; Bentley, 1999). Evolution-

ary computation methods may be applied to constructive design problems in the hopes of finding optimal designs or in the hopes of discovering novel design concepts that may not necessarily be optimal. For these reasons, evolutionary approaches to solving constructive design problems are sometimes referred to as creative evolutionary systems.

These differences in representation have important implications for the types of genetic operators that may be used when solving either design optimization or constructive design problems. In optimization tasks, it is common for mutation operators to randomly select locations in an individual's genotype and make random modification to the value of genes at those locations. These types of mutation operators change the value of parameters, but leave the overall structure of the genotype unchanged. In contrast, in constructive design problems, mutation operations often involve adding, removing or re-arranging design components, and in the process often change the form of an individual's genotype. For example, in the Lego-block design problem mentioned earlier, mutations may add, remove or adjust the arrangement between blocks in an individual design (Funes and Pollack, 1999). In Chapters 3 and 4 of this dissertation, methods for causally-guided evolution are applied to design optimization problems, while in Chapter 5 and 6 these methods are extended to address design construction problems. As discussed later in Chapter 5, these differences in genetic operators have important implications for the ways in which they may be causally-guided.

2.6 Causal-Based Diagnostic Problem Solving

Causality is a very difficult and controversial issue (Pearl, 2000). Historically, there has been much interest in causality from the AI community, where the importance of causal reasoning and explanation is widely recognized (Korb and Nicholson, 2004; Spirtes et al., 2000). This interest has driven considerable work in areas such as Bayesian belief networks and statistical relational networks (Charniak, 1991; Josephson, 1994; Pearl, 1988; Poole, 1998; Pearl, 2000). Techniques for causal reasoning have shown to be an effective tool in a number of domains, from geospatial reasoning (Couclelis, 2009; Shakarian and Subrahmanian, 2011) to human computer interaction Patokorpi (2009).

One such area is diagnostic problem solving, in which the task is to generate a hypothesis that best explains a set of observations (Peng and Reggia, 1990). This relates directly to the cause-effect reasoning that is typically employed by human designers in an iterative design process, in which the performance of prospective designs is examined and inferences are then made about the ways in which the design may be improved. As noted earlier, this cause-effect reasoning is lacking from current evolutionary computation methods, and the central goal of the research presented here is to explore the feasibility of incorporating causal relations into evolutionary computation methods. As such, diagnostic problem solving is clearly relevant to the research presented in this dissertation. However, it should be noted that in

this dissertation neither causal networks or rigorous diagnostic problem solving is actually used. Instead, past work in these areas informs the research conducted here in a more general sense (e.g., the form of diagnostic causal relations).

Domain knowledge in diagnostic problems includes a set of disorders and their prior probabilities, a set of manifestations, and the causal relationships between disorders and manifestations along with their causal strengths. The causal strength of a relationship between a disorder and a manifestation is the probability that the disorder will cause the manifestation, given that the disorder is present. This information can sometimes be represented by a bipartite graph, such as the one presented in Figure 2.2. In such a graph, each node represents either a disorder or a manifestation, and is labeled with the relevant prior probability. Each link in the graph represents a causal relationship between a disorder and a manifestation, and is labeled with the causal strength of that relationship. Such a graph can be thought of as a special case of Bayesian belief network where all links are causal and probability calculations are performed using Bayesian methods. The task is to generate a set of disorders, or explanation, that best explains the presence of the observed manifestations.

Given this causal knowledge, it is straightforward to evaluate the likelihood of any single explanation using classical methods such as Bayesian networks. However, the challenge of diagnostic problem solving lies in the

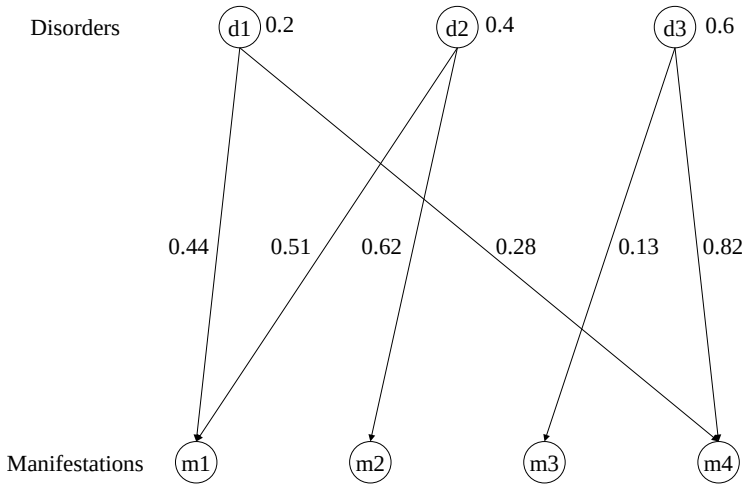

Figure 2.2: Example graph representation of causal knowledge for diagnostic problem solving. Disorders are represented by nodes at the top of the figure and are labeled with prior probabilities. Manifestations are represented by nodes at the bottom. Each link represents a causal relationship between a disorder and a manifestations, and is labeled with the causal strength of that relationship.

extraordinarily large number of explanations that are possible. For example, in a problem with D possible disorders, there are 2^D possible explanations. With this in mind, numerous algorithms have been designed to mitigate this exponential explosion of explanations by using logic to restrict the space of explanations that are explored.

Parsimonious covering theory is one approach that seeks to generate an explanation that satisfies two competing criteria: the explanation should cover all observed manifestations and should be parsimonious (Peng and Reggia, 1990). An explanation is a cover for a set of manifestations if each present manifestation may be explained by at least one disorder in the expla-

nation. Parsimony, in this context, may mean a number of things including being a minimal cover, an irredundant cover, or a relevant cover. Parsimonious covering theory allows for the efficient generation of explanations using only knowledge of the presences of causal relationships (Nau et al., 1983). By integrating these methods with Bayesian methods, using prior probability and causal strength data, one can evaluate competing explanations based on how probable they are instead of how parsimonious they are (Peng and Reggia, 1990), thereby providing relative likelihood scores for the various explanations.

In this dissertation, the full power of diagnostic and Bayesian methods as described above are not used. For example, situations in which there are multiple simultaneous disorders and many-to-many relationships between disorders and manifestations are not considered here. However, past work in these areas inform the research presented here in a more general sense (e.g., the form of diagnostic causal relations).

Chapter 3

Causally-Guided Mutation for Design Optimization

3.1 Introduction

The goal of this chapter is to provide an initial proof-of-concept study of the feasibility of causally-guided evolutionary computation. By "causally-guided evolutionary computation," I mean an evolutionary system where cause-effect relations are used as the basis to examine the performance characteristics of individuals in an evolving population and to bias (but not to control) the application of genetic operators to those individuals in order to address identified performance problems. To this end, a general framework for causally-guided evolutionary computation is designed and presented in detail, and the form of causal knowledge that is supplied by domain experts and used by causally-guided evolution is defined. The high-level form of causally-guided genetic operators in general and for mutation specifically are defined. To evaluate these ideas, the performance of causally-guided evolution is evaluated when applied to the task of designing neural networks that detect mirror symmetry of input patterns, a task that is of particular historical relevance in the field of neural network research and one in which causal knowledge is apparent. The performance of the causally-guided evolutionary

system is compared to a carefully matched control system that does not use causal guidance but is otherwise identical, demonstrating for the first time the feasibility of casually-guided evolution.

3.2 Framework for Causally-Guided Evolution

3.2.1 General Form of Diagnostic Causal Knowledge

Causally-guided evolutionary computation causal knowledge that is supplied by domain experts prior to the execution of the evolutionary process. While mechanistic causal knowledge is introduced in Chapter 5, diagnostic causal knowledge is used here. Each piece of diagnostic causal knowledge details the cause-effect relationship between a flawed aspect of the genotype and a problematic characteristic of the phenotype:

Genotypic Disorder $\rightarrow$ Phenotypic Problem

The arrow here is not logical implication, but causality. A *genotypic disorder* is simply a non-optimal aspect of the genotype; it is analogous to part of a diagnostic explanation. A *phenotypic symptom* is a performance problem that is caused by a genotypic disorder. It is analogous to a manifestation in diagnostic reasoning. For example, consider a genetic representation of house designs in which "number of windows" is a "gene". Suppose that an architect believes that too few windows may cause the interior of a house to be too dark:

Too Few Windows → Interior Too Dark

Here "too few windows" refers to a flaw in the genetic material that defines the size of windows, while "interior too dark" refers to a phenotypic problem of the house. This relationship between genotypic disorders and phenotypic performance problems mirrors the more general disorder-manifestation relationship that is seen in numerous domains, including the disease-symptom relationship in medicine, and the design flaw-performance problem relationship in manufacturing.

It should be noted that the term "non-optimal" does not refer to *any* genotypic aspect that does not *exactly* match the optimal value with arbitrary precision. Indeed, by such a strict definition almost all aspects of all evolved individuals would be non-optimal throughout most of the evolutionary process. Instead, in this research the terms "non-optimal" or "flawed" are used to describe genotypic aspects that deviate from the optimal values to a sufficiently large extent that the functionality of the design is impacted. Furthermore, it should be noted that individual genotypic aspects cannot truly be optimal or non-optimal individually, but instead must be optimized in concert with each other.

The list of causal relationships that a domain expert supplies does not need to be exhaustive. For most problems it is unlikely that all of the potentially relevant relationships between genotypic disorders and their resulting phenotypic problems will be known. Domain experts only need to list those

Generic Causally-Guided Genetic Operator
1) Assess Symptoms: Examine the individual's performance characteristics to identify its phenotypic symptoms.
2) Diagnose Disorders: Based on supplied causal relations and the individual's phenotypic symptoms, make inferences about the likelihoods of genotypic disorders in the individual, i.e., determine what parts of the individual's genome are likely to be flawed.
3) Prescribe Treatment: Bias the application of the genetic operator to the individual in order to address the diagnosed disorders.

Figure 3.1: High level overview of the three step process that is followed in using causally-guided genetic operators.

causal relationships that they believe to be most important. Even a single causal relation may be used in causally-guided evolutionary computation to influence the evolutionary search process.

3.2.2 General Form of Causally-Guided Genetic Operators

With causally-guided evolution, once it has been determined in the usual fashion that a genetic operator is to be applied to a specific individual, that individual's performance characteristics, along with causal knowledge and causal inference, are used to bias the execution of the operator, i.e., to bias manner in which the operator is applied to the individual. This is accomplished in three steps, as shown in Figure 3.1. The exact manner in which the genetic operations are biased in step 3 depends upon the specific type of the genetic operator in question.

3.2.3 General Form of Causally-Guided Mutation

When causally-guided genetic operations are applied to individuals, causal reasoning is used to bias the execution of the operators. First, the performance characteristics of the individuals in questions are examined, and any phenotypic problems that are present are identified. Next, causal reasoning uses the identified problems and human-provided causal knowledge to assess the relative likelihoods of each of the possible genotypic disorders (i.e., as potential explanations). Finally, this information is used to bias the execution of the genetic operators as described below.

Causally-guided mutation operations are biased such that those parts of the genotype with higher relative likelihoods of being flawed are made more likely to be mutated. Conversely, those parts of the genotype with lower relative likelihoods of being flawed become less likely to be mutated. In this work, causal guidance does not change which individuals are selected for reproduction or the number of modifications that will be made to an individual, only where the modifications are made.

It remains to be determined experimentally whether these biased genetic operators mislead the evolutionary process toward local minima, have no significant effect, or improve the quality of and speed with which solutions are produced. It is worth noting that causal guidance is used to **bias** genetic operators, but does not explicitly control them. Just as with fitness-guided selection, the use of causally-guided operations is probabilistic and does not

prevent the occurrence of poorly fit individuals that arise in the population due to random alterations; it simply aims to bias the process towards the formation of more fit individuals and decrease the formation of very poor individuals that often occur with standard evolutionary computation methods.

3.3 Symmetry Neural Network Design

As a first step to assess the effectiveness of these ideas, a causally-guided evolutionary system was developed and applied to the task of optimizing weights in a fixed-architecture neural network. The performance of this system was compared to a carefully matched evolutionary system that does not employ causal guidance but is otherwise identical. The goal of this experiment was simply to evaluate, for the first time, whether causal reasoning could be an effective means to guide evolution, thereby making it more efficient and effective.

3.3.1 Neuroevolution

Neural network design is traditionally a time and labor-intensive process in which human experts with extensive domain knowledge and experience manually design the networks. The expensive nature of this process has fueled recent interest in automated design methods such as evolutionary computation. *Neuroevolution*, the use of evolutionary computation methods to design neural networks, has been successfully used to design various aspects

of neural networks, including architectures, activation rules, learning rules, and connection strengths (Yao, 1999; Miikkulainen, 2010; Yang et al., 2010; Chen et al., 2012; Gomez, 2012). There are many reasons that neuroevolution is an appealing method for designing connection strengths in a fixed neural architecture. Gradient-descent algorithms are one common method for finding connection weights in supervised learning tasks. However, these algorithms are prone to getting stuck in local optima (Sutton, 1986), and require activation rules that are continuous and differentiable. In contrast, evolutionary computation methods have been shown to be good at finding globally optimal solutions, or good approximations to them, in a number of problem domains.

This task is an ideal one for an initial study of causally-guided evolutionary computation in part because of the availability of cause-effect knowledge in the domain that is already implicitly used. As explained in the sections below, conventional learning algorithms such as error back-propagation are based on some simple and well-understood cause-effect relations, which are adapted as the basis for causally-guided evolution in this study. While I hypothesize that causally-guided evolutionary computation will ultimately be most beneficial in application domains where causal knowledge is available but limited, it is important to conduct an initial evaluation of the idea in a domain for which well-understood and valid causal knowledge is readily available. This allows for an evaluation of causally-guided evolution while mitigating the risk that the causal knowledge being used is not correct.

3.3.2 Symmetry Networks

In this work, causally-guided evolutionary computation is used to design the connections strengths of a neural network with a fixed architecture for the task of detecting mirror symmetry of 1D input patterns. Examples of symmetric and asymmetric input patterns are illustrated in Figure 3.2. Symmetric and asymmetric input patterns are not linearly separable, and thus any neural network based on linear threshold neurons that would correctly classify the input patterns must have hidden neurons (Rumelhart, 1987). While the concept of symmetry is easily grasped by humans, the design of a neural network to detect symmetry is much more difficult, and is of historical significance in the field of neural networks (Mehrotra, 1997). Along with odd-parity classification and exclusive-or classification, symmetry classification was long seen as being a problem for which neural network learning was ill-suited (Minsky and Papert, 1969). Accordingly, the success of modern error back-propagation algorithms at solving these problems was instrumental in convincing people that modern EBP is an effective way to train neural networks (Rumelhart et al., 1986). Crucially for us here, it is generally very hard for an evolutionary process to discover an effective set of weights and biases to solve symmetry problems.

In this problem, connection weights are evolved for a structurally fixed neural network. The structure of the network is such that it has two hidden neurons, and is fully connected between the input neurons and the hidden

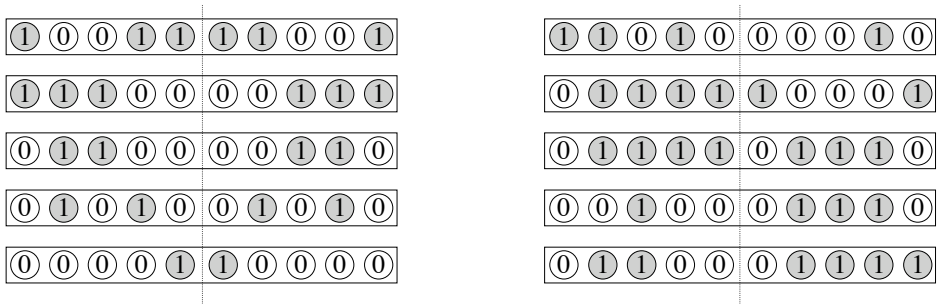

Figure 3.2: Example input patterns for 10-input 1D mirror symmetry problem. Each box of 10 numbers constitutes an example input pattern. The input patterns on the left half of the page are each symmetric, while those on the right are asymmetric.

neurons. Limiting the structure to two hidden neurons creates a narrow pathway through which the network may perform computation, and makes the design problem more challenging. The number of input neurons is dictated by the size of the input patterns that are being processed. For example, a network that will recognize mirror symmetry of input patterns consisting of 8 input bits will necessarily have 8 input neurons. The two hidden neurons are fully connected to a single output neuron. Each non-input neuron n is a linear threshold unit with activation dynamics as follows, where $\vec{a}$ is a vector of the activation levels of all neurons that have connections to n, $\vec{w_r}$ is a vector of the strengths of those connections, Θ_n is the threshold of node n, and a_r is the resulting activation of neuron n.

$$in_n = \vec{w_n} \cdot \vec{a} = \sum_i w_{ni} * a_i$$

$$a_n = \begin{cases} +1 & \text{if } in_n > \Theta_n \\ 0 & \text{if } in_n \leq \Theta_n \end{cases}$$

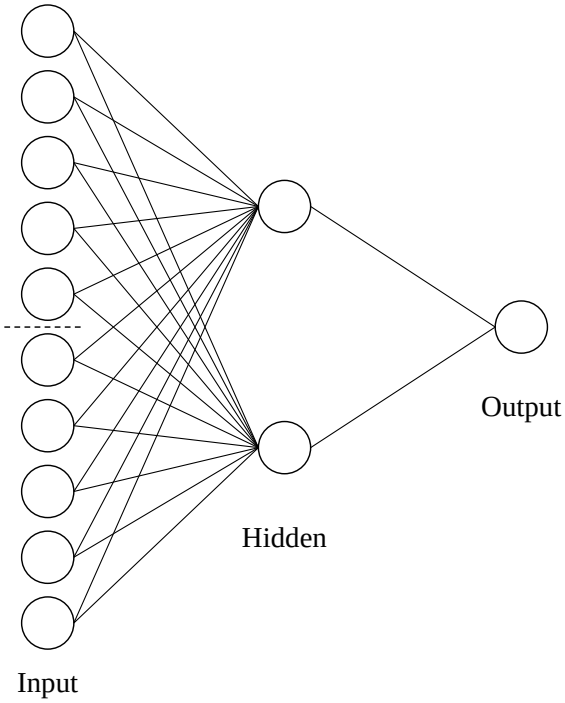

Figure 3.3: A fixed neural architecture for 10-input 1D mirror symmetry task. Circles are used to represent neurons, while arcs represent connections. Input patterns are fed into input neurons (left) and activation spreads through hidden neurons (middle) to a single output neuron (right).

3.4 Genetic Representation and Fitness Function

A linear chromosome of real-valued genes is used to represent the biases and connection strengths in the network. Genes representing biases and connections weights (incoming and outgoing) associated with the first hidden neuron are listed first, followed by genes representing biases and connection weights associated with the second hidden neuron. The gene representing the bias of the single output neuron is listed last. This genetic representation is illustrated in Figure 3.4. Each gene in the chromosome has a genotypic value between 0 and 1, which maps linearly to a phenotypic connection strength

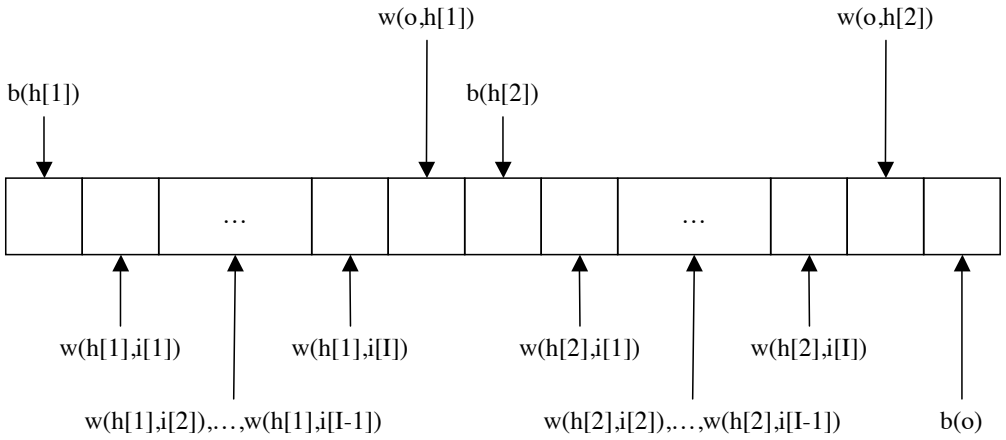

Figure 3.4: Genetic representation of a neural network. Terms are defined as follows: b(h[x]) is the bias of the x^{th} hidden neuron, b(o) is the bias of the output neuron, w(h[x],i[y]) is the connection weight from the y^{th} input neuron to the x^{th} hidden neuron, and w(o,h[x]) is the connection weight from the x^{th} hidden neuron to the output neuron.

between -1 and +1. When using real-valued chromosomes, it is common practice to keep genotypic values in the 0 to 1 range, and map them to whatever range of phenotypic values is desired. This allows for the same genetic mutation operators to be applied to all values in the chromosome, regardless of whether the genes map to different phenotypic ranges.

The fitness of an individual is evaluated as follows. A linear threshold unit neural network is created with the connection weights that are defined by the individual. The network is used to process each possible unique input pattern, and the resulting network outputs are examined. The desired output value for a network is 1 if the input pattern is symmetric and 0 otherwise. The number of patterns for which the network produces the desired output value is recorded, and the fitness value assigned to the individual is the percent of input patterns that are processed correctly by the network.

3.5 Causal Knowledge

Causal knowledge about neural networks could potentially be constructed in multiple ways. For this preliminary work, I took the following approach, forming one type of causal relation. For every input neuron i and hidden neuron h, there is one possible genotypic disorder:

f(i,h): the connection weight from i to h is sub-optimal

Thus there are a total of 2*I possible genotypic disorders, where I is the number of input bits. For each input pattern p, there is one possible phenotypic performance problem:

x(p): network produces incorrect output when presented with input pattern p

There is a causal relationship between each genotypic disorder f(i,h) and each performance problem x(p) based on whether the i^{th} bit of p is on. That is to say, if the connection weight from input i to hidden neuron h is defective, it can cause the network to fail to correctly process inputs in which the ith input is on. For all i, p, and h:

f(i,h) $\rightarrow$ x(p) iff (i^{th} bit of p = 1)

where $\rightarrow$ is used to indicate causality ("may cause"), and not logical implication. If the i^{th} bit of an input pattern is off, connection weights from the

i^{th} input bit will have no impact on the output produced by the network. For this reason, there is no causal relationships between f(i,h) and x(p) for those values of i and p where the i^{th} bit of p is off.

The same piece of causal knowledge can be found implicitly in standard methods for error back-propagation learning in neural networks:

$$\Delta w_{k,j} = \eta \delta_{p,k} x_{p,j}$$

in which $\Delta w_{k,j}$ represents the amount of weight change to the connection between neuron j and k, $\delta_{p,k}$ is the error of neuron k when input pattern p is processed, $x_{p,j}$ is the activation of node j when input pattern p is processed, and η is a constant (Mehrotra, 1997). Note that if the $x_{p,j}$ term is 0, that is if the activation of node j is zero, then there will be no change to the weight from j to k, because that connection cannot be contributing to the error at k.

For a symmetry problem with 10 inputs and 2 hidden neurons, there are thus 20 potential genotypic disorders and 1024 potential performance problems. Each of the 20 genotypic disorders has a causal relationship with 512 of the performance problems. It is worth noting that these 10,240 (20x512) relationships may be equivalently expressed in one parameterized causal relation, as was presented above. This raises the question of whether the causal relations presented here constitute a single relation or many relations. In

my opinion, the single parameterized causal relation is best viewed as a single piece of causal *knowledge* which abstractly describes the 10,240 causal *relations* in the domain.

3.6 Causally-Guided Mutation

When causally-guided genetic operators are applied to an individual network, that network's performance characteristics and the given causal knowledge are used to bias the execution of these operators. As indicated in Figure 3.1, this is accomplished in three steps, as follows. The first step is straightforward: if an individual network fails to process an input pattern p correctly, it is assessed as having the phenotypic symptom s(p). Each assessed symptom provides evidence that the genotypic disorders that it may be caused by may be present in the individual. For example, if the phenotypic symptom s(0000000100) is assessed in an individual, this can be seen as evidence that the genotypic disorders f(8,1) and f(8,2) may be present, i.e., that the weight from input neuron 8 to hidden unit 1 or 2 is not optimal.

In the second step, a heuristic is used to assess the relative likelihood (RL) score for each genotypic disorder. The specific heuristic for the RL score of a genotypic disorder used in this work is

$$\mathrm{RL}(\mathrm{f(i,h)}) = \max(\ |\mathrm{effects(f(i,h))} \cap \mathrm{S}|\ ,\ \tfrac{|S|}{2}\ ) - \tfrac{|S|}{2}$$

$$\mathrm{effects(f(i,h))} = \{\mathrm{x}|\mathrm{f(i,h)} \rightarrow \mathrm{x}\}$$

where S is the set of observed symptoms. Thus, the RL score for a particular disorder is based on the number of observed symptoms that may be caused by that disorder. Note that if a disorder causes less than half of the observed symptoms, it is deemed to be unlikely and its RL score is set to 0. The minimum and maximum possible RL scores of 0 and $\frac{|S|}{2}$ occur when the disorder in question causes less than half or all of the observed symptoms, respectively.

Finally, in the third step, the genetic operators are causally biased. A complicating factor when attempting to evolve connection weights of neural network architectures is the permutation problem (Hancock, 1992). This problem arises from the fact that multiple networks that are functionally equivalent may be genetically different simply because their hidden units are arranged in different orders. The permutation problem makes crossover operators very inefficient and ineffective (Yao, 1999). For example, in some preliminary trials that were performed it was found that evolutionary systems employing crossover were able to successfully produce neural networks for the 8-input symmetry problem only 1/5th as frequently as otherwise equivalent evolutionary systems that employed only mutation. For these reasons, crossover and causally-guided crossover are not employed in this study, and only causally-guided and control mutation genetic operators are used here.

Causally-guided mutation operators are biased such that those parts of the genotype with higher relative likelihoods of being flawed are made

more likely to be mutated. First, a single gene is randomly selected with uniform probability from the set of all genes. If the selected gene represents a hidden-to-output weight, a hidden bias, or the output bias, then the gene value is modified as described below. However, if the selected gene represents an input-to-hidden weight, then causal knowledge is used in reconsidering which input-to-hidden gene should be modified. Specifically, a new input-to-hidden gene is selected from the set of all input-to-hidden genes, with probability proportional to each gene g's RL(f(g)) score. The selected gene is then modified as described below. In this way, causally guided mutation steers modifications to those input-to-hidden connections that have higher relative likelihoods of being flawed. It is worth noting that causally-guided mutation is probabilistic and does not prevent the occurrence of poorly fit individuals that arise in the populations due to random alterations; it simply aims to bias the process towards the formation of more fit individuals and decrease the formation of very poor individuals that often occur with standard evolutionary computation methods.

Whichever gene is ultimately selected for mutation is modified by adding or subtracting (with equal probability) a small randomly generated number to the gene value. If the resulting gene value is outside the legal range ([0,1]), it is incremented / decremented back within range. The small random number is generated in a manner that is reminiscent of a Gaussian distribution, in which the majority of values are clustered about a central mean. However, unlike a normal distribution, the method used here avoids generating num-

bers that are excessively small to the point of being irrelevant. Specifically, the random number is generated from one of the following five uniform distributions: $[\frac{1}{32}, \frac{1}{16}]$, $[\frac{1}{16}, \frac{1}{8}]$, $[\frac{1}{8}, \frac{1}{4}]$, $[\frac{1}{4}, \frac{1}{2}]$, $[\frac{1}{2}, 1]$. Which uniform distribution is used is selected randomly and with equal probability.

The control mutation operator is not causally-guided, but otherwise operates in the same way as the causally-guided mutation operator described above. When control mutation is applied to an individual, exactly one gene is selected uniformly and randomly and its value is mutated as described above.

3.7 Experimental Methods

Two different evolutionary systems were used to adapt symmetry neural networks. One version uses control mutations rather than causally-guided mutations, and serves as the control in these experiments (CONTROL). The other version (CAUSAL) uses causally-guided mutations as describe above. Each of the systems was used to optimize the weights of symmetry networks with 8 inputs and symmetry networks with 10 inputs. One hundred trials of each evolutionary system were run for each symmetry problem, yielding 200 trials overall. Creating successful network weights for the 10 input task requires optimizing 25 real-valued genes/weights to construct a network that correctly processes 1024 input patterns. This is a non-trivial task even for learning methods such as error back-propagation (which could not be used

here as a direct comparison due to the non-differentiable activation functions used). Each evolutionary trial was allowed to run for up to 15000 generations or until a set of network weights that correctly processes all input patterns was found. As discussed earlier, crossover was not used. In each of the experiments conducted, a population size of 500, a mutation rate of 0.80, tournament selection with tournament size of two, and single-individual elitism were used. For each system, the fraction of trials that found optimal network weights was measured at various generations.

3.8 Results

As shown in Figure 3.5 (left), for an 8 input symmetry problem, the CONTROL algorithm was able to produce a successful network in 27 out of 100 trials. In contrast, the matched CAUSAL algorithm was able to produce successful networks 51 times out of 100 trials. This difference in performance is statistically significant at a 99% confidence level. As shown in Figure 3.6 (right) the difference in performance between the CONTROL and CAUSAL systems was even more dramatic when applied to the 10 input symmetry problem. In this problem, the CONTROL system failed to find a successful network in all 100 trials. In contrast, the CAUSAL system was able to find successful networks in 20 out of 100 trials, within 15000 generations.

Exploring the performance of the evolutionary systems in terms of number of generations required does not fully address the computational costs

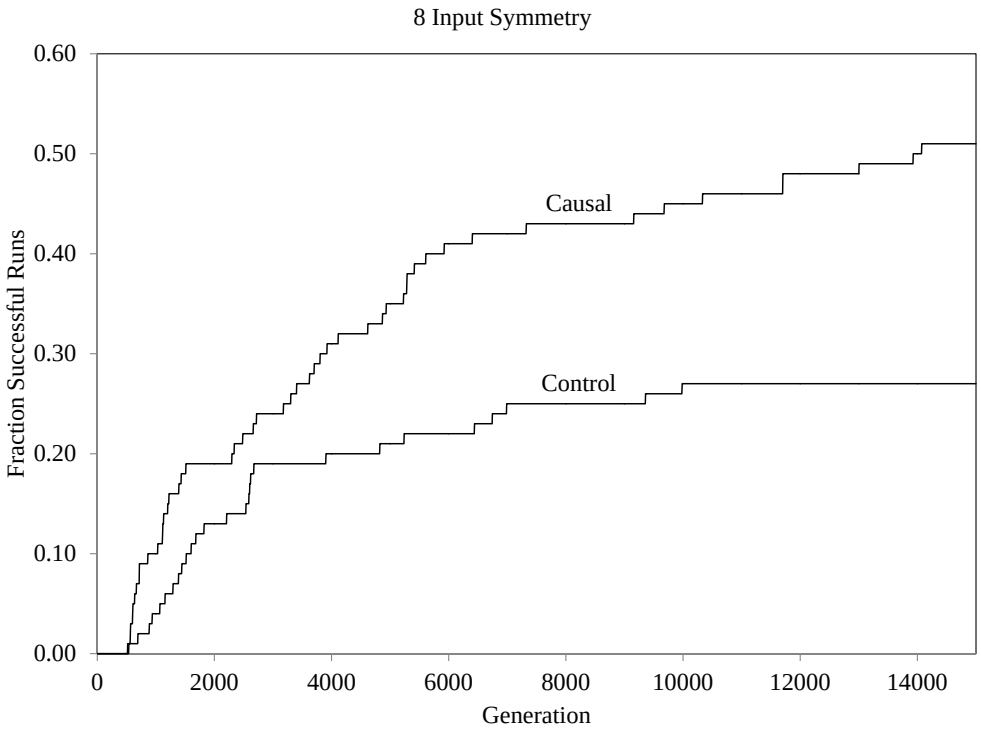

Figure 3.5: Fraction of successful runs when using the CAUSAL and CONTROL algorithms to solve an 8 input mirror symmetry recognition problem.

involved. Specifically, introducing causal guidance into the evolutionary process requires additional computation per generation to make inferences and influence genetic operations. To better understand this issue, 30 additional trials of the CONTROL and CAUSAL system were applied to the 8 and 10-input problems, the CPU time required by each trial was measured, and the differences between the two systems were analyzed.

For the 8-input problem, it was found that a CONTROL system trial averaged 480 seconds to execute 15,000 generations, while a CAUSAL system trial averaged 491 seconds. Similar results were found for the 10-input problem, for which a CONTROL system trial averaged 1753 seconds, and a CAUSAL system trial averaged 1805 seconds. For both problems, at a 95%

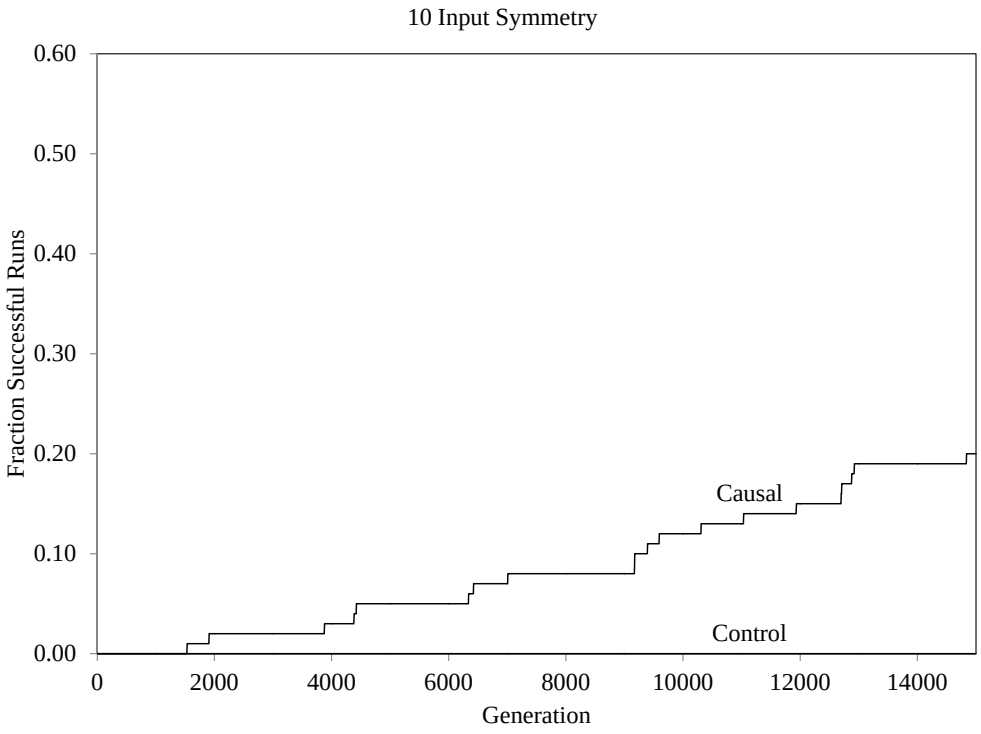

Figure 3.6: Fraction of successful runs when using the CAUSAL and CONTROL algorithms to solve a 10 input mirror symmetry recognition problem. Results with the CONTROL algorithm cannot be seen as all runs failed (i.e., CONTROL results are a horizontal line at 0% that is obscured by the x-axis).

confidence level a CAUSAL system trial requires less than 4% more CPU time than a CONTROL system trial.

3.9 Discussion

In this first chapter, I have taken some initial steps to evaluating the effectiveness of using evolutionary computation methods that have been modified so that casual knowledge can guide the application of genetic operators.

I have used causally-guided evolutionary methods to design neural networks that recognize 1D mirror symmetry. There was no guarantee a priori

that introducing causally-guided genetic operators would make the evolutionary system more effective. It was entirely possible that just the opposite would be true, i.e., adding causal influences could produce evolutionary systems that are less effective and less computationally efficient.

Instead I found that causally-guided evolution could be used successfully to design neural networks that recognize mirror symmetry in one dimensional input patterns. The performance of this system was compared to a carefully matched control evolutionary system that does not employ causal guidance. When applied to an eight input symmetry problem, the systems employing causally-guided mutation found successful networks nearly twice as frequently (88% more frequently) as the system employing non-causally-guided or "control" mutation. Furthermore, when applied to a ten input symmetry problem, the causally-guided system found successful networks 20% of the time, whereas the control system failed to find a successful network in any of the 100 trials. It was found that the causally-guided system required less than 4% more CPU time per generation than the control system. This very marginal increase in computational costs per generation would be overwhelmingly outweighed in practice by the fewer number of generations that would be required to find optimal solutions.

The fact that causally-guided systems were able to solve the symmetry network design task with greater frequency than the control systems provides a first demonstration that incorporating human-provided causal knowledge

into the evolutionary search process can have a meaningful positive impact. The fact that evolutionary systems that employ causally-guided mutation are able to solve the 10-input problem and control systems are not is quite encouraging, but raises the question of whether this result will carry over to real world problems. The next two chapters address this question.

Chapter 4

Integration of Causally-Guided Crossover for Design Optimization

4.1 Introduction

In this chapter the initial work presented in Chapter 3 is extended in a number of important ways, both in terms of the causally-guided evolutionary methods themselves as well as the types of application problems to which causally-guided evolution is applied and evaluated. While the previous chapter described the use of a single causally-guided operator (mutation), in this chapter a second causally-guided operator (crossover) is introduced. An important goal of this chapter is to evaluate the feasibility of causally-guided crossover when used in isolation and when combined with causally-guided mutation in the same evolutionary process. Can multiple causally-guided genetic operators be used synergistically, or do they interfere with each other?

To evaluate these ideas, causally-guided evolutionary systems are developed and applied to a real-world antenna design task. This evaluation is significant for two reasons. First, successfully applying causally-guided evolution to a second application problem demonstrates the general applicability of these methods. Second, the antenna design task is a real-world

problem in which causal knowledge is available but hardly comprehensive. When designing antennas, human experts regularly use vague heuristics or "rules of thumb" to iteratively revise designs based on performance characteristics. However, this knowledge is far from complete and a long trial and error design process is typically needed. While the neural network design task in Chapter 3 was selected for an initial study precisely because causal knowledge was well understood and known to be effective, the application of causally-guided evolution to antenna design is an important step in demonstrating that these methods are applicable even when causal knowledge is less comprehensive.

In the remainder of this chapter, the high-level (i.e., application independent) form of causally-guided crossover is presented. Next, antenna design is discussed in greater detail, including the particulars for the dipole antenna array design task considered here. The causal knowledge that is used in this study (provided by two expert electrical engineers) and the application-specific forms of causally-guided mutation and crossover operators are presented. An experimental evaluation is conducted in which the performance of evolutionary systems that employ causal guidance are compared to carefully matched control systems that do not. Further analysis examines the types of designs that are produced by these various systems and the relationship between those designs and causal guidance are discussed. Evolved optimal antenna designs are systematically varied and their changes in performance examined to validate the causal knowledge that was used in

this study as well as to learn about cause-effect relations in this domain more generally.

4.2 General Form of Causally-Guided Crossover

In Chapter 3, the general form of causally-guided genetic operators and causally-guided mutation was defined. Here, causally-guided crossover is defined analogously as follows: *Causally-guided crossover operations are biased such that those parts of parent individuals' genotypes that have lower relative likelihoods of being flawed are made more likely to be combined together when creating offspring. Consequently, those parts of the individuals' genotypes that have higher relative likelihoods of being flawed are made less likely to be used when creating offspring.* In this way, causally-guided crossover increases the chances that the best parts from each parent are combined into the produced offspring.

As with all causally-guided genetic operators, the causal guidance here is used to bias but not explicitly control crossover. The use of causally-guided crossover is probabilistic and does not prevent the occurrence of poorly fit individuals that arise in the population due to random alterations; it simply influences the process towards the formation of more fit individuals and fewer very poor individuals than would otherwise occur.

4.3 Dipole Antenna Array Design

To assess the effectiveness of these ideas, I explore the use of causally-guided evolution to solve an antenna array design problem. A generational genetic algorithm augmented with causally-guided genetic operators was developed, and its performance in solving the antenna array design problem was compared to a carefully-matched genetic algorithm that uses no causal-guidance but is otherwise equivalent. Unlike the previous chapter, a causally-guided crossover operator is defined and used, in addition to causally-guided mutation. The effects of using both causally-guided genetic operators together and in isolation are examined. With the exception of the causally-guided genetic operators, all aspects of the evolutionary systems (described in more detail below) are conventional and widely used (De Jong, 2006). The goal of these experiments is to determine if causally-guided genetic operators mislead the evolutionary process toward local minima, have no significant effect, or improve the quality of and speed with which solutions are produced.

4.3.1 Evolutionary Antenna Design

The design of many real-world antennas is a challenging problem that requires significant domain expertise (Altshuler and O'Donnell, 2011; Drabowitch et al., 1998; Elliot, 2003; Setiean, 1998). Complex interactions between neighboring components of an antenna make it impossible to solve all but the most simple antenna design problems in closed form. Instead, human

designers typically employ an iterative trial-and-error design process, which is both time and labor intensive and often results in very simple antenna designs that may not be optimal. While there have been a few previous studies of evolving antenna arrays, most past related work has focused on evolving single isolated antennas.

These difficulties have motivated research into automated methods for antenna design. While various methods have been studied (including particle swarm optimization (Robinson and Rahmat-Samii, 2004), ant colony optimization (Rajo-Iglesias and Quevedo-Teruel, 2007; Panduro et al., 2009), and simulated annealing (C.M. Coleman and Ross, 2004)), evolutionary computation methods appear to be particularly well-suited to the antenna domain, and have been successfully used to design Yagi-Uda antennas, quadrifiliar antennas, and crooked wire monopole antennas (Luo et al., 2010; Siakavara, 2010; Haupt and Werner, 2007; Lohn et al., 2002, 2004, 2008, 2001). The proven ability of evolutionary methods to effectively search large and unknown design spaces make them a natural fit for antenna design problems (Lohn et al., 2001). Compared to optimization techniques such as gradient descent and hill-climbing, evolutionary methods have been shown to be less susceptible to getting stuck in local optima when applied to problems in which the fitness landscape may be discontinuous and include numerous local optima, as is the case with many antenna design problems (Haupt, 1995).

While past studies have shown the effectiveness of applying conventional evolutionary computation methods to antenna design problems, these

methods do not effectively leverage available human expertise regarding cause-effect dynamics in the application domain. As others have noted, effective antenna design by humans requires not only knowledge and intelligence about antennas, but also experience and artistry (Lohn et al., 2001). Indeed, human antenna design experts often modify and refine existing antenna designs based on intuitions and rules-of-thumb that have been acquired through years of experience. In contrast to conventional evolutionary methods, causally-guided evolutionary computation is designed specifically to leverage this cause-effect knowledge in order to guide the evolutionary process. To our knowledge, no antenna system has ever been designed through causally-guided evolution, as described here.

4.3.2 Antenna Performance Characteristics

There are a number of performance characteristics that are important to antenna design, and that will be used later in this chapter. The term *directivity* refers to the capability of an antenna to radiate more energy in certain directions than in others. *Gain* is a measure of the amount of energy that an antenna radiates in a specific direction. It is calculated by computing the ratio of energy radiated in that direction to the amount that would be radiated by an isotropic radiator (a theoretical antenna that radiates equally in all directions). Gain often has very high values and is most often expressed in decibels. Another important characteristic is the impedance mismatch be-

tween transmission lines and antenna, which can cause electrical signals to reflect back through the feed network. These reflected waves react constructively and destructively with incoming signals, resulting in peaks and valleys in the signal envelope along a transmission line. At best, impedance mismatches in an antenna result in inefficient use of energy. In some cases, the constructive interference can even result in voltages that are high enough to damage circuitry in the feed network. *Voltage standing wave ratio* (VSWR) quantifies impedance mismatch between transmission lines and radiating elements (VSWR = 1 is ideal). Finally, with antenna design problems, cost can mean many different things, including manufacturing difficulty, weight, size, volume of material, etc.

4.3.3 Dipole Antenna Arrays

The specific task used in this work is that of designing a directional dipole antenna array that meets prespecified performance criteria. Dipole antenna arrays consist of an array of parallel lengths of wires, known as dipoles, which are positioned above a ground plane. A transmission line connects to the center of each dipole and carries the signal that is radiated or received by the antenna. The complete design specifications for such an antenna include the number of dipoles, the lengths of dipoles, the height of dipoles off the ground plane, the spacing between dipoles, and the phases and voltages with which each dipole is fed. For this work, dipole antenna

arrays were limited to having uniform spacing, height, and length. Known formulas are used to calculate the desired voltage and phase with which each dipole should be fed, based on each dipole location and the desired direction of broadcast. The uniform nature of such designs makes them appear to be very simple. However, in practice and despite the small dimensionality of the search space, it is quite difficult for a human designer to optimize these four values by hand because of the rugged fitness landscape. Furthermore, greedy or local search algorithms often get stuck in the many local optima that exist.

4.3.4 Performance Criteria

In the particular antenna design task considered here the specific goal, provided a-priori, is to maximize gain between -10 and +10 degrees off boresight in the plane that bisects the dipoles, minimize VSWR, and minimize cost. Specifically, a successful antenna must have an average gain of at least 10 dB in the target angle range and a VSWR of less than 3.0 (a commonly used limit for VSWR in antenna design). The number of dipoles in the antenna array is used as a rough approximation for cost, which should be minimized but does not have a required value. The antenna is to be operated at 1200 MHz with 50 ohm transmission lines. These particular design specifications and performance requirements define an easily understandable antenna design problem that is complex enough to be of real-world interest.

As in previous studies, the antennas are simulated over an infinite ground plane in order to keep computational costs down (Lohn et al., 2004). Software was implemented in Java and C, and runs on Linux-based PC's. All antennas were simulated using an open source version of the Numerical Electromagnetic Code software package (Burke and Poggio, 1981) available at http://www.si-list.net/swindex.html.

4.4 Fitness Evaluation and Genetic Representation

A fitness function was designed to capture the specific performance criteria outlined above. The overall fitness function is made up of three components that reflect the three distinct performance criteria:

$$Fitness_{Overall} = Fitness_{VSWR} + Fitness_{Directivity} + Fitness_{Cost}$$

The $Fitness_{VSWR}$ component rewards low VSWR values, and was calculated as:

$$Fitness_{VSWR} = \begin{cases} -2 * VSWR_{max} & \text{if } VSWR_{max} \geq 3.0, \\ -1 * VSWR_{max} & \text{if } VSWR_{max} < 3.0. \end{cases}$$

where $VSWR_{max}$ is equal to the maximum VSWR observed at any dipole in the antenna. If the $VSWR_{max}$ is above 3.0, the $Fitness_{VSWR}$ score is multiplied by -2 instead of -1, increasing its negative impact on the overall fitness value. This was done because, as noted earlier, a VSWR value of 3

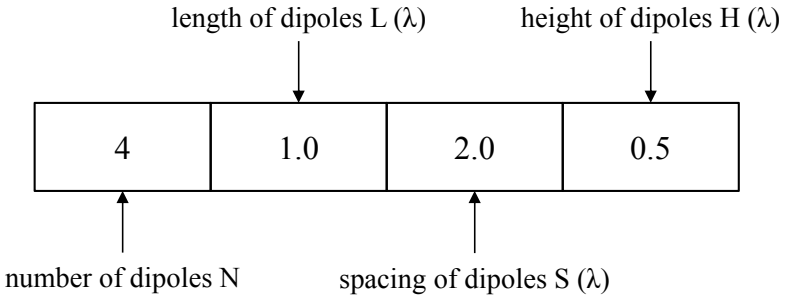

Figure 4.1: Genetic representation of a dipole antenna array.

is a commonly used limit in antenna design, beyond which antennas are not useable. The $Fitness_{Directivity}$ score measures the directivity of the antenna in the target angle range, and is calculated as:

$$Fitness_{Directivity} = \sum_{i=-10}^{10} Gain_i$$

where $Gain_i$ is the amount of gain observed in the XY-plane at i degrees off boresight. There are 21 terms here, each in the -10 to +10 degree range, so an antenna that meets the design requirements of 10 dB average in this range would have a $Fitness_{Directivity}$ score of at least 210. Lastly, the $Fitness_{Cost}$ component is equal to the number of dipoles in the antenna array multiplied by -1. These three components are summed to yield the overall fitness score ($Fitness_{Overall}$), which is maximized by the evolutionary system. This multi-component fitness function, in which each component reflects a different performance goal, and the stepped nature of the VSWR component are inspired by previous studies showing such an approach to be successful when evolving antenna designs (Hornby et al., 2006; Lohn et al., 2004).

The genetic representation used to represent a dipole antenna array consists of a vector of four numbers (see Figure 4.1). The first is a whole number between 1 and 10 that represents the number of dipoles. The remaining numbers in the vector are real-valued and may have values between 0.1 and 2.0. These three values represent distances, expressed in wavelengths, for the length of dipoles, height of dipoles and spacing between dipoles. Since the operating frequency of this antenna is 1200 MHz, a wavelength corresponds to roughly 0.25 meters. For example, an individual in the population with genetic vector of [4, 1.0, 2.0, 0.5] describes an antenna with 4 dipoles of 0.25 m length, spaced 0.5 m apart and 0.125 m off the ground plane. The four vector elements are distinct parts of the genome and are referred to as *genes*. The letters N, L, S, and H are used to refer to these four genes.

4.5 Causal Knowledge

The non-linear interactions between antenna design aspects and performance qualities are not easily characterized and are quite complex. Here the case in which only a few pieces of diagnostic causal knowledge are incorporated into the system is considered. This allows an exploration into the feasibility of causally-guided evolution, while delaying the need to seriously address complex situations in which multiple genotypic disorders influence multiple phenotypic symptoms simultaneously. It is known by antenna design experts that sub-optimal dipole lengths can cause high VSWR, but that

the height, length, and number of dipoles have limited effect on the VSWR of an antenna [1]. This is the diagnostic causal knowledge that I use in this study:

Sub-optimal Dipole Length → High VSWR

The defects on the left side of these causal relations are genotypic disorders because they refer to sub-optimal gene values in the genetic representation (see Figure 4.1). The right side of these causal relations is a phenotypic symptom. As noted earlier, a high VSWR value is an indication of impedance mismatch between transmission lines and dipoles. Such impedance mismatches result in inefficient use of energy, and can even result in voltages that are high enough to damage circuitry in the feed network. As discussed earlier, the → symbol is not logical implication but causality. There are three additional genotypic disorders: sub-optimal number of dipoles, sub-optimal spacing of dipoles, and sub-optimal height of dipoles. These three genotypic disorders do not cause high VSWR. The terms d(N), d(L), d(S), and d(H) are used to represent the four genotypic disorders.

4.6 Causally-Guided Genetic Operators

As discussed in Section 3.2, causally-guided genetic operations occur in three steps: the performance characteristics of the individuals in question

[1] These causal relationships were derived with assistance from electrical engineers who are experts in antenna design - Jason Lohn at CMU and Derek Linden of X5 Systems.

are examined to assess phenotypic symptoms, inferences are made about the likelihoods of the genotypic disorders, and these inferred likelihoods are used to bias the execution of the genetic operator. In the antenna array design task, identifying phenotypic symptoms (Step 1) and making inferences about the relative likelihood of genotypic disorders (Step 2) is straightforward. If an antenna has a VSWR greater than 3.0, it is assessed as having the phenotypic symptom of high VSWR, otherwise it is assessed as not having the symptom of high VSWR. An antenna that has the symptom of high VSWR can be reasoned to have an increased chance of having sub-optimal dipole lengths, as this is the only genotypic disorder that is known to cause the symptom. On the other hand, if the antenna lacks the symptom of high VSWR, it can be reasoned that there is a decreased chance that the antenna has sub-optimal dipole lengths. Finally, the way in which these inferences are used to bias the execution of genetic operators (Step 3) depends on the particular genetic operator in question. Because real probability values of causal relationships were not available, the heuristic methods for causal guidance presented below were developed based on intuition and not on any formal probabilistic reasoning. However, as described in the Appendix to this chapter, these heuristic methods are consistent with probabilistic approaches.

When an individual is selected for causally-guided mutation, there are a number of specific mutations that may or may not be applied to the individual, and these are controlled by the algorithm Causal-Mutation that is outlined in Figure 4.2. The decision of whether to apply each specific mu-

tation is made stochastically and independently, and is influenced by causal guidance. There are eight specific mutations that may be applied to an individual during each causally-guided mutation: for each of the four genes there is a specific mutation that makes large changes to the gene value, and a specific mutation that makes small changes. In this manner, large changes allow the evolutionary process to make big steps through the solution space, while small mutations make finer-grained changes to a gene's value. The terms M_N, m_N, M_L, m_L, M_S, m_S, M_H, and m_H are used to refer to these eight specific mutations. A lowercase m is used for small mutations, an uppercase M is used for large mutations, and single character indices are used to identify the relevant gene.

Each large specific mutation replaces the relevant gene with a random value, selected uniformly from the appropriate legal range for that gene. Thus large specific mutations very often (though not always) make large changes to the gene value. For example, if M_S is applied to an individual, that individual's spacing gene will be set to a random value between 0.1 and 2.0. If m_N is applied to an individual, the value of the gene that specifies the number of dipoles is either incremented or decremented by 1, with equal probability. For the other three genes, small mutations either increase or decrease the gene's value by a small random value. This small random value is selected with equal probability from one of five uniform distributions: [0, 0.2], [0, 0.1], [0, 0.01], [0, 0.001], and [0, 0.0001]; this has the effect of making smaller changes (e.g., < 0.0001) more probable than other small changes

(e.g., 0.1 to 0.2). These small mutations allow the evolutionary process to make smaller moves through the solution space and fine tune solutions.

When causally-guided mutation is applied to an individual, the probabilities with which each of these specific mutations are applied is biased based on the likelihoods of the various genotypic disorders (see Figure 4.2). Specifically, a utility score is calculated for each specific mutation. The utility score of a specific mutation is used as an indication of how useful it would be to apply that specific mutation to the individual. Initially, each specific mutation is assigned a utility score of 1.0. For each gene that has an increased likelihood of being flawed (i.e., the corresponding genotypic disorder has an increased likelihood of being present), the utility value of the corresponding large mutation is increased by multiplying by a constant $\Delta_1 > 1.0$. For those genes with lower likelihoods of being flawed, the utility score of the corresponding large mutation is decreased by dividing by a constant $\Delta_2 > 1.0$. Lastly, the utility scores are rescaled so that the sum of all eight specific mutations' utility scores is equal to one. Thus large mutations that correspond to genes with higher relative likelihoods of being flawed have higher utility scores. Each specific mutation is then applied with probability equal to its utility score. It is worth nothing that, because the utility scores are scaled to sum to 1.0, the expected number of specific mutations that will be applied during any single causally-guided mutation operation is 1.0.

For example, when causally-guided mutation is applied to an antenna design with the symptom of high VSWR, the causal knowledge indicates an

```
function CAUSAL-MUTATION(Individual A)
    // STEP 1: assess phenotypic symptoms
    if (A has VSWR > 3.0) then
        hasSymptom(A, High VSWR) = true
    else
        hasSymptom(A, High VSWR) = false
    end if
    // STEP 2: make inferences about likelihood of genotypic disorders
    increasedLikelihood(A, d(g)) = false ∀ g ∈ {N,L,H,S}
    decreasedLikelihood(A, d(g)) = false ∀ g ∈ {N,L,H,S}
    if (hasSymptom(High VSWR)) then
        increasedLikelihood(A, d(L)) = true
    else
        decreasedLikelihood(A, d(L)) = true
    end if
    // STEP 3: bias mutation
    utility(s) = 1.0 ∀ specific mutations s
    for all genes g ∈ {N,L,H,S} do
        if (increasedLikelihood(A, d(g))) then
            utility(M_g) = utility(M_g) * Δ1
        else if (decreasedLikelihood(A, d(g))) then
            utility(M_g) = utility(M_g) / Δ2
        end if
    end for
    rescale all utility scores to sum to 1.0
    for all specific mutations s do
        if RANDOM(0,1) < utility(s) then
            apply s to A
        end if
    end for
end function
```

Figure 4.2: Pseudocode for the causally-guided mutation operator as implemented for the dipole antenna array design task. Input argument A is an individual antenna array design, while d(g) represents the assertion that gene g of A is flawed (see text). Mutations s refer to the 8 specific mutations listed in the text. Constants Δ_1 and Δ_2 are constrained to be greater than 1.0, and RANDOM(0,1) returns a uniformly random floating point number in (0.0, 1.0).

```
function CAUSAL-CROSSOVER(Individual M, Individual F)
    // STEP 1: assess phenotypic symptoms
    for all parents P ∈ {M, F} do
        if (P has VSWR > 3.0) then
            hasSymptom(P, High VSWR) = true
        else
            hasSymptom(P, High VSWR) = false
        end if
    end for
    // STEP 2: make inferences about likelihood of genotypic disorders
    increasedLikelihood(P, d(g)) = false ∀ g ∈ {N,L,H,S}, P ∈ {M, F}
    decreasedLikelihood(P, d(g)) = false ∀ g ∈ {N,L,H,S}, P ∈ {M, F}
    for all P ∈ {M, F} do
        if (hasSymptom(P, High VSWR) then
            increasedLikelihood(P, d(L)) = true
        else
            decreasedLikelihood(P, d(L)) = true
        end if
    end for
    // STEP 3: bias crossover
    create child Individual C
    inh(P, g) = 1.0 ∀ g ∈ {N,L,H,S}, P ∈ {M, F}
    for all P ∈ {M,F} do
        for all genes g ∈ {N,L,H,S} do
            if increasedLikelihood(P, d(g)) then
                inh(P,g) = inh(P,g) - Δ3
            else if decreasedLikelihood(P, d(g)) then
                inh(P,g) = inh(P,g) + Δ3
            end if
        end for
    end for
    for all genes g ∈ {N,L,H,S} do
        if RANDOM(0,1) < (inh(M,g) / (inh(M,g) + inh(F,g))) then
            replace C's value of g with M's
        else
            replace C's value of g with F's
        end if
    end for
    return C
end function
```

Figure 4.3: Pseudocode for the causally-guided crossover operator as implemented for the dipole antenna array design task. Same notation as in Figure 4.2, where $0 < \Delta_3 < 1$ is a constant. A parent P's inheritance score for gene g is given by inh(P,g), as explained in text.

increased probability that dipole lengths are sub-optimal. Thus, the utility score of the specific mutation M_L, which makes changes to the gene associated with dipole length, is increased, while the utility scores of all other specific mutations are effectively decreased through the normalization process. Thus, the specific mutation M_L will be applied with a higher probability than each of the other specific mutations. In this way, causal guidance biases the mutation operator towards modifying those parts of the individual that have higher relative likelihoods of being flawed.

As implemented in algorithm CAUSAL-CROSSOVER (see Figure 4.3), causally-guided crossover is a variation of uniform crossover, in that offspring are created by stochastically selecting one copy of each gene from each of the two parents M and F, and each gene is inherited independently. Unlike in typical uniform crossover, where two offspring are produced simultaneously, the genes that are inherited by one offspring have no influence on which genes are inherited by the other offspring. The inferred likelihood of the various genotypic disorders is used to bias the crossover operation as follows. As shown in Figure 4.3, each gene in each parent is initially assigned an *inheritance score* (inh) of 1.0. The inheritance score of a gene is designed to be an indication of how unlikely it is that the gene is sub-optimal, and accordingly how useful it would be for offspring to inherit that gene. The inheritance score of each gene that has an increased likelihood of being flawed is decreased by subtracting the constant $0 < \Delta_3 < 1$. This same constant is added to the inheritance score of each gene that has a decreased likelihood

of being flawed. The chance that an offspring will inherit a copy of a gene from one parent is equal to that parent's gene's inheritance score divided by the sum of both parents' genes' inheritance scores. In this manner, those genes in a parent with high relative likelihoods of being flawed will have lower inheritance scores and therefore be less likely to be inherited by offspring. Compared to more commonly used crossover operators, such as single-point crossover, uniform crossover can sometimes be very destructive, due to the high number of crossover points. However, for the antenna array design task the chromosome consists of only four genes, which limits the potential for overly destructive crossover. Additionally, using a variant of uniform crossover allows for using causal guidance to steer the inheritance of each gene independently.

The *control mutation* and *control crossover* operators are not biased by causal reasoning, but operate very similarly to their causally-guided counterparts. As their names suggest, control genetic operations are used to serve as a baseline to which one may compare the effectiveness of causally-guided operations. During control mutation, each of the eight specific mutations may or may not be applied to an individual, just as in causally-guided mutation. However, in control mutation, the probability of applying each specific mutation is fixed at $\frac{1}{8}$. In this sense, the control mutation is exactly like the causally-guided mutation, except that probabilities of applying specific mutations are fixed and not influenced by causal reasoning. During control crossover, offspring are created by stochastically selecting one copy of each

gene from one of the two parents, just as in causally-guided crossover. However, in control crossover, each gene that is inherited has an equal chance of coming from either of the two parents. Control crossover can be viewed as being exactly like causal crossover, except that the probabilities that govern from which parent a gene will be inherited are fixed and not influenced by causal reasoning. The fact that control operators are so similar to their causally-guided counterparts helps to ensure that any differences in performance between systems employing causal and control operators are due only to the presence or absence of causal guidance. What is critical for this current comparative study is that the causally-guided and control crossover operators are identical except for the use of causal guidance.

4.7 Experimental Methods

Four different evolutionary systems were used to design dipole antenna arrays that satisfy the pre-specified performance criteria described above. The CONTROL system used control crossover and control mutation operators, and serves a control process in these experiments because it makes no use of causally-guided genetic operators. In contrast, causal mutation and control crossover were used by the CAUSAL_{M} system, control mutation and causal crossover were used by the CAUSAL_{C} system, and both causal mutation and causal crossover were used by the $\text{CAUSAL}_{\text{CM}}$ system. Two hundred trials of each of these four evolutionary systems were conducted using a dif-

ferent random number stream for each trial. Each trial was started with a randomly generated initial population of 50 antennas and executed for 1000 generations, yielding 50,000 antenna array simulations. Each individual in the initial population was created by selecting gene values uniformly from the range of legal values. A population of size 50 and tournament selection with tournament size two were used in all trials. In each generation, exactly one offspring was created by elitism. Each of the remaining 49 offspring in each generation was created by using exactly one of the following stochastically chosen operators: crossover (47.5% chance), mutation (47.5%), or reproduction (5%). Thus, the number of offspring created by each method in each generation was not constant or predetermined. However, given a population size of 50, in each generation the expected number of individuals created by crossover, mutation and reproduction were approximately 23.3, 23.3, and 2.4, respectively (with an additional single offspring via elitism). The constant values Δ_1, Δ_2, and Δ_3 were fixed at 20.0, 2.0, and 0.2, respectively. These parameter values were found via a small number of test runs; they may not be optimal, but were found to be effective in this study, and were the same in both control and experiment trials.

The results of the 800 trials (200 trials times 4 evolutionary systems) were examined and analysis was performed as follows. Of all the antenna designs produced by the evolutionary systems, there appears to be a clear delineation between those that reach a fitness level of 310 and those that do not. That is, the fitness of each antenna produced by each evolutionary pro-

cess is either just above 310 or else considerably lower. Thus, a fitness level of 310 offers useful criteria by which to classify antennas as being "optimal" for the purpose of analysis. Data was collected to determine how often each of the four evolutionary systems was able to find an optimal antenna within various numbers of generations. Additionally, the average number of generations required by each system to find an optimal antenna was calculated. When computing these averages, trials that did not find an optimal antenna design in 1000 generations were counted as having found one in generation 1000. Therefore, this average is actually a rough approximation of the true average.

Multi-start strategies, in which evolutionary processes are terminated and restarted if an adequate solution is not found within a certain generational limit, often find adequate solutions faster than by running a single process indefinitely (Hornby et al., 2006). This is because once an evolutionary process converges to a solution it is unlikely that the process will move away from that solution. Accordingly, additional computation time may be better spent starting a new evolutionary process from scratch rather than continuing the already-converged process.

A multi-start strategy was not used in the 200 trials of each system done in this work. However, by using the results of each system's 200 trials as an approximation for how that system performs across all random number streams, one can calculate the expected number of generations required by

each system to find an optimal antenna when used in conjunction with a multi-start strategy, thus providing a measure of computational cost that is very practical. To do this, $E(req\_gens(S, f, g))$ is calculated, which represents the expected value of the number of generations required by system S to find an antenna with fitness f when used with a multi-start strategy and a generation limit of g. Here $success\_rate(S, f, g)$ is the fraction of system Ss 200 trials that find an antenna with fitness of at least f by generation g, and $avg\_gens(S, f, g)$ represents the average number of generations required by these trials.

$$E(req\_gens(S, f, g)) = \left(\frac{g}{success\_rate(S,f,g)}\right) - g + avg\_gens(S, f, g)$$

This follows from probability formulas related to Bernoulli trials, as each evolutionary process can be thought of as a Bernoulli random variable and the multi-start approach is a Bernoulli trial. Because it is difficult to know a good generation-limit value a priori, the expected number of generations required by each system to find an optimal antenna was calculated with a variety of generation-limits: 100, 200, 300, 400 and 500. This gives a very practical measure of the different computational costs associated with using each of the four evolutionary systems to find an optimal antenna design.

The various antenna designs produced by the four systems were examined. The antennas were visually inspected, found to fall into clusters according to their genotype similarity, and the clusters were arbitrarily labeled. The frequencies with which the various systems arrived at these dif-

ferent designs were calculated, in an effort to understand the ways in which causal-guidance affects the types of designs that are produced.

Introducing causal guidance into the evolutionary process requires additional computation per generation relative to control processes in order to make inferences and influence genetic operations. Causally-guided evolutionary computation is only worthwhile if the additional computational costs per generation are outweighed by the reduced number of generations required to find solutions. The analysis presented above explores performance in terms of the number of generations required, which does not directly address the computational costs involved. For this reason, 5 trials of the CONTROL and $\text{CAUSAL}_{\text{CM}}$ systems were executed for 5000 generations, the CPU time required by each trial was measured, and differences between the two systems were analyzed.

4.8 Results

Numerous trials from all four evolutionary systems successfully designed antennas that met the prespecified performance criteria. The most fit individual, which was discovered by some trials in each system, was a five-element dipole array with dipoles of length 0.4635 λ, height of 1.7094 λ off the ground plane, and spacing of 0.6956 λ. This antenna had a VSWR of only 1.31 and the total directivity score was just over 316, indicating an average of just over 15 dB of gain in the target range. A schematic of this

antenna design and a radiation plot illustrating its directivity can be seen in Figure 4.4, and 4.5. The ground plane, which would occupy the XY-plane where Z is equal to 0, and the feed-lines are not pictured. The radiation plot can be thought of as corresponding to the XZ-plane where Y is equal to 0, which bisects the dipoles (main lobe points in the positive Z-direction). The axis labeled 0 deg in Figure 4.5 corresponds with the positive Z axis in Figure 4.4. The radiation plot is in terms of gain, which simply shows relative strength in particular directions. The overall fitness score of this most fit antenna was 310.04, compared to typical fitness values of 250 to 290 in the initial generation. The distribution of the fitness values of the evolved antennas is such that there is a clear delineation between the fittest antennas and the less fit ones. For the most part, evolved antennas either have a fitness score of just over 310 or a fitness score that is much lower (< 308.5). As noted earlier, any antenna with a fitness of 310 or higher is considered to be an "optimal" antenna design.

A higher fraction of the causally-guided evolutionary systems' trials than control systems' trials found an optimal antenna design within 1000 generations. An individual trial is said to be successful by generation g if it finds an optimal antenna design (as defined above) at or before generation g. Figure 4.6 illustrates, for each of the four evolutionary systems, the fraction of trials that were successful by generation 100, 250, 500 and 1000. At each generation listed, the causally-guided systems found optimal antennas with greater frequency than the control system. Furthermore, the performance of

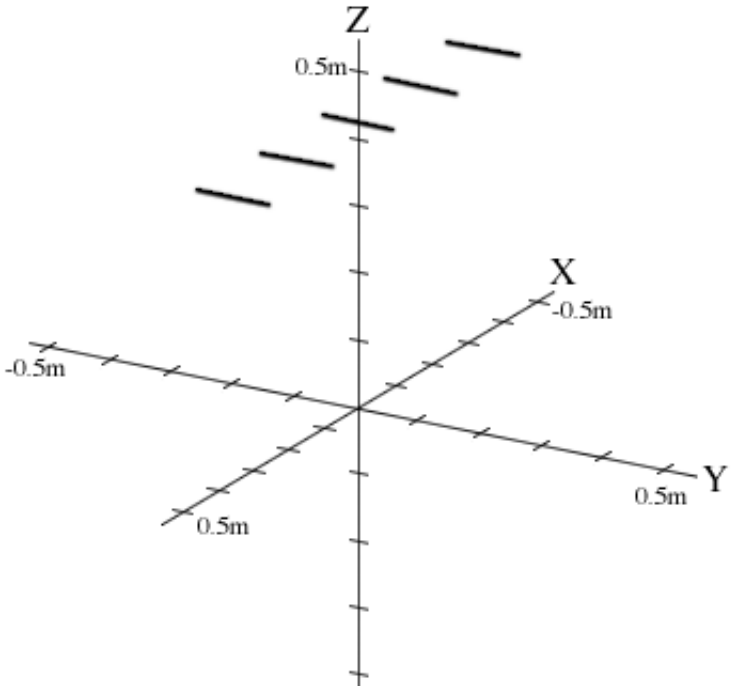

Figure 4.4: A schematic of the fittest evolved antenna. The ground plane is located in the XY plane (not shown).

the four systems relative to each other appears to be the same in all generations. $\text{CAUSAL}_{\text{CM}}$ outperforms CAUSAL_{M}, which outperforms CAUSAL_{C}, which outperforms the CONTROL system. A z-test revealed the difference between $\text{CAUSAL}_{\text{CM}}$ and CONTROL to be statistically significant at a 99% confidence level at generation 250, 500 and 1000. At generation 1000 and 500, the differences between all pairs of systems were statistically significant at a 95% confidence level, with the exception of CONTROL and CAUSAL_{C}, which still had a low p-value of less than 0.10 in generation 1000. Note that the 99% confidence intervals of CONTROL and $\text{CAUSAL}_{\text{CM}}$ never overlap.

The average number of generations required by each evolutionary system to find antenna designs with scores of 308, 309, and 310 are illustrated in

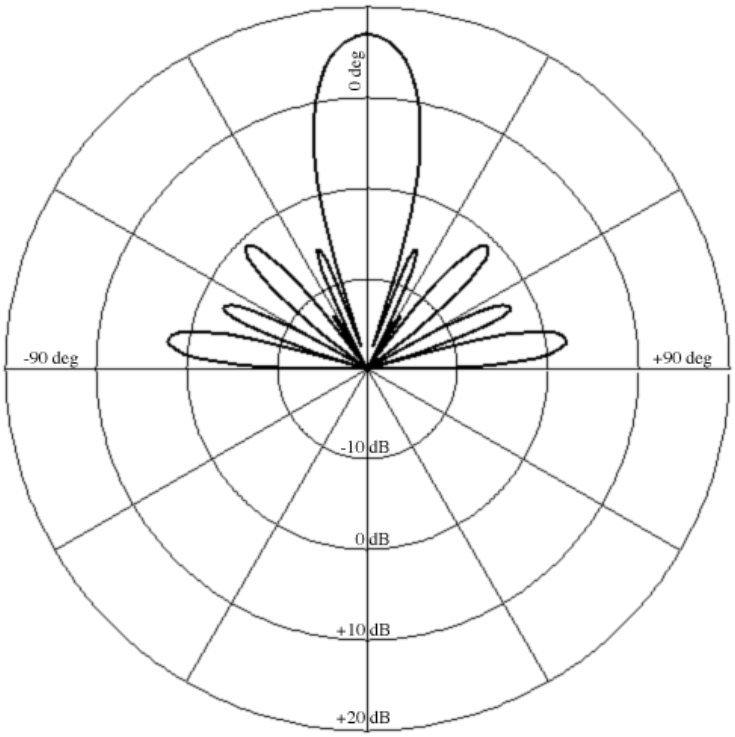

Figure 4.5: A radiation plot illustrating the directivity of the fittest evolved antenna. The ground plane is located in the XY plane (now shown).

Figure 4.7. For each fitness score, the $\text{CAUSAL}_{\text{CM}}$ system averaged the lowest number of generations, followed by CAUSAL_{M}, CAUSAL_{C}, and CONTROL. The $\text{CAUSAL}_{\text{CM}}$ system averaged less than 16%, 29%, and 42% as many generations as the CONTROL system. The differences between $\text{CAUSAL}_{\text{CM}}$ and all other systems, as well as CAUSAL_{M} and all other systems, was statistically significant to a 99% confidence level, for each fitness score. The difference between the CAUSAL_{C} system and the CONTROL system was statistically significant to a 95% confidence level.

By using the 200 trials for each system as an approximation for how the system performs across all initial random seeds, one is able to calculate the

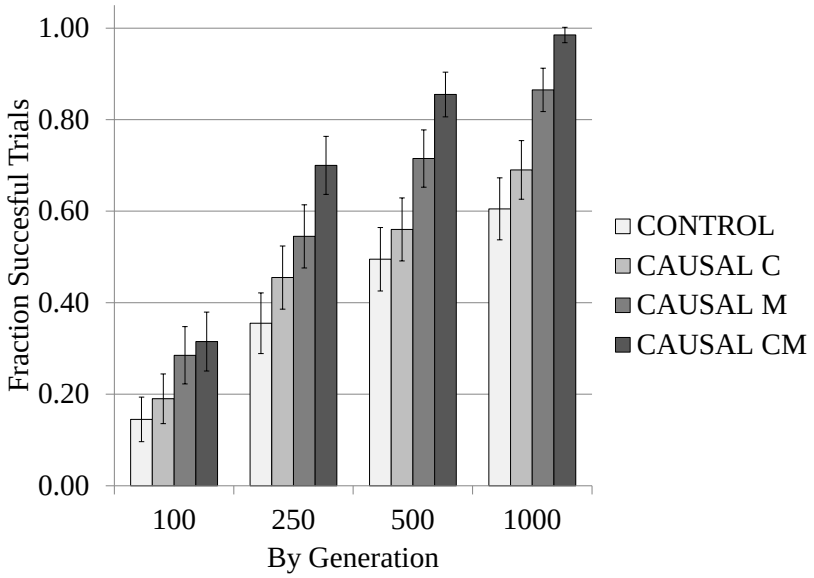

Figure 4.6: Fraction of trials of each system that find an optimal antenna design within 100, 250, 500 and 1000 generations. Vertical bars are used to illustrate a 99% confidence interval.

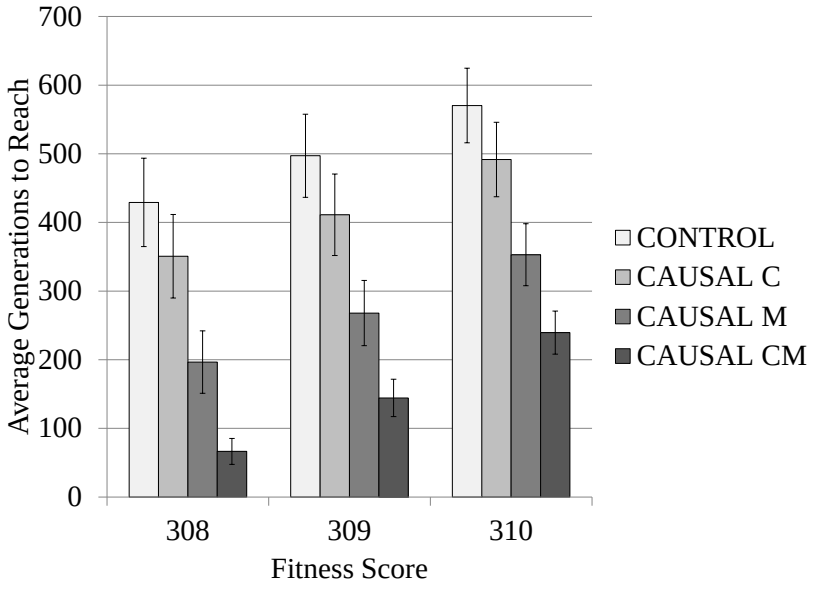

Figure 4.7: The average number of generations that each system requires to find antenna designs of various fitness scores. Vertical bars are used to illustrate a 99% confidence interval.

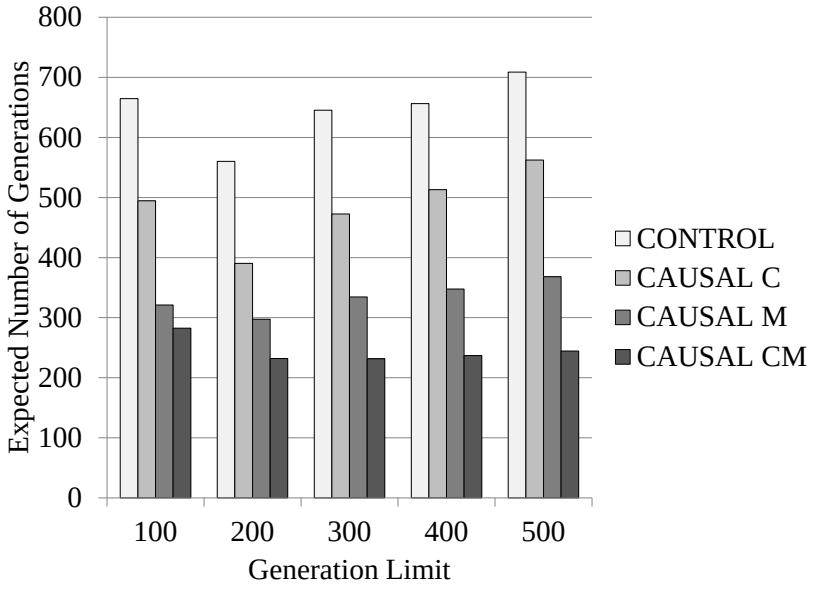

Figure 4.8: Expected number of generations required by each system to find an optimal antenna design when used in conjunction with a multi-start strategy, with varying generation limits.

expected number of generations required to find an optimal antenna, when used in conjunction with multi-start strategies using generation limits of 100, 200, 300, 400 and 500 (see Section 4.7). As illustrated in Figure 4.8, all of the causally-guided systems outperformed the CONTROL system, regardless of which generation limit was used. The $\text{CAUSAL}_{\text{CM}}$ system has the lowest expected value, followed by CAUSAL_{M} and CAUSAL_{C}. The $\text{CAUSAL}_{\text{CM}}$ system requires less than 43% as many generations as the CONTROL system does, regardless of the generation limit.

It was also found that each of the most fit antenna designs that were produced by each trial of each evolutionary system may be grouped, based on the similarity of their genotypes, into one of seven design categories. The mean values of the design aspects and performance characteristics of anten-

nas from each of these categories are detailed in Table 4.1, and the categories are assigned arbitrary labels A through G. In categories A through F, there is very little variation among antenna designs. The maximum Euclidean distance of any antenna from the average characteristics of its assigned category is less than 0.08. Category G captures 3 outlier antennas that do not fit into any of the other six categories. These outlier antennas are tightly clustered in terms of dipole length, dipole height, and dipole spacing but, unlike antennas from the other six categories, may have different numbers of dipoles (9 or 10). Category-A antennas represent the fittest class of antenna designs. The other categories of antennas represent local optima at which the evolutionary systems sometimes got stuck.

Table 4.1: The Seven Classes of Antenna Designs Produced by All Evolutionary Systems.

	Mean Defining Feature of Genome				Mean Fitness Scores		
Type	Number	Length	Height	Spacing	Overall	Directivity	VSWR
A	5	0.4635	1.7094	0.6957	310.03	316.35	1.32
B	6	0.5047	1.7094	0.5689	308.35	316.89	2.54
C	7	0.5222	1.7130	0.4850	307.39	317.39	3.00
D	7	1.2827	1.7394	0.4740	307.29	346.89	16.30
E	8	1.2588	1.7342	0.4269	307.37	350.66	17.65
F	6	1.3267	0.7589	0.5384	306.72	336.12	11.70
G	9.33	1.2636	1.7244	0.3710	306.15	353.42	18.97

There appears to be little difference between the types of antenna designs that were evolved by the control system and causal systems. With the exception of category-G antennas, which are only 3 out of 800 evolved antennas, there are no antenna designs that were produced by the causal

systems that were not evolved, in at least one trial, by the control system. However, there are significant differences in the frequency with which the different systems converged to antenna designs in the seven categories. The distribution of categories is illustrated in Figure 4.9. Of the 200 CONTROL system trials, 123 converged on an optimal category-A antenna design, compared to 139, 177, and 199 of the CAUSAL_{C}, CAUSAL_{M}, and $\text{CAUSAL}_{\text{CM}}$ trials, respectively. The causally-guided system trials converged to category D, E, and F with less frequency than the control system trials. All 200 of the $\text{CAUSAL}_{\text{CM}}$ trials avoided D, E, and F and all but one (category-C) converged to an optimal category-A solution. It is worth noting that the categories that causal systems avoided (D, E, and F) typically have dipole lengths that are longer than optimal and VSWR values that are so high as to be unusable. This relates directly to the user-supplied causal knowledge that is employed by the causal evolutionary systems in these simulations, supporting the hypothesis that the causal relations are contributing effectively to the design process.

To explore the issue of computational cost per generation, 5 trials of the CONTROL and $\text{CAUSAL}_{\text{CM}}$ systems were executed for 5000 generations and the CPU time required for each trial was measured. Surprisingly, it was found that despite the expected increased costs associated with the causally-guided systems, on average the CONTROL system required more than 7 times as much CPU time as the $\text{CAUSAL}_{\text{CM}}$ system (5893 seconds and 793 seconds, respectively). Further investigation revealed that this dif-

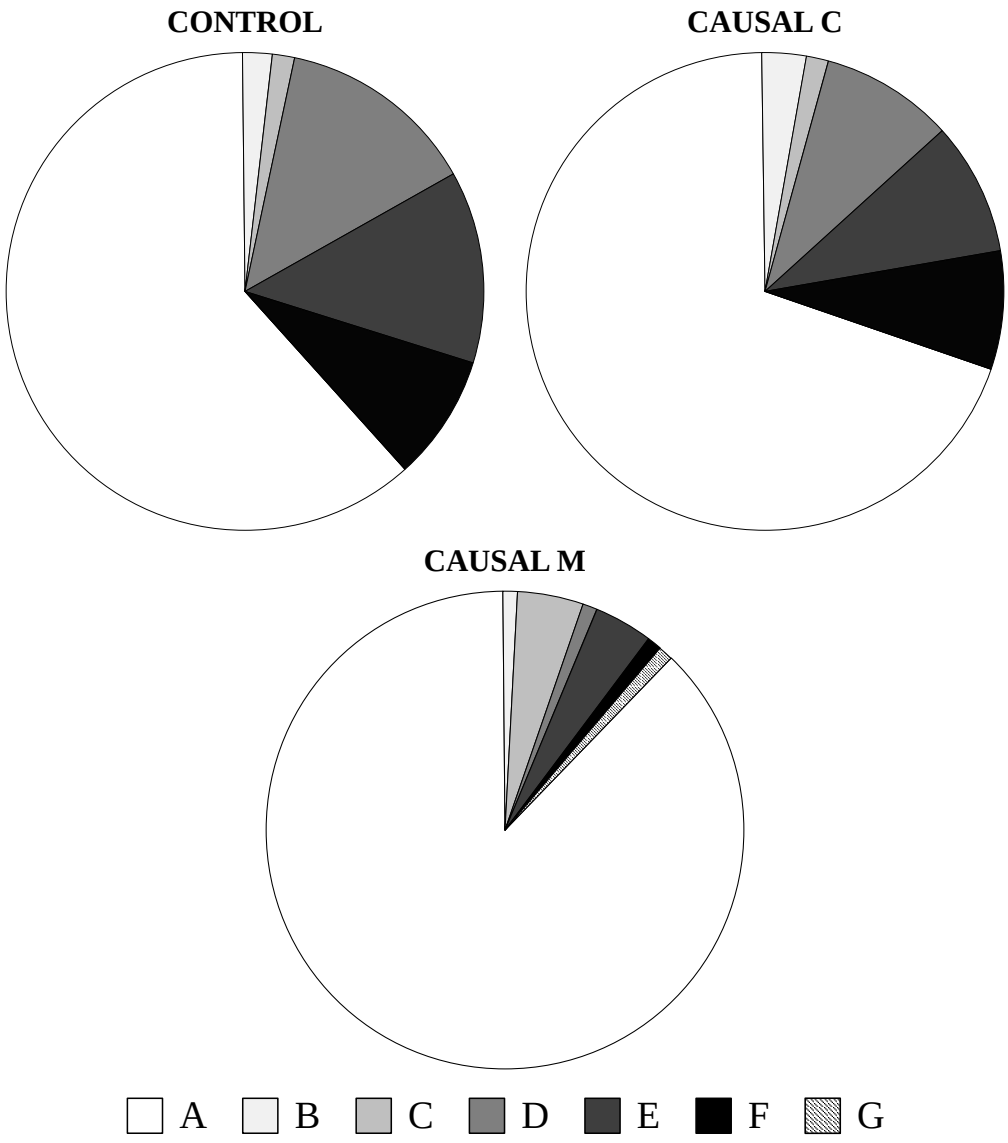

Figure 4.9: Distribution of the categories of antenna designs to which each system's trials converge. All but one CAUSAL$_{CM}$ trials (not shown) converged to category-A designs.

ference is largely due to differences in the types of antenna designs that are explored by the two systems. Specifically, the CONTROL system tends to explore antenna designs with longer and more numerous dipole lengths than the causally-guided systems. These types of antennas are more computationally expensive to simulate during fitness evaluation than smaller antennas. Thus, the increased computation per generation required by the $\text{CAUSAL}_{\text{CM}}$ system to provide causal guidance is dwarfed by the computational savings of evaluating smaller antennas.

An attempt was made to incorporate an alternative causal relation into the evolutionary system, but this failed to improve performance. Specifically, antenna design experts indicate that there is a strong cause-effect relationship between the height of dipoles and the directivity of an antenna. However, causal relations to that effect did not improve the performance of the system. In an attempt to better understand the cause-effect relationships in this domain and the nature of causal relationships in general, some additional experimentation was performed. The single most highly fit antenna of all trials described above was identified and experimented upon. Four separate experiments were conducted. In each experiment, three of the four antenna genes were held fixed, while the fourth was incrementally adjusted, and changes in antenna performance were observed, yielding insight into the causality in this domain and presumably more generally.

Figures 4.10 through 4.13 illustrate the effects that changing the optimal antenna's number of dipoles, length of dipoles, height of dipoles, and

spacing between dipoles has on that antenna's VSWR and directivity. In each graph, a thick black hash on the horizontal axis indicates the unmodified value of the optimal antenna. Consistent with the causal knowledge employed by our system, it can be seen in Figure 4.11 that the length of dipoles has a large effect on the VSWR of an antenna. Noting that the right vertical axis labels of Figure 4.11 differ from those of the other plots shown here, the range of VSWR values when dipole lengths are varied are seen to be an order of magnitude larger than when number, spacing, or height of dipoles are varied. Figure 4.12 reveals a clear and seemingly cyclical causal relationship between the height of dipoles and the directivity of antennas. It is clear from Figure 4.11 and 4.13 that the length of dipoles and the spacing between dipoles also has an effect on the directivity of an antenna, but it is difficult to characterize these interactions. These results indicate the presence of a number of causal relationships in this application domain, but also illustrate the complexity of these relationships.

4.9 Discussion

While my hypothesis was that introducing causally-guided genetic operators would ultimately make evolutionary systems more effective and efficient, there was no guarantee of this fact a priori. It was entirely possible that just the opposite would be true, i.e., adding causal guidance could produce evolutionary systems that are less effective and less computationally

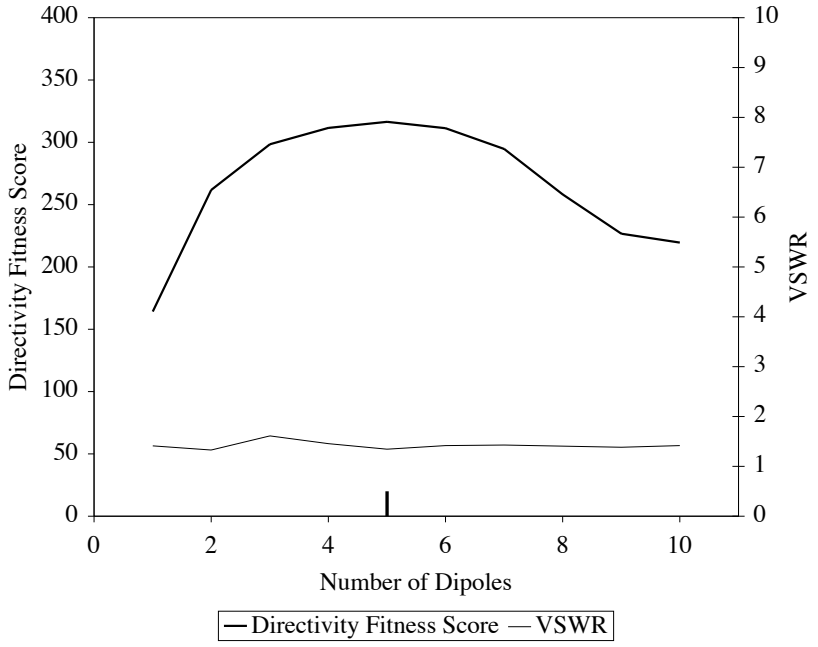

Figure 4.10: Effect that changing the number of dipoles has on FitnessDirectivity score (left axis) and VSWR (right axis) of the optimal antenna.

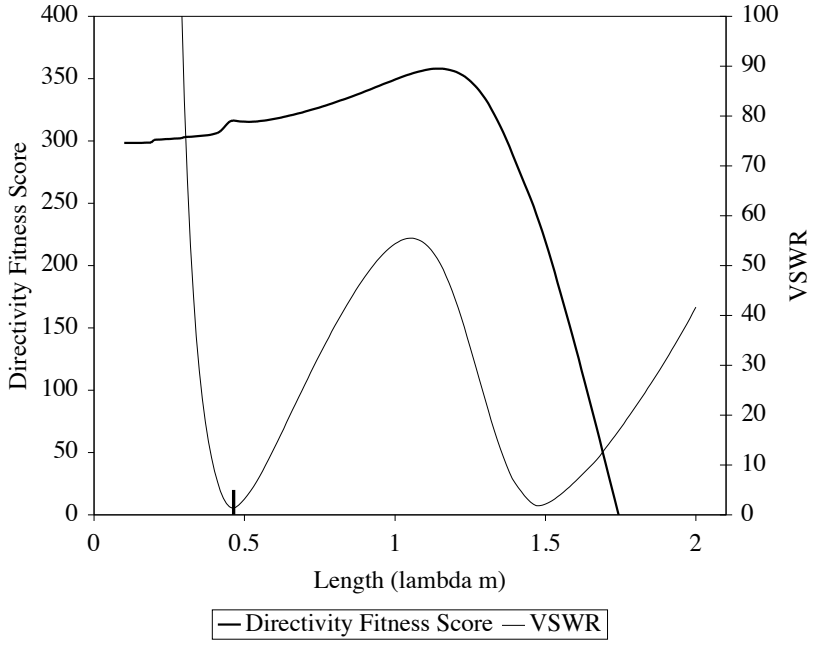

Figure 4.11: Effect that changing the length of dipoles has on FitnessDirectivity score (left axis) and VSWR (right axis) of the optimal antenna. The range of VSWR values plotted here is an order of magnitude larger than Figures 4.10, 4.12 and 4.13.

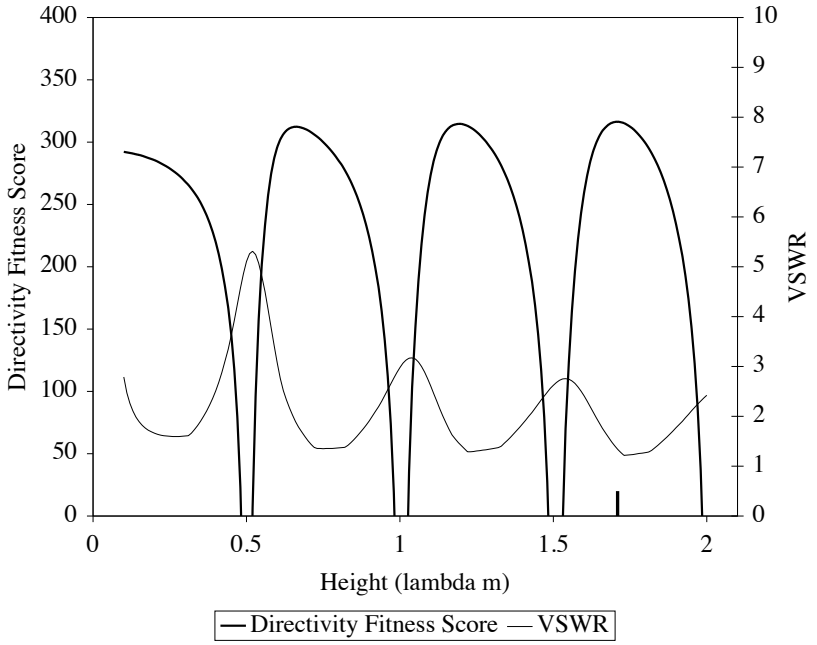

Figure 4.12: Effect that changing the height of dipoles has on FitnessDirectivity score (left axis) and VSWR (right axis) of the optimal antenna.

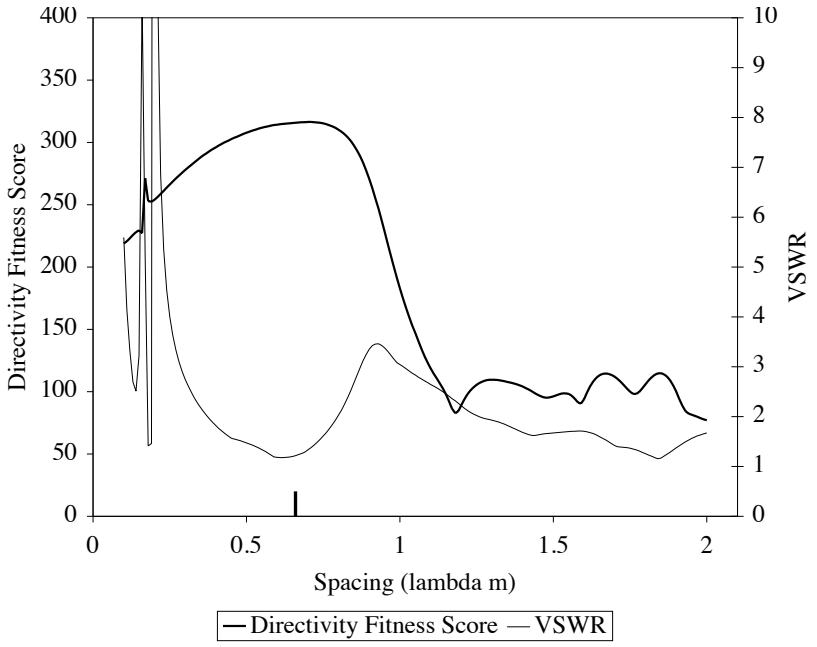

Figure 4.13: Effect that changing the spacing of dipoles has on FitnessDirectivity score (left axis) and VSWR (right axis) of the optimal antenna.

efficient. Instead, I found that causally-guided evolutionary systems could be used successfully to design dipole antenna arrays that meet pre-specified performance criteria. The performance of these systems was compared to carefully matched control systems that do not employ causally-guided genetic operators. At various generations it was found that the causally-guided systems discovered the fittest antennas with significantly greater frequency than the control system. On average, the causally-guided systems also required significantly fewer generations to find antenna designs with various fitness scores. The causally-guided systems found optimal antenna designs much more frequently, largely by avoiding specific sub-optimal designs. Interestingly, these sub-optimal designs were characterized by dipoles that are longer than optimal and have high VSWR values, factors that relate directly to the specific cause-effect relations that the causally-guided system employed. In each result discussed, it was found that the systems using only causal mutation or only causal crossover outperformed the control system, but that the system employing both causal mutation and causal crossover performed even better, indicating that these causally-guided operators were synergistic/complementary.

The additional computational costs required by the causally-guided system to perform causal inference were very minor compared to the overall computational costs of the system. Specifically, it was found that the causally-guided system required 1/7th as much CPU time per generation as the control system. This unexpected result was due to differences in the

characteristics of antennas that were explored by the various systems, and the different computational costs of simulating those antennas. While this is a promising result for causally-guided evolutionary computation in this particular domain, this result is of limited interest as it is domain-specific and may not be relevant to causally-guided evolutionary computation in general. However, it does demonstrate the possibility that in some domains the computational costs of causal inference will be dwarfed by the overall computational costs of the evolutionary systems.

The fact that causally-guided systems were able to solve the dipole antenna array design problem with greater frequency than the control systems demonstrates the positive effect of using causal guidance to bias genetic operations in real-world application domains for which causal knowledge is limited. The tremendous computational savings of using causally-guided systems with a multi-start strategy are particularly convincing, as this is a very practical measure of computational cost. The fact that the causally-guided system avoids local optima that are directly related to the supplied causal relations is especially encouraging, as this suggests that the causal knowledge is successfully steering the search process away from local optima, much as a human designer might do. It is also encouraging that incorporating either causally-guided mutation or causally-guided crossover into the evolutionary process results in improved performance, and that incorporating both results in even greater performance improvements, suggesting that the value of

causal guidance is not critically dependent upon the specific genetic operator being used.

In an effort to explore the causal relationships in this antenna design domain and to better understand causality in general, additional experiments were conducted in which design aspects of the fittest antenna were systematically varied and changes in performance were measured. These experiments validated our belief in a causal relationship between dipole length and antenna VSWR. Furthermore, the influence of dipole length on VSWR was found to be an order of magnitude stronger than the influence of any other design aspect on VSWR (at least in the vicinity of an optimum). It is clear from these experiments that there are other causal relationships in the domain, but that such causality can be quite complex. For example, there appears to be a cyclical relationship between dipole height and antenna directivity. This is because some of the energy that radiates from an antenna is reflected off of the ground plane and passes back over the antenna. There are certain dipole heights at which these reflected waves react destructively with energy radiating directly from the antenna, creating nulls in the antenna's directivity. The heights at which this happens are directly related to the wavelength at which the antenna is operated. Unlike the effect of dipole length on VSWR, it appears that the effects of all design aspects on directivity are of similar magnitude. These results help to explain why preliminary efforts to include a causal relationship between dipole height and antenna directivity into the evolutionary system were unsuccessful, while the causal

relationship between dipole length and VSWR was successfully incorporated. They suggest that with our current approach very strong and straightforward relationships may be incorporated, while more complex relationships involving multiple design aspects may be problematic.

4.10 Appendix

The methods described in Section 4.6 for influencing mutation were developed based on intuition, and as such are not grounded in any formal probabilistic reasoning. However, in this Appendix, I show that these heuristic calculations are consistent with a more principled approach for assessing the relative likelihood of disorders, such as one based on a naive Bayesian classifier.

This approach can be represented as a very simple Bayesian network as pictured in Figure 4.14, in which a single symptom node, VSWR, is dependent upon a single disorder node, D. The disorder node can take a value of either d(N), d(L), d(S), or d(H), which represent the genotypic disorders of suboptimal number of dipoles, length of dipoles, spacing of dipoles, or height of dipoles, respectively. In naive Bayesian classifiers such as this one, there is an implicit assumption that the possible values of the disorder node are mutually exclusive and exhaustive. In this application problem, this is an incorrect assumption as it is clearly possible for multiple disorders to be present (e.g., the number of dipoles and the length of dipoles in a design can

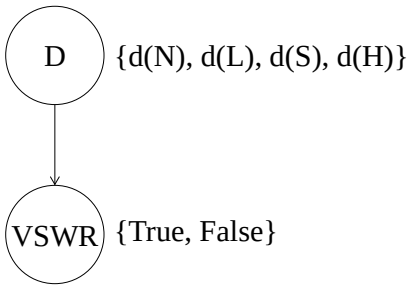

Figure 4.14: A simple Bayesian network representation of a naive Bayesian classifier. The node labeled D represents different mutually exclusive and exhaustive genotypic disorders and the node labeled VSWR represents the presence or absence of the single phenotypic symptom in this application problem.

both be flawed). However, despite their naive assumptions, naive Bayesian classifiers have been shown to be effective in many real-world applications. The symptom node can have a value of true or false, indicating the presence or absence of high VSWR. Given the state of the symptom node (i.e., that high VSWR is present or that it is absent) the posterior probability of the disorder node may be calculated by applying Bayes theorem. For example:

$$\text{P(D=d(L) | VSWR=true)} = \frac{\text{P(VSWR=true | D=d(L)) * P(D=d(L))}}{\text{P(VSWR=true)}}$$

In order to perform this calculation, the prior probability distribution of the disorder node and the conditional probability distribution of the symptom node must be specified. Absent any clear reason to set the prior probability of the disorder node to some particular values, we assume that each possible value is equally probable. Given this assumption, the actual value of this probability does not matter when calculating the posterior probability of each disorder. A variety of possible values for the conditional probability of

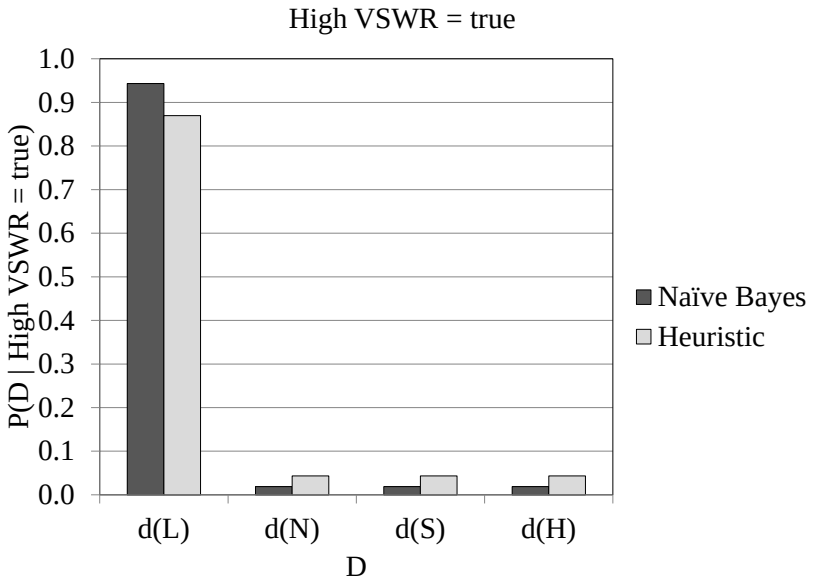

Figure 4.15: The posterior probability of each disorder given that high VSWR is present, calculated through a naive Bayesian classifier and by heuristic methods described in Section 4.6.

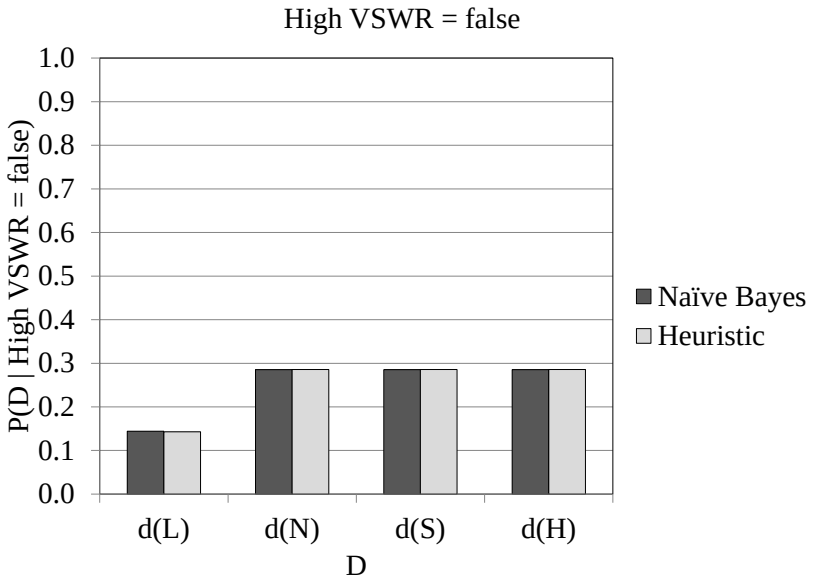

Figure 4.16: The posterior probability of each disorder given that high VSWR is absent, calculated through a naive Bayesian classifier and by heuristic methods described in Section 4.6.

VSWR were evaluated through a trial-and-error approach, and it was found that the following values yield a naive Bayesian classifier that behaves very similarly to the heuristic methods described in this chapter. Specifically, the relative values of the posterior probability of each disorder as produced by the naive Bayesian classifier closely match the relative likelihood with which each disorder is targeted for mutation (via a large mutation operator being applied to the associated gene) in the heuristic causally-guided mutation operator described in Section 4.6.

P(VSWR=true | D=d(L)) = 0.50

P(VSWR=true | D=d(N)) = 0.01

P(VSWR=true | D=d(S)) = 0.01

P(VSWR=true | D=d(S)) = 0.01

Interestingly, the conditional probabilities specified above are consistent with the causal relations described in this chapter, which suggest that P(VSWR=true | D=d(L)) should be much higher than P(VSWR=true | D=d) for all d $\in$ {d(N), d(S), d(H)}. The relative likelihood of each disorder, as calculated by the heuristic and naive Bayesian classifier approach are illustrated in Figure 4.15 and 4.16.

It should be noted that a full Bayesian (i.e., non-naive) network in which four distinct nodes are used to represent the disorders (and accordingly the disorders are not assumed to be mutually exclusive) can also be used to produce posterior probability values that are arbitrarily similar to

the heuristic methods. However, the intent of this appendix is not to produce arbitrarily similar values to the heuristic methods, but rather to demonstrate that the methods used in this chapter, while based on heuristics, are compatible with a more rigorous probabilistic approach.

Chapter 5

Causally-Guided Mutation for Design Construction based on Mechanistic Relations

5.1 Introduction

The goal of this chapter is to extend the ways in which causal-guidance can influence genetic operators, particularly as applied to open-ended or "constructive" design problems. Specifically, while previous chapters explored the use of diagnostic causal relations to influence *where* mutations occur in an individual, in this chapter a second type of causal knowledge in the form of mechanistic causal relations is used to influence *how* each mutation is done. The chapter begins with a more detailed presentation of the motivation for this extension and its relationship to "constructive design." Next, the form of mechanistic causal relations is presented, followed by a general (i.e., application independent) framework for using these relations during causally-guided mutation. To evaluate these ideas, evolutionary systems that use diagnostic relations to influence where mutations are applied and use mechanistic relations to influence how mutations are done are applied to the task of designing synthetic social networks that match predefined

characteristics of real-world social networks. The performance of these evolutionary systems are compared to carefully matched control systems that do not employ causal guidance. Finally, the results of these experiments and their implications for causally-guided evolution are discussed.

5.2 Mechanistic Causal Relations and Design Construction

There are two further extensions made in this chapter. First, in Chapters 3 and 4, diagnostic causal relations are used to bias mutation operators such that those parts of the individual with higher relative likelihoods of being flawed are made more likely to be mutated. In this manner, causal guidance is used to influence *where* mutation is applied to an individual. In this chapter, a second type of causal knowledge in the form of mechanistic causal relations is introduced and used to influence *how* mutation is done. Second, we now consider design construction for the first time. Recall the distinction between between design optimization and design construction discussed in Section 2.5. In design optimization problems it is common to represent solutions to problems as fixed-length lists of parameter values. Mutations to these individuals commonly consist of selecting one parameter value for mutation, and then modifying its value by adding or subtracting some randomly selected value. In contrast, in design construction problems the very structure of the solution is unknown and therefore must be designed by the evolutionary process. For this reason, component-based representations are

often used, and the evolutionary process must arrange and re-arrange these components together to create new solutions.

This distinction also suggests that the benefits of guiding how mutation occurs may be particularly beneficial when applied to constructive design problems. In these types of problems the possible mutations that can be applied to an individual design are often quite numerous and qualitatively distinct. For example, consider an approach to evolving the design of a bridge, in which each individual bridge design consists of a number of trusses arranged together. Any single mutation of this design can involve adding, removing, re-orienting, or re-sizing a truss. Each of these changes also requires additional specifications, such as the new start and end points of a re-oriented or newly added truss. In any sizable bridge design, this is a non-trivial decision with numerous distinct possibilities. In contrast, while there are arbitrarily many ways in which to modify a numeric parameter value in a design optimization problem, it is difficult to view these much simpler changes as being qualitatively different. In constructive design, the qualitative difference between possible mutations provides an opportunity to reason about which mutation is best to apply, and the large number of possible mutations means that any guidance to help select the best mutations may be very beneficial.

5.2.1 General Form of Mechanistic Causal Relations

In this work, as in Chapters 3 and 4, diagnostic causal relations that describe the cause-effect relationship between genotypic disorders and phenotypic symptoms are used. The term "phenotype" is used broadly here, as it is sometimes used in biology, to include not only the structure/form that the genotype develops into, but also the behavioral/performance characteristics of that form. As described below, the experimental study presented in this chapter involves an evolutionary approach to designing networks in which a direct representation is used, i.e., there is no distinction between the genotype and the structure/form that the genotype develops into. Instead, the genotype of the individual is the network itself. In order to avoid confusion regarding the terms genotype and phenotype in this context, I have reformulated diagnostic causal relations with alternate but equivalent terminology, as follows:

Design Flaw $\rightarrow$ Performance Symptom

As in Chapters 3 and 4, the left hand of the expression refers to some part of the individual design that is sub-optimal, while the right side refers to some performance or fitness problem that the design flaw can cause. For example, in the bridge design problem described above in which an evolutionary process arranges and re-arranges trusses in order to construct a bridge, a design flaw of a particular truss being too thin may cause the performance symptom of poor structural integrity.

In addition to diagnostic causal relations, a second type of causal relation is introduced here. *Mechanistic causal relations* describe the cause-effect relationship between the application of particular mutation types to particular design components, and the resulting mitigation of performance symptoms:

Mutation Type x Design Component → Mitigation of Symptoms

The arrow here is not logical implication, but causality. A design component is any type of "building block" that the evolutionary process arranges to construct a solution. For example, in the bridge design example problem described above a design component may be a particular type of truss. A mutation type is any particular type of change that may be made to or "with" a design component, e.g., a particular truss may be added, removed, re-sized, or re-oriented so that its start and end connection points are changed. The mitigated symptoms are the expected effect of applying the mutation to the design component, e.g., removing a large truss could result in the mitigation of the performance symptom of excessive bridge weight.

5.2.2 Causally-Guided Mutation Using Mechanistic Relations

In Chapters 3 and 4, diagnostic causal relations are used to influence where mutation is applied to an individual such that those parts of the individual that are more likely to be flawed are made more likely to be mutated.

For convenience, and to distinguish it from the new form of causal guidance introduced in this chapter, we refer to this diagnostic relation-based guidance as *"where" guidance*, i.e., it influences where mutation is applied. In this chapter, a second type of causal guidance is introduced that uses mechanistic causal relations to influence how mutation is done. *"How" guidance uses mechanistic causal relations to influence how mutation is done, such that mutation types that mitigate observed performance symptoms are made more likely to be applied.* Note that "where" guidance and "how" guidance are not mutually exclusive and can be used together in a single causally-guided mutation operator.

As described in Figure 3.1, causally-guided mutation occurs in three steps. In the "Assess Symptoms" step, performance characteristics of individuals are examined and performance symptoms are assessed. In the "Diagnose Flaws" step, diagnostic causal relations are used to assess the relatively likelihood of various design flaws. Both "where" and "how" guidance are applied in the "Prescribe Treatment" step to influence the execution of the mutation operation. As seen in Chapters 3 and 4, "where" guidance is used to direct mutation toward those parts of the individual that have higher relative likelihoods of being flawed. "How" guidance is used to influence mutation such that those mutation types that mitigate observed performance symptoms are made more likely to be selected.

In the work presented in this chapter, an evolutionary system is used to design synthetic social networks that match predefined characteristics of

real-world social networks. While past studies have examined evolving network structures and developing crossover operators to work on these network structures, there is currently no standard way of performing crossover that has been shown to be broadly effective. Accordingly, in this chapter crossover operators are not used, and instead we focus on using mechanistic causal relations to influence the mutation operator.

5.3 Synthetic Social Network Design

To evaluate the effectiveness of incorporating "how" guidance based on mechanistic causal relations, evolutionary computation systems that employ "where" and "how" guidance were applied to the task of designing synthetic social networks. A generational genetic algorithm augmented with "where" and "how"-guided mutation operators was developed, and its performance in solving this problem was compared to a carefully matched genetic algorithm that uses no causal guidance but is otherwise equivalent. Additional experimentation compares the effects of "how" and "where" guidance when used independently. With the exception of the causally-guided genetic operators, all aspects of the evolutionary systems (described in more detail below) are conventional and widely used (De Jong, 2006).

There are a number of goals for these experiments. The first is to determine if the increased effectiveness and efficiency of "where"-guided mutation that was observed in previous studies is repeated in this new application

problem. Second, these experiments evaluate whether using "how"-guided mutation yields benefits, has no significant effects, or misleads the evolutionary process toward local minima. Third, the combined effects of "where" and "how" guidance for mutation are examined in order to determine if these mechanisms for causal guidance are synergistic or if they interfere with each other.

5.3.1 Motivation for Synthetic Social Network Data

Driven in part by the recent rise of social media as a means of widespread communication, interest in social network analysis and research has increased dramatically in recent years. This has resulted in increasing demands for social network and social media data from researchers engaged in these areas. While large amounts of social network data are being collected by various military, government, and commercial entities, this data is not freely shared due to proprietary and privacy concerns. This limited availability of social network and media data for researchers is a bottleneck for the field of social network research.

For this reason, there has been much recent interest in developing methods for creating synthetic social network data that has high fidelity with respect to real-world data. Such methods would allow researchers to take a small set of real-world data and generate synthetic social network data that is very similar. Research could be conducted on this synthetic data, and the

data itself could be shared freely. Furthermore, the ability to create synthetic data with pre-specified characteristics would allow researchers to systematically vary characteristics of the data, facilitating the evaluation of analysis algorithms on a range of social network data sets in a controlled and rigorous fashion.

5.3.2 Related Work

One approach to generating synthetic data is through the use of random graph models. Numerous past studies have explored the use of graph models to explain various commonly observed characteristics of real world data (Erdos and Renyi, 1959; Watts and Strogatz, 1998; Barabasi and Albert, 1999; Newman, 2009; Reittu and Norros, 2012; Jin et al., 2001). For example, the Watts-Stogartz random graph model (commonly referred to as the "small-world" model) was a ground-breaking study that illustrated how the tendency of individuals to connect to others who are like them combined with the existence of a small number of weak ties connecting distant parts of a network can result in networks that are both highly clustered and have low average path length; two characteristics that may be incorrectly perceived as being contradictory. Other prominent graph models (Erdos and Renyi, 1959; Barabasi and Albert, 1999) have been essential to explaining how the interplay of simple local interaction can lead to the emergence of global net-

work phenomena such as the existence of very large connected components and power law distributions of connections.

In addition to explaining the existence of observed phenomena, these graph models can also be used to produce synthetic network data. However, this type of approach has some important limitations. First, many graph models have been designed to explain aggregate characteristics of networks (Barrett et al., 2009) but do not produce synthetic networks that match on node and connection-level characteristics. For example, the Watts-Stogartz model is capable of producing networks with desired average clustering and path length values, but these networks tend to have degree distributions that are unrealistic. Furthermore, many real-world networks cannot be easily matched to any single graph model. For example, some terrorist networks exhibit properties that are consistent with both small-world and scale-free graph models (Rothenberg, 2002). Indeed, some past studies evaluating the effectiveness of prominent graph models to produce high fidelity synthetic data have found poor performance in most instances (Sala et al., 2010). This is in part because graph models have been designed to explain general phenomena that are observed in a wide variety of networks, and have not been tailored to specific domains or data sets. Instead, what is needed is a way to automatically produce synthetic data that matches the characteristics of target real-world data sets.

A second approach to generating synthetic networks is through simulation. In these types of approaches, real-world data sources are combined with

behavioral and social theories to synthesize networks. For example, (Barrett et al., 2009) use a variety of real-world information sources to construct an entire synthetic urban population of individuals and households. Models of expected activities for each household are then constructed and used to simulate the actions of these synthetic persons, resulting in the construction of a social contact network. While this very ambitious approach has shown some promise and continues to be an active area of research, one important limitation is the availability of data. In the absence of extensive information about the demographics and household activities of real persons, simulating their behavior is not possible. In the work presented here, the task is to generate synthetic network data that matches the characteristics of a real-world network, with only the real-world network itself as a guide.

Many past studies have explored using evolutionary computation methods to solve a variety of problems related to social network analysis (Firat et al., 2007; Stonedahl et al., 2010), but only one has explored using evolutionary computation to create synthetic social network data (Bailey et al., 2012). In that study, a genetic programming approach is used to create a graph model that can be used to generate a synthetic network that matches the characteristics of a target network. In that regard, the problem addressed by (Bailey et al., 2012) is very similar to the one that is used in this study to evaluate causally-guided evolution. However, there are a few important differences. First, in the work presented here the social network itself is directly evolved, while in Bailey's work a graph model is evolved. While there

are potential pros and cons to each approach (an evolved graph model may scale better whereas a directly evolved network may have higher fidelity to real-world data), the main importance of this distinction is that it is not possible to directly compare the techniques presented here with Bailey's study. Second, in this study I evaluate the performance of the evolutionary system in terms of its ability to match the characteristics of real-world data sets, while in Bailey's study the performance is evaluated in terms of its ability to match the characteristics of other synthetic data sets that were created by well-known graph models. Third, there are some differences in terms of the graph metrics that are used in each study, with Bailey's work focusing on aggregate clustering, path length, and distribution of node degree, while my work focuses on the distribution of clustering coefficients. As noted in (Bailey et al., 2012), the selection of characteristics is somewhat subjective. Last, and most importantly in the context of the research interests of this dissertation, Bailey's approach employs a conventional genetic programming approach that does not involve any causal knowledge or guidance.

5.3.3 The Importance of Clustering

Synthetic social networks are only useful if they exhibit a high degree of fidelity to real world networks. However, the set of specific characteristics and criteria that constitute "high-fidelity" is not objectively clear. As others have noted, if we view any real-world social networks as being the result of some

process then our goal is to create synthetic data that could have plausibly been created by that same process (Bailey et al., 2012). The central challenge here is that the process that created the real-world network is not known a priori and so the fidelity of synthetic social network data to this process cannot be directly evaluated. Instead, we must select some set of graph metrics by which we evaluate the "similarity" of the synthetic data to the real-world target data. The problem here is not one of graph isomorphism, because the goal is not to produce the same graph but rather a graph with "similar properties as the given graph" (Bailey et al., 2012).

In the interest of simplicity for this first-ever experimental evaluation of "how" guidance based on mechanistic relations, the task considered here is to design a synthetic social network that matches target real-world networks in one very important aspect: the distribution of local clustering coefficients in the network. The term *triadic closure* refers to the phenomena that if person B and person C have a common friend in person A, then there is an increased likelihood that B and C will be friends (Easley and Kleinberg, 2010). It has been theorized that triadic closure occurs in social networks for three major reasons: 1) there is an increased opportunity for B and C to meet since they share a common friend; 2) there is a basis for trust between B and C given that they both trust and are trusted by A; and 3) there is incentive for B and C to become friends as it alleviates stress for A that may be present if B and C are not friends (Easley and Kleinberg, 2010). Triadic closure is observed in most real-world social networks, in which the prevalence of tightly knit

groups of persons is far more prevalent than could be explained by random phenomena. Furthermore, the number of closed triads surrounding nodes has been shown to be a salient characteristic that has greater implications. For example, previous studies have found that teenage girls with small numbers of triadic closures among their friends tend to be more likely to contemplate suicide (Bearman and Moody, 2004).

The importance of triadic closure in social networks has led to the formulation of measures to quantify it, one of which is the *clustering coefficient*. The clustering coefficient of a node A is defined as the probability that two randomly selected neighbors of A are neighbors with each other. In other words, it is the fraction of pairs of A's neighbors that are connected to each other by edges (Easley and Kleinberg, 2010). For example, consider an undirected graph in which a node has four neighbors. There are six possible connections between these neighbors. If five of those six connections actually exist then the clustering coefficient of the node is $\frac{5}{6}$.

Note that the overall clustering of a network can be quantified in a single aggregate numeric value by calculating the global clustering coefficient or the network averaged clustering coefficient. However, it is possible to have two networks that are similar in terms of these aggregate clustering metrics but have dramatically different local clustering coefficients of nodes. The goal of the problem considered here is not to match aggregate clustering of a network, but rather to evolve synthetic social networks whose distribution of

local clustering coefficients closely matches the distribution of local clustering coefficients in a target real-world network.

5.4 Fitness Measure and Genetic Representation

A fitness evaluator was constructed to quantify the difference between the distributions of clustering coefficients in an evolved synthetic network and in a target network (i.e., the real-world network). Fitness evaluation takes place in three steps. First, the clustering coefficients of each node in the synthetic network and each node in the target network are calculated. Second, a histogram is created for the clustering coefficients observed in the synthetic social network and for those observed in the real-world network. Third, the histograms for the synthetic and target networks are compared to each other and the difference between them is quantified. This difference is used as the inverse fitness (cost or error) of the synthetic individual, which the evolutionary process seeks to minimize. Each of these steps is discussed in greater detail below.

First, the clustering of each node in the target and synthetic network is calculated in a straightforward process. For clarity, we differentiate between the clustering of a node and the clustering coefficient of a node. The clustering of a node i, represented by the term C_i, is defined by the number of nodes in i's open neighborhood and the number of connections between those nodes. The notation $\left\{ {e \atop k} \right\}$ is used to represent a clustering in which a

node has k neighbors and e connections between those neighbors. The *open neighborhood* of a node i is defined as the set of nodes that are neighbors of i. The open neighborhood of each node i is examined and the following calculations are performed, where N is the set of nodes in the open neighborhood of i and E is the set of edges between those nodes. Not that for each node in the target network, these calculations only need to be performed once, at the very beginning of the evolutionary process.

$$C_i = \left\{{e \atop k}\right\}, \text{where } e = |\{e_{jl} : v_j, v_l \in N, e_{jl} \in E\}|, \text{and } k = |N|$$

$$\text{Coeff}(C_i) = \frac{e}{k * (k-1)/2}$$

All target and synthetic networks in this study are treated as being undirected. This decision was made because creating synthetic networks that match the distribution of clustering coefficients of real-world networks is much more difficult in the undirected case than the directed case.

Second, histograms of the clustering coefficients are constructed for both the target and the synthetic network. Each histogram has 12 bins, with the first two bins representing isolates and pendants, and the remaining ten representing clustering coefficients with value $[0.0, 0.1)$, $[0.1, 0.2)$, $[0.2, 0.3)$, $[0.3, 0.4)$, $[0.8, 0.9)$, $[0.9, 1.0]$. *Isolates* are nodes that are completely disconnected from the rest of the network, while *pendants* are nodes that are connected to the network by only one tie. Because isolates do not have any neighbors and pendants have only one neighbor, there is no opportunity for

clustering to take place and therefore their clustering coefficients are not defined. Pendants cannot be ignored (e.g., by removing them from the target network) as their presence defines the clustering coefficient of the nodes to which they are connected. As such they are an important and non-trivial part of the overall clustering characteristic of a target network, which our synthetic data should ideally match.

In constrast, isolates are removed from the target network a priori. This decision was made because isolates, unlike pendants, are not very interesting or relevant in this task. For example, if 10% of a target real-world network nodes are isolates, this can be easily replicated by simply adding an appropriate number of isolates to the synthetic network after it has been generated. The presence of these disconnected nodes does not need to be addressed as part of the synthesis process. In this context, the key task examined here is to design synthetic social networks that match the clustering characteristics of the *connected components* of a target real-world network. Figure 5.1 shows a graph representation of a social network derived from the study of a karate club (Zachary, 1977), while Figure 5.2 shows a histogram plot of the clustering coefficients of nodes in that social network.

Third, the difference between the target and synthetic histograms is quantified. To accomplish this, a root mean square error (RMSE) calculation is used as follows, where T is the target network, S is the synthetic network, $error(T, S)$ is the quantified difference between the histograms for T and S,

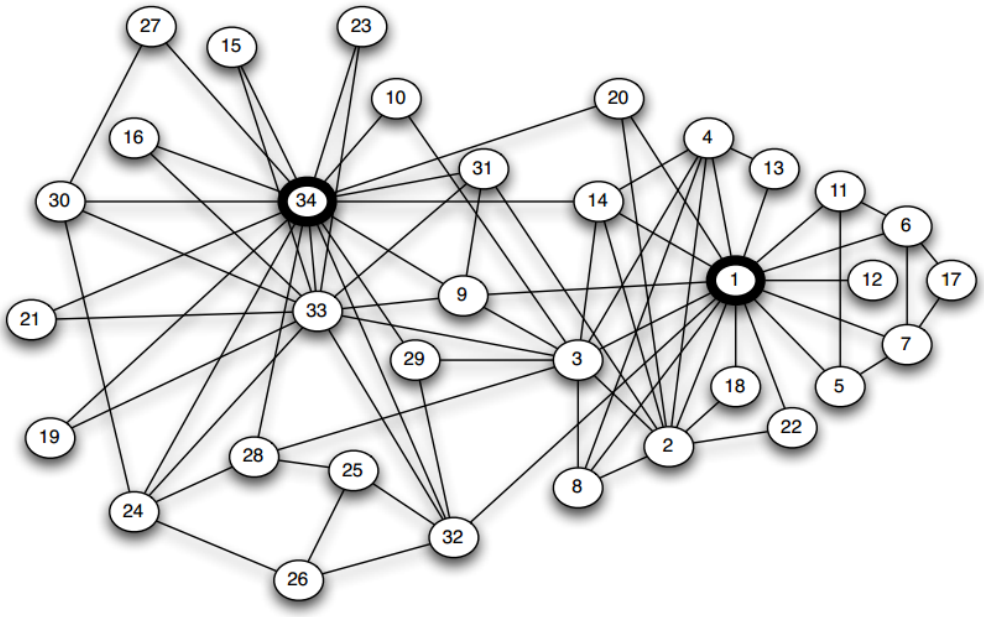

Figure 5.1: Social network from the study of a karate club. The highlighted nodes are two leaders within the group: the club president (node 34) and the instructor (node 1).

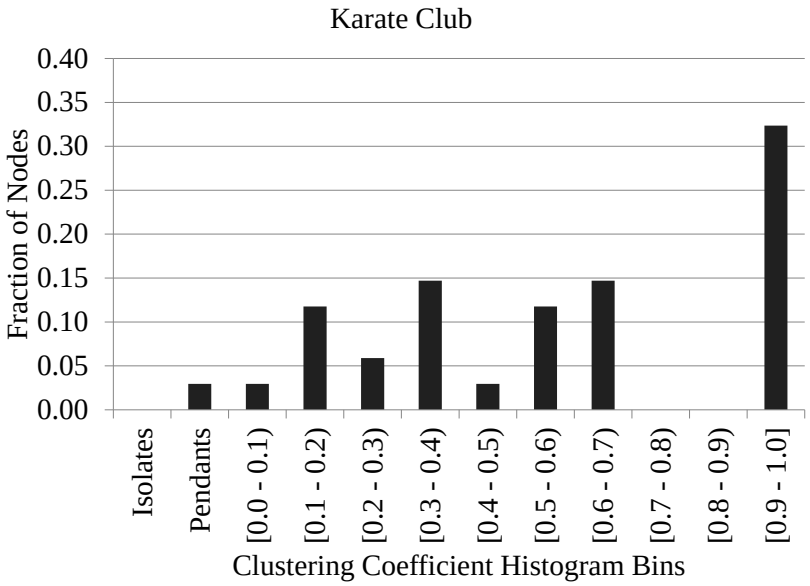

Figure 5.2: Histogram of the clustering coefficient of all 34 nodes in the karate social network.

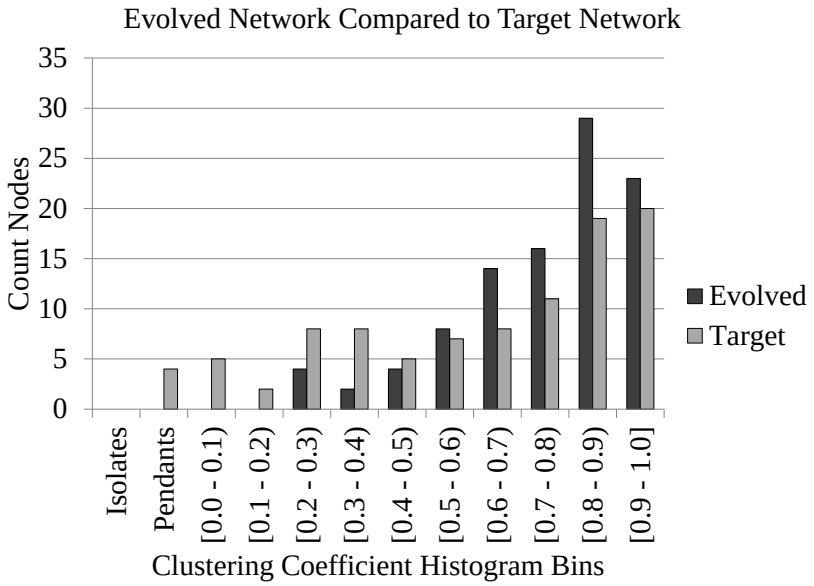

Figure 5.3: Histogram of clustering coefficients observed in two different social networks. The RMSE of these two distributions is 4.73.

T_i is the i^{th} value in T's histogram, and S_i is the i^{th} value in S's histogram. Figure 5.3 shows histogram data for two social networks. The RMSE measure

$$error(T, S) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T_i - S_i)^2}$$

of the difference between these two histograms is 4.73. In the RMSE calculation, a single large deviation results in a larger error measure than a number of small deviations that sum to the same magnitude as that large deviation. That is to say, RMSE has the effect of magnifying large deviations more than small deviations. In the context of social network synthesis this is a desirable property as exact matches of histogram data are not as important as avoiding large mismatches.

A direct genetic representation is used in which there is no distinction between the genotypic and the phenotype of the individual. Instead, the

genetic representation is the network itself and consists of nodes and connections. Accordingly, all genetic operators modify the individual by making changes directly to the network. Each individual network has a fixed number of nodes, defined a priori, which does not change throughout the evolutionary process. While it would be possible to allow mutation to add and remove nodes from the network, for simplicity in this study these types of mutations were not used. Instead, in all of the evolutionary systems in this study (described below) four distinct types of mutations are defined which either add or remove connections: 1) adding a connection from node n to another node; 2) deleting a connection from n to one of its neighbors; 3) adding a connection between two of n's neighbors thereby introducing an instance of triadic closure; and 4) deleting a connection between two of n's neighbors thereby removing an instance of triadic closure. These mutation are referred to using the notation ADD_E, DEL_E, ADD_T, DEL_T, respectively. The subscripts T and E are used to indicate "triad" and "edge."

5.5 Causal Knowledge

Two types of causal relations are used in this study. First, diagnostic relations are used to relate design flaws to performance symptoms, as in Chapter 3 and 4. For every node in a network, there is one possible design flaw:

flaw(node n): the local neighborhood of node n is sub-optimal,

i.e., the presence or absence of connections differ from what would be observed in an optimal design

Thus, in a network with N nodes, there are N possible design flaws. Additionally, for each histogram bin b there are two types performance symptoms:

excess(bin b) : there are too many nodes in histogram bin b

insufficient(bin b) : there are too few nodes in histogram bin b

Note that in any individual design, there may be multiple *instances* of any one performance symptom. For example, an individual that has six more nodes in the fifth histogram bin than is desired is viewed as exhibiting six instances of the symptom excess(5). For every node n and histogram b, there is a causal relationship between flaw(n) and excess(b) whenever the clustering coefficient of n is such that it is placed in histogram bin b. Put differently, if the clustering coefficient of node n is sub-optimal and its clustering coefficient is such that it is placed in histogram bin b, this can cause there to be too many nodes in histogram bin b. For all n and b:

flaw(n) $\rightarrow$ excess(b),iff Bin(Coeff(C_n)) = b

where $\rightarrow$ is used to represent causality and Bin(X) is the histogram bin in which clustering coefficient X belongs.

Note that this causal knowledge is intuitive and yet incomplete. It is obvious that a node with a sub-optimal local neighborhood may have a

sub-optimal clustering coefficient, and that this can result in the node being placed into a histogram bin that is different than where it would be placed in an optimal design. While this is clearly possible, it is not necessarily the case in all instances. For example, consider the scenario in which node n is "supposed to" have a clustering coefficient of 0.33. It is possible for the local neighborhood of n to be sup-optimal in the sense that a misplaced connection between n's neighbors results in one of n's neighbors having a sub-optimal clustering coefficient, and yet the clustering coefficient of n remains 0.33. In this scenario, the flaw would not necessarily cause the count of nodes in the histogram containing 0.33 to be higher than it would be in an optimal network. This highlights one of the central questions of the research: can cause-effect relationships that are only approximately correct still productively guide the evolutionary process?

Note that no diagnostic causal relations are specified between design flaws and insufficient() performance symptoms. It could be argued that flaw(n) should have a causal relationship with insufficient(b), for every b such that Bin(Coeff(C_n)) != b, based on the notion that the flawed local neighborhood of node n can cause it to be placed in a histogram bin that is not ideal, thus depriving the ideal histogram bin of a needed entry and resulting in the insufficient() symptom. However, I opted to not include such relations in this study because I believe they will be ineffective. Specifically, these relations are so non-specific (each flaw would be a potential cause of almost every insufficient symptom) and weak that I suspect they would not

be useful in guiding genetic operators. Instead, and in keeping with the description of causally-guided evolution in previous chapters, I opted to only include the strongest causal relations: those between the flaw()'s and excess() symptoms.

Mechanistic causal relations, the second type of relation used, describe the cause-effect relationship between the application of a particular type of mutation to particular design components, and the performance symptoms that are expected to be mitigated as a result. In this study, the design components are nodes with particular clustering characteristics. The notation n(C) is used to represent a node with clustering of C. Alternatively, the notation $\mathrm{n}\left\{{e \atop k}\right\}$ is used to represent a node with clustering $\left\{{e \atop k}\right\}$. The mutation types are $\mathrm{ADD_E}$, $\mathrm{DEL_E}$, $\mathrm{ADD_T}$, $\mathrm{DEL_T}$, as described previously. In order to describe the mechanistic causal relations used in this study, we must first define a number of supporting terms.

The term ExpMtoN(m,n(C)) is used to represent the expected effect (i.e., how the design component will be changed) of applying mutation m to design component n(C), and is central to the mechanistic causal relations used in this study. It is calculated as follows:

$$\mathrm{ExpMtoN}(\mathrm{ADD_E},\ \mathrm{n}\left\{{e \atop k}\right\}) = \mathrm{n}\left\{{e \atop k+1}\right\}$$

$$\mathrm{ExpMtoN}(\mathrm{DEL_E},\ \mathrm{n}\left\{{e \atop k}\right\}) = \mathrm{n}\left\{{\mathrm{round}(e-\frac{2e}{k}) \atop k-1}\right\}$$

$$\mathrm{ExpMtoN}(\mathrm{ADD_T},\ \mathrm{n}\left\{{e \atop k}\right\}) = \mathrm{n}\left\{{\min(e+1, k*(k-1)*\frac{1}{2}) \atop k}\right\}$$

$$\mathrm{ExpMtoN}(\mathrm{DEL_T},\ \mathrm{n}\left\{{e \atop k}\right\}) = \mathrm{n}\left\{{\max(e-1,0) \atop k}\right\}$$

Each of these equations is explained here and examples are provided.

The ADD_E mutation, which adds a connection between a node n and a stranger (i.e., a node that was previously not directly connected to n), has the expected effect of increasing the number of nodes in n's open neighborhood by one while leaving the number of edges unchanged. For example, applying ADD_E to a node with clustering of $\left\{{2 \atop 3}\right\}$ will result in one more neighbor but no additional edges between those neighbors: $\left\{{2 \atop 4}\right\}$.

The DEL_E mutation, which deletes a connection between node n and one of its neighbors is the most complex. Each node in the open neighborhood is involved in an average of $\frac{2e}{k}$ edges. Therefore, removing any node from n's open neighborhood has the expected effect, on average, of decreasing the number of edges in the open neighborhood by $\frac{2e}{k}$ while decreasing the number of nodes by one. For example, removing a random edge from a node with clustering of $\left\{{6 \atop 5}\right\}$ will on average result in clustering of $\left\{{4 \atop 4}\right\}$. Note that rounding is used to ensure that a valid clustering measure is produced (e.g., $\left\{{4 \atop 4}\right\}$ rather than $\left\{{3.6 \atop 4}\right\}$ in the above example).

The ADD_T mutation, which adds a connection between two of node n's neighbors, has the expected effect of increasing the number of edges in n's open neighborhood by one, while leaving the number of nodes unchanged. For example, when ADD_T is applied to a node with clustering of $\left\{{4 \atop 4}\right\}$, the expected clustering measure that will result is $\left\{{5 \atop 4}\right\}$. If there is no edge to be added between two of n's neighbors (i.e., all $k * (k-1) * \frac{1}{2}$ edges already

exist) then the number of edges in n's open neighborhood is not changed by the mutation.

Similarly, the $\mathrm{DEL_T}$ mutation, which removes a connection between two of node n's neighbors, has the expected effect of decreasing the number of edges in n's open neighborhood by one, while leaving the number of nodes unchanged. For example, applying $\mathrm{DEL_T}$ to a node with clustering of $\binom{5}{4}$ will result in clustering of $\binom{4}{4}$. If there is no edge to be removed, then the number of edges in n's open neighborhood remains zero.

Note that the causal knowledge used here is not universally accurate. For example, in the case of the $\mathrm{ADD_E}$ mutation, it is possible to add a connection between node n and a non-neighbor node o where o is already connected to one of n's neighbors. In that instances, the number of edges in n's open neighborhood would also increase. Similarly, the ExpMtoN formula for $\mathrm{DEL_E}$ is based on an average expected outcome and may not be accurate in many or even most specific instances. However, just as in the relationship between design flaws and performance symptoms discussed previously, the causal knowledge described here is approximately correct. It remains to be seen experimentally if this is sufficient to guide the evolutionary process.

The mechanistic causal relations which describe the cause-effect relationship between the application of particular mutation types to nodes with particular clustering characteristics and the performance symptoms that are addressed as a result, are as follows:

Mutation m x Node n(C) $\rightarrow$ Mitigation of insufficient(b)

where b = Bin(Coeff(ExpMtoN(m,n)). For example, these mechanistic relations tell us that the expected effect of applying the ADD_E mutation type to a node with clustering of $\left\{{3 \atop 4}\right\}$ is that an instance of the performance symptom insufficient(5) will be mitigated, if any are present. The details of this calculation are as follows:

ADD_E x n$\left\{{3 \atop 4}\right\}$ $\rightarrow$ Mitigation of insufficient(b), where

b = Bin(Coeff(ExpMtoN(ADD_E, n$\left\{{3 \atop 4}\right\}$)))

b = Bin(Coeff(n$\left\{{3 \atop 5}\right\}$))

b = Bin(0.30)

b = 5

5.6 Causally-Guided Mutation with "Where" and "How" Guidance

As discussed in Section 3.2, causally-guided genetic operations occur in three steps: the performance characteristics of the individuals in question are examined and performance symptoms are identified, inferences are made about the likelihoods of design flaws, and these inferred likelihoods are used to bias the execution of the genetic operator.

5.6.1 Assessing Symptoms

In this chapter, the performance characteristics of each individual social network consist of histogram data of the clustering coefficients of nodes in that individual, which is created during the fitness evaluation of each individual. Once an individual social network is selected for mutation, the histogram data is examined and deviations from the target histogram are used to identify performance symptoms. For each histogram bin b in which $T_b > S_b$, $(T_b - S_b)$ instances of insufficient(b) are identified, where T_b is the number of entries in bin b of the target network histogram, and S_b is the number of entries in bin b of the synthetic network (i.e., evolved individual) network. For each histogram bin b in which $T_b < S_b$, $(S_b - T_b)$ instances of excess(b) are identified. For example, if the histogram data for the target network has 16 entries in the 5th bin, and the histogram data for an evolved synthetic network has 30 entries in the 5th bin, then 14 instances of the performance symptom excess(5) are identified for the evolved individual. In this manner, the severity of the deviation between the target and actual number of entries in histogram is captured by the number of instances of the associated performance symptom (i.e., larger deviations result in more instances of the symptom).

5.6.2 Diagnosing Flaws

Next, the identified performance symptoms are examined and inferences are made about the relative likelihood of design flaws. For each node n in an individual evolved network the relative likelihood of flaw(n), which means that the local neighborhood of node n is sub-optimal, is calculated based on the set of performance symptoms that are present in the individual. To perform this calculation, first observe that according to our causal knowledge, flaw(n) can cause one instance of excess(b) where b = Bin(Coeff(C_n)) and does not cause any other symptoms. Furthermore, all rival explanations for the presence of excess(b) (i.e., flaw(m) where Bin(Coeff(C_m)) = b and n != m) cannot cause any symptoms other than excess(b). Therefore the likelihood of flaw(n) depends only upon the number of instances of the symptom that it can cause and on the number of rival explanations. Specifically, the following equation is used to calculate the likelihood of flaw(n) given the observed symptoms:

$$\text{likelihood}(\text{flaw}(\text{n})) = \frac{|\text{instances of excess}(\text{b})|}{|\{\text{node m}|\text{flaw}(\text{m}) \rightarrow \text{excess}(\text{b})\}|}$$

where b = Bin(Coeff(C_n)). This equation is intuitive when described in plain language using a concrete example. Consider a situation in which an individual evolved network has been evaluated and the resulting histogram has 24 entries in the 4th bin, which is 10 more than in the target network histogram. In this scenario it can be reasoned that 10 of the 24 nodes are

flawed, thereby causing 10 instances of the phenotypic symptom excess(4). Without any additional knowledge regarding which of the 24 are more or less likely to be flawed, it is concluded that the likelihood of each node being flawed is simply $\frac{10}{24}$.

5.6.3 Causal Guidance of Mutation

Finally, the application of the genetic operator is biased through "where" and "how" guidance. In this study, mutation occurs in two distinct steps. First, an individual node in the social network is selected for mutation. "Where" guidance is used to influence which node is selected for mutation such that those nodes with higher likelihoods of being flawed are made more likely to be mutated. Specifically, a single node n is selected from the set of all nodes with probability proportional to each node's likelihood(flaw()) score. To dampen the effects of "where" guidance, a mitigation constant m (between 0.0 and 1.0) is used. Before the selection of a node to modify, a random number is generated between 0.0 and 1.0. If this random number is less than m, then "where" guidance is not used and instead a node is selected from the set of all nodes with equal probability. In this study, a value of m = 0.10 was found to be effective through a small number of trial and error runs.

Once a node has been selected for modification, "how" guidance is used to influence which mutation type is selected such that those mutation

types that address observed phenotypic symptoms are made more likely to be used. Once an individual node has been selected for modification, one of the following four types of mutations is selected and applied to that node: $\mathrm{ADD_E}$, $\mathrm{DEL_E}$, $\mathrm{ADD_T}$, and $\mathrm{DEL_T}$. To that end, a utility function is defined, which calculates the expected utility of applying each mutation type m to a node n:

$$\text{Utility(m,n(C))} = \text{UtilityMitigate(s)}$$

$$\text{UtilityMitigate(s)} = |\text{instances of s}|$$

where s is symptom that is mitigated by applying m to n(C), (i.e., m x n(C) $\rightarrow$ Mitigation of s).

In other words, to calculate the utility of applying a particular mutation type to a particular node, we first determine which performance symptoms such an action is expected to mitigate. The utility is then assessed as being equal to the number of such instances that have in fact been observed in the individual. In this manner, higher utility scores are assigned to mutation type that address phenotypic symptoms of which there are many instances. More concretely, higher utility scores are assigned to those mutation types that are expected to result in a node being placed into a histogram bin b in which there are currently too few entries, thereby addressing the performance symptom insufficient(b). On the other hand, if the expected effect of applying the mutation type is that it will place the node in a histogram bin b for which

there are already enough entries then the utility will be zero as there are no instances of insufficient(b) in such a scenario.

For example, consider the application of the ADD_E mutation type to a node with clustering of $\left\{\frac{2}{5}\right\}$. According to our mechanistic causal relations, the expected effect of applying this mutation type is that the performance symptom insufficient(3) is mitigated. If the network has 10 entries in histogram bin 3 but the target count for that bin is 18, then there are 8 instances of insufficient(3) and the utility of applying ADD_E to the node is assessed to be 8.

Once the utility score of all mutation types have been calculated, a single mutation type is selected with probability proportional to its utility score. As in "where" guidance, a mitigation constant of 0.10 is used to dampen the effects of "how" guidance, resulting in "how" guidance being skipped in 10% of mutation operations. In these instances one of the four mutation types is selected randomly with equal probability. Once a mutation type is selected, it is applied to the individual. For any selected type of mutation there may be numerous specific ways in which the individual may be changed. For example, if the ADD_T mutation type is selected for a node n, there may be several distinct connections that could be added in order to connect two of node n's neighbors. Whenever a mutation is selected to be applied to an individual, exactly one of these distinct changes is selected randomly and with equal probability from the space of all possible changes. In some instances

there may be no way to make a change of the selected type. For example, a DEL_{T} modification could be applied to a node whose neighbors are all disconnected from each other. When this happens, a different mutation type is selected and applied.

The "where" guidance and "how" guidance described above is used to influence where mutation is applied to an individual and how that mutation is done, and mutation operators that employ these forms of guidance serve as the experimental condition in this study. These mutation operators are referred to as CAUSAL-W-H where the letters W and H refer to the "where" and "how," respectively. A control mutation, referred to as CONTROL, was defined in which neither "where" nor "how" guidance is used. The CONTROL mutation follows the same two step process as the CAUSAL-W-H mutation: first a node is selected for mutation, second a specific type of mutation is selected and applied. However, unlike the CAUSAL-W-H operator, in the CONTROL operator neither of these steps is influenced by causal guidance. Instead, the node that is selected for mutation is simply selected from the set of all nodes with uniform probability. Similarly, the specific mutation type to apply to the selected node is selected from the set of all mutation types with equal probability. In this manner, the CONTROL operator is an important baseline in that it contains no causal guidance.

However, because the CAUSAL-W-H operator includes guidance for both where and how mutation occurs, the CONTROL operator is not adequate

by itself as a basis for control experiments. Additional operators are needed, which are referred to as Causal-W and Causal-H, to better distinguish the contribution of "where" guidance and "how" guidance. The Causal-W operator includes causal-guidance for where mutation is applied (i.e., the selection of a node for mutation is performed exactly as in the Causal-W-H operator) but the does not include causal-guidance for how the mutation occurs (i.e., the modification to the selected node is determined exactly as in Control). The Causal-H operator is just the opposite: a node is selected for mutation without any causal guidance as in the Control operator, but the modification to that node is selected using "how" guidance as in the Causal-W-H operator.

5.6.4 Experimental Methods

Four evolutionary systems were developed and used to evolve synthetic social networks, where each evolutionary system used one of the four different mutation operators described above. For convenience, these evolutionary systems are referred to by the name of the mutation operator that they employ: Control, Causal-H, Causal-W, and Causal-W-H. Each of the evolutionary systems were used to generate synthetic social networks that match four different real-world social network data sets. While these data sets include directional ties and multiple connection types, for this experimental evaluation these facets are ignored (i.e., the connections are treated

as undirected and all connection types are treated equivalently as one generic type). These four social networks are described below:

- **Academy**: This data set contains information gathered from the Chronicle of Higher Education's annual report about 284 PhD granting sociology departments and their connections to each other in terms of graduation and hiring. A connection from department A to department B indicates that department B hired a PhD produced by department A (Grannis, 2010).

- **Gang**: This data set contains information about 140 residents of a gang-dominated neighborhood in the Los Angeles area, their connections with each other, and their affiliation to either gangs, youth groups, or neither (Grannis, 2009).

- **Terrorist**: This data set contains a subset of data from the Profile in Terror (PIT) project which was collected by the MIND Lab at the University of Maryland. The PIT knowledge base captures terrorism intelligence extracted from various sources like news media reports. This particular subset is comprised of over 800 actors such as terrorists, terrorist leaders, politicians or people while connections represent relationships between them (Zhao et al., 2006).

- **Attacks**: This data set is also a subset from the PIT project. In this data set nodes represent just under 500 terrorist attacks such as bomb-

ings, kidnapping, arson, and other attacks, while connections indicate co-located attacks (Zhao et al., 2006).

Histograms of clustering coefficients for each of these data sets are shown in Figures 5.4 through 5.7. Note that the four data sets have quite distinct histograms. The Academy and Gang histograms are much more evenly distributed than the Attacks and Terrorist histograms. Furthermore, while most nodes in the Terrorist data set appear to have clustering coefficients in the middle range, most nodes in the Attacks data set have very high clustering with a second smaller group of nodes that are pendants.

Three hundred trials of each evolutionary system were conducted for each target data set with the goal of producing a matching synthetic social network with 100 nodes. In each trial, an initial population of 128 individuals was randomly generated and the evolutionary process was executed for 5000 generations. Individuals in the initial population were created by constructing a network of 100 nodes and adding 100 random connections. Tournament selection with tournament size two were used in all trials. In each generation, exactly one offspring was created by elitism. Each of the remaining 127 offspring in each generation was created by using tournament selection to choose a parent, and then either copying that parent without modification (10% of the time) or applying mutation (90% of the time). A mitigation factor of 0.10 was used in all causally-guided variants. This pa-

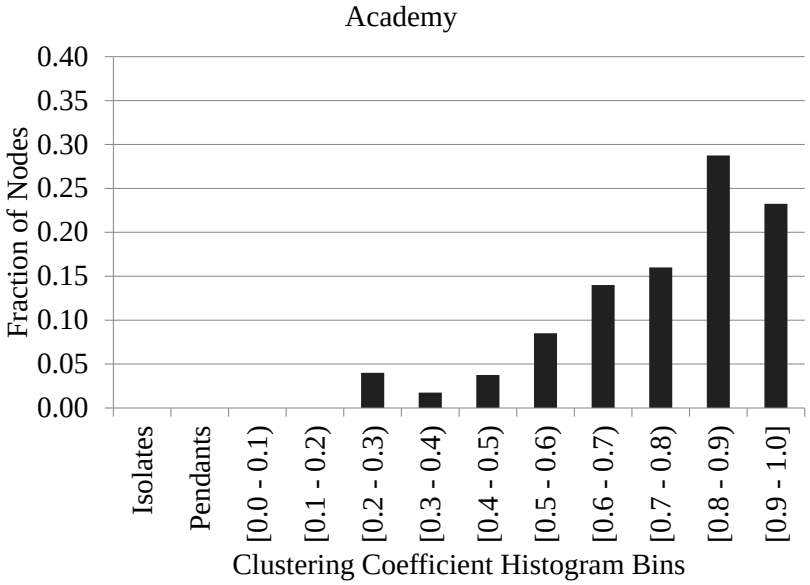

Figure 5.4: Histogram plot illustrating the distribution of clustering coefficients observed in the Academy social network.

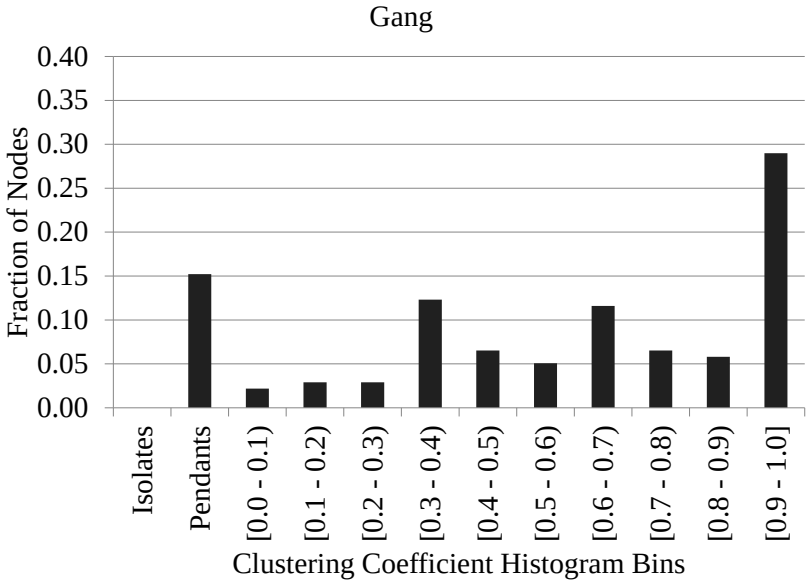

Figure 5.5: Histogram plot illustrating the distribution of clustering coefficients observed in the Gang social network.

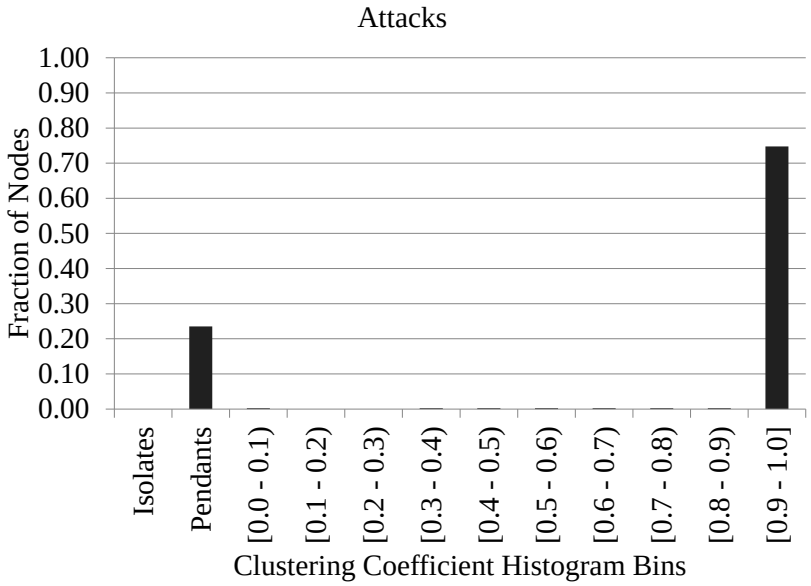

Figure 5.6: Histogram plot illustrating the distribution of clustering coefficients observed in the Attacks social network.

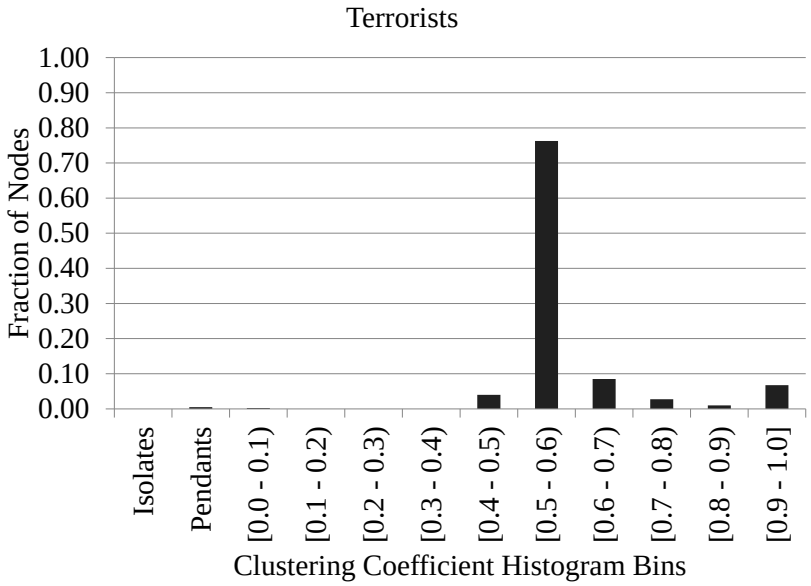

Figure 5.7: Histogram plot illustrating the distribution of clustering coefficients observed in the Terrorist social network.

rameter value was found via a small number of test runs; it may be not be optimal but was found to be effective in this study.

The results of all 4800 evolutionary trials (300 trials * 4 evolutionary systems * 4 target data sets) were examined, and analysis was performed as follows. For each target data set, data was collected to determine how often each of the four evolutionary systems reached a number of different error measures by generation 5000. In the context of generating synthetic social networks, the selection of a single goal for the error measure is subjective and largely application-dependent. Furthermore, the ability of the evolutionary systems to reach any particular error measures is not the same across target data sets (i.e., some target data sets are "harder" for the evolutionary systems and therefore the evolutionary systems do not reach as low of an error measure as with an "easier" target data set). For these reasons, the goal error measures that were used to evaluate the performance of the various evolutionary systems were not held constant across all target data sets. As is discussed in the next section, it was determined that for each target data set, there is a range of goal error measures across which the difference between evolutionary systems is most pronounced. It is across this range of error measures that results are reported below. Additionally, for each target data set a single goal error measure was selected at which the performance of the evolutionary systems was most distinct, and additional data was collected to determine how often each evolutionary system was able to reach this error measure within various numbers of generation (rather than just by

the end of the trial). Combined, these analyses reveal how frequently as well as how quickly trials of each evolutionary system reach goal error measures for various target data sets. Z-tests with 99% confidence levels were used to evaluate the statistical significance of the reported results.

5.6.5 Results

Figures 5.8 through 5.15 show the fraction of each evolutionary system's trials that were able to reach various error measures by the 5000th generation as well the fraction of each evolutionary system's trials that were able to reach specific error measures by various generation limits when applied to various target data sets. Error bars in all figures are used to indicated 99% confidence intervals. There was a fair amount of variation in the performance of the evolutionary systems when applied the different target data sets. In other words, some target data sets were "easier" for the evolutionary systems to match and lower error measures were reached than with other more "difficult" target data sets. Accordingly, the error measures used in Figures 5.8 through 5.15 are not the same across all data sets, and instead were selected to highlight the difference in performance between the evolutionary systems on each particular target data set.

The main results are summarized in a bulleted list here, for convenience, while more detailed description of the results appear in the paragraphs below.

- The evolutionary systems that employ "where" guidance clearly outperform analogous systems that do not. For example, the fraction of trials of CAUSAL-W vs. CONTROL systems that reached various measure are as follows: 95% vs. 0% reached error measure of 0.0 on Gang data set, 70% vs. 0% reached error measure of 1.75 on Attacks data set, 70% vs. 3% reached error measure of 1.00 on Academy data set, and 90% vs. 20% reached error measure of 3.25 on Terrorist data set.

- This relationship appears to hold throughout the evolutionary process, rather than just at generation 5000. At no point in the evolutionary process did CONTROL or CAUSAL-H systems ever outperform the CAUSAL-W system.

- While there are numerous data points at which CAUSAL-W-H outperforms CAUSAL-W or CAUSAL-H outperforms CONTROL, only very few of these differences are statistically significant. "How" guidance is particularly ineffective when applied to the Attacks data set, with 0% of CAUSAL-H trials reaching any of the error measures examined.

Figure 5.8 shows the fraction of each evolutionary system's trials that were able to reach various error measures by the 5000th generation when applied to the Academy target data set. None of the trials of any of the systems reached an error measure of 0.40, and all trials of all systems reached an error measure of 2.50. A very high fraction of trials (more than 75%) of all evolutionary systems were able to reach an error measure of 1.75, and

very few (about 12% of CAUSAL-W-H trials) were able to reach an error measure of 0.50. Between these two extremes, however, there was a clear difference between the various systems. The CONTROL evolutionary system was generally outperformed by the causally-guided variants. The difference between CONTROL and both CAUSAL-W and CAUSAL-W-H was very clear at particular error measures, such as 1.00, which only 3% of the CONTROL trials reached compared to more than 70% of the CAUSAL-W and CAUSAL-W-H trials. The CAUSAL-H outperformed the CONTROL at the 1.00, 1.25, 1.50 and 1.75 error measures to a statistically signficant degree with 99% confidence level. Contrary to my expectations, the difference in performance between CAUSAL-W and CAUSAL-W-H is difficult to characterize. Only at the 0.75 error measure is the difference between the two statistically significant, with CAUSAL-W-H outperforming CAUSAL-W.

Figure 5.9 shows the fraction of each evolutionary system's trials that reach an error measure of 0.75 at various points in the evolutionary process when applied to the Academy target data set. The CAUSAL-W-H system outperforms all other systems throughout the evolutionary process. The difference between the CAUSAL-W-H and CAUSAL-W remains about the same throughout the evolutionary process, with an extra 10 to 20% of CAUSAL-W-H reaching the error measure as compared to CAUSAL-W trials. This difference is statistically significant at all generations listed. Only a very small fraction of the CAUSAL-H trials (less than 2%) and none of the CONTROL trials reached this error measure.

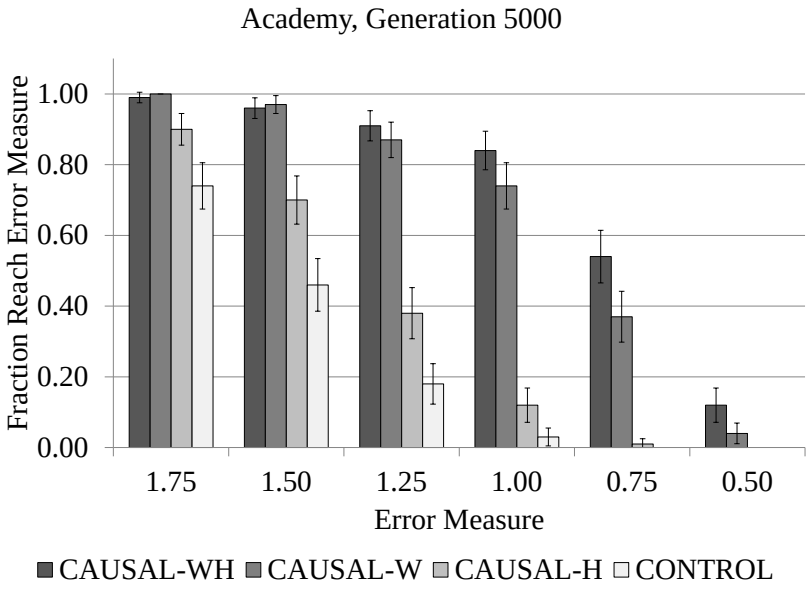

Figure 5.8: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach various error measures by generation 5000, when applied to the Academy target social network. Error bars indicate 99% confidence intervals.

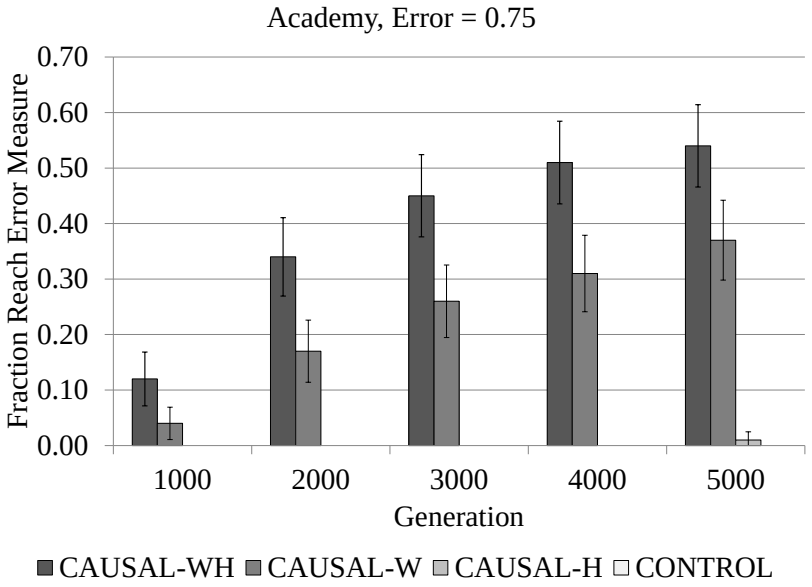

Figure 5.9: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach specific error measures by various generations, when applied to the Academy target social network. Error bars indicate 99% confidence intervals. CONTROL and CAUSAL-H results are not visible because no trials of those systems reached the target error measure (i.e., each bar for those systems is at 0).

Figure 5.10 and 5.11 illustrate similar analyses when the evolutionary systems are applied to the Terrorists target data set. None of the trials of any of the systems reached an error measure of 1.50, and all trials of every system reached an error measure of 5.50. As shown in Figure 5.10, more than 90% of CAUSAL-W and CAUSAL-W-H trials reached an error measure of 3.25 by generation 5000, while only 35% of the CAUSAL-H and 20% of the CONTROL trials did. A small fraction (less than 25%) of the CAUSAL-W and CAUSAL-W-H trials were able to reach an error measure of 2.00, while none of the trials from the other variants did. As with the Academy data set, the CAUSAL-W and CAUSAL-W-H variants outperform the CONTROL system to a 99% confidence level at all observed error measures. The CAUSAL-H variant outperforms the CONTROL as well, but this difference is only statistically significant at the 3.25 error measure. As with the Academy data set, it is more difficult to characterize the difference in performance between the CAUSAL-W and CAUSAL-W-H variants. While the CAUSAL-W-H variant outperforms CAUSAL-W at all error measures, this difference is not ever statistically significant.

Figure 5.11 shows the fraction of each evolutionary system's trials that were able to reach an error measure of 2.50 by various generation limits when applied to the Terrorists data set. At generation 1000, roughly 9% of the CAUSAL-W-H trials reached this error measure, compared to only 1% of the CAUSAL-W trials, and none of the trials from the other evolutionary systems. By generation 2000, almost twice (32% vs. 18%) as many trials of the

Causal-W-H reached this error measure as did the Causal-W system. This margin of difference between Causal-W and Causal-W-H (about 10%) remains the same from generation 2000 to 5000, but is not statistically significant. Only very small fractions (less than 5%) of the Control and Causal-H trials were able to reach the 2.50 error measure, even by generation 5000.

The Gang target data set proved to be the "easiest" for the evolutionary systems, as large proportions of trials from all variants were able to reach substantially lower error measures than when applied to the other target data sets. Figure 5.12 illustrates the fraction of each evolutionary system's trials to reach various fitness levels by generation 5000. All of the trials of every system reached an error measure of 1.00. Nearly all (100% for Causal-W and Causal-W-H, 99% for Causal-W, and 89% for Control) the trials from all evolutionary systems were able to reach an error measure of 0.75. However, for lower error measures of 0.00, 0.15, 0.30, 0.45 and 0.60, there was a large difference between the various evolutionary systems. While over 95% of the Causal-W-H trials and over 87% of the Causal-W trials reached a 0.00 error measure, less than 10% of the Causal-H and none of the Control trials reached this fitness level. The difference between Causal-W-H and Causal-W and the difference between these two variants and the Causal-H and Control systems are all statistically significant to a 99% confidence level, at error measure 0.0.

Figure 5.13 shows the fraction of each evolutionary system's trials that

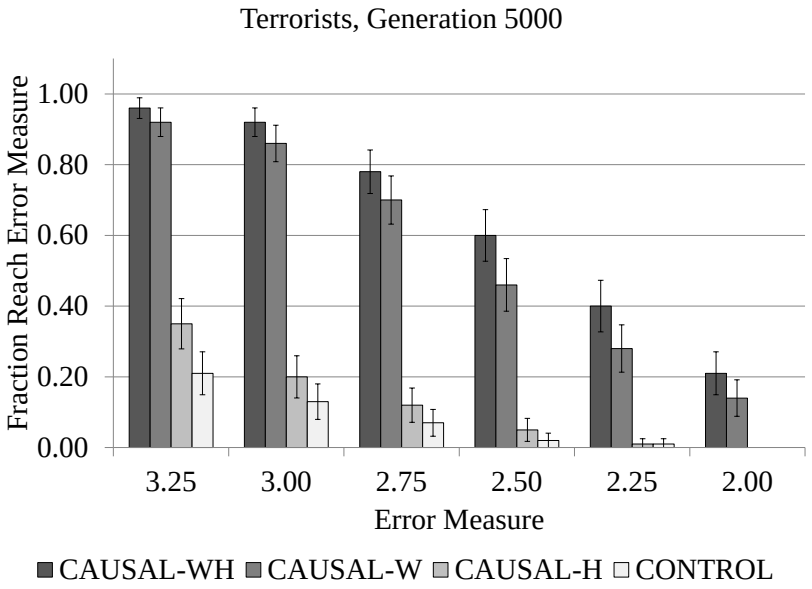

Figure 5.10: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach various error measures by generation 5000, when applied to the Terrorist target social network. Error bars indicate 99% confidence intervals.

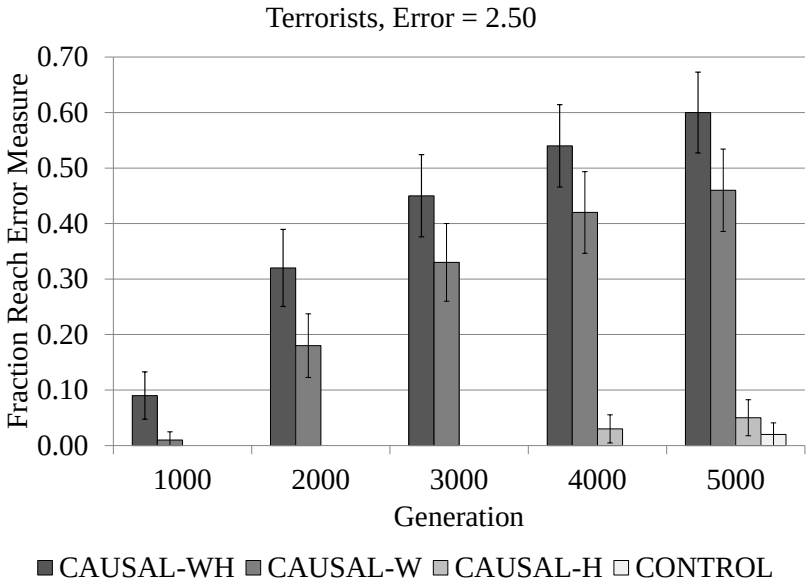

Figure 5.11: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach specific error measures by various generations, when applied to the Terrorist target social network. Error bars indicate 99% confidence intervals. In some instances, CONTROL and CAUSAL-H results are not visible because no trials of those systems reached the target error measure (i.e., each bar for those systems is at 0).

reached an error measure of 0.00 over the course of the evolutionary process. The difference in performance between the CAUSAL-W and CAUSAL-W-H variants was larger at generation 1000 than it was at generation 5000. At generation 1000, about 80% of the CAUSAL-W-H trials were able to reach an error measure of 0.0, while only 43% of the CAUSAL-W trials were. The CAUSAL-W-H outperforms CAUSAL-W to a statistically significant level across all generations listed. Only very small fractions (less than 5%) of the CAUSAL-H trials reached this error measure, and none of the CONTROL trials.

Figure 5.14 shows the fraction of each evolutionary system's trials that reached various error measures when applied to the Attacks target data set. None of the trials of any of the systems were reached an error measure of 0.50, and all of trials of all systems reached an error measure of 5.00. Between these two extremes, there was a clear difference in performance between the systems. Over 70% of the CAUSAL-W and CAUSAL-W-H trials reached an error measure of 1.75, and almost 20% reached an error measure of 1.00. None of the trials of the CONTROL or CAUSAL-H systems were able to reach any of these error measures. There were no large difference between the CAUSAL-W and CAUSAL-W-H trials at any of the error measures, and none were statistically significant.

The fraction of each evolutionary system's trials to reach an error measure of 1.00 over the course of the evolutionary process is illustrated in Figure

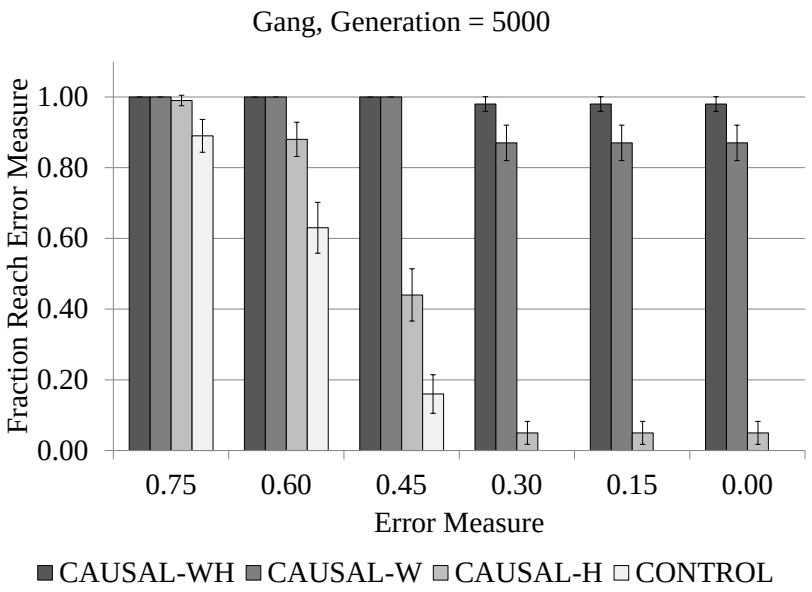

Figure 5.12: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach various error measures by generation 5000, when applied to the Gang target social network. Error bars indicate 99% confidence intervals.

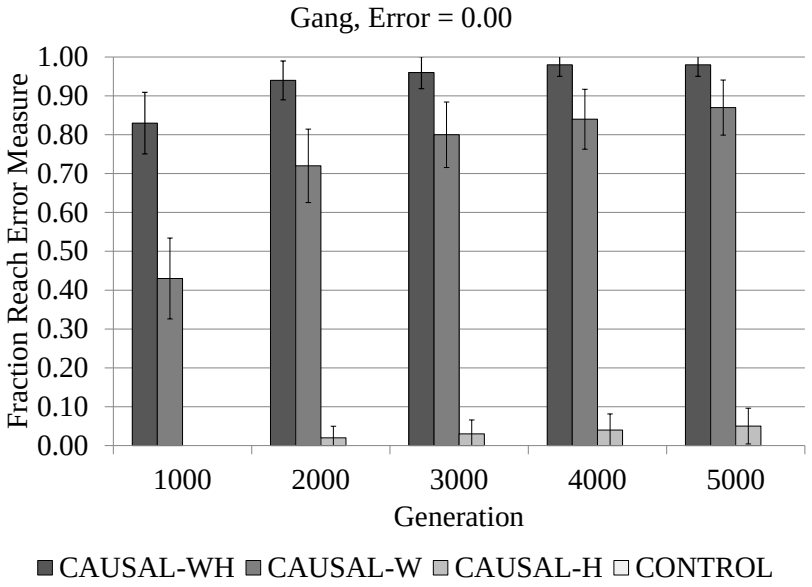

Figure 5.13: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach specific error measures by various generations, when applied to the Gang target social network. Error bars indicate 99% confidence intervals. In some instances, CONTROL and CAUSAL-H results are not visible because no trials of those systems reached the target error measure (i.e., each bar for those systems is at 0).

5.15. At each generation listed, there is virtually no difference between the performance of the CAUSAL-W and CAUSAL-W-H system. None of the CAUSAL-H or CONTROL trials reached an error measure of 1.00.

5.7 Multi-Step "How" Guidance

It is immediately clear from the results described above that the benefits of using causal-guidance to influence how mutation is applied is questionable. The CAUSAL-H system did outperform the CONTROL system in most instances, and this difference was statistically significant in some instances when applied to the Terrorist, Academy, and Gang target data sets. However, for the Attacks data set, none of the trials of the CONTROL or CAUSAL-H were systems were able to reach the error measures examined. Furthermore, the hypothesized synergistic benefits of combining both "where" and "how" guidance is not supported by the data. Across most target data sets and most error measures the CAUSAL-W-H does outperform the CAUSAL-W system. However, this difference is rarely statistically significant, and there are a number of instances where the CAUSAL-W system outperform the CAUSAL-W-H. That is, adding causal guidance of how mutation is applied does not seem to improve performance. In particular, when applied to the Attacks data set, CAUSAL-W-H offers no benefit at all over CAUSAL-W, and CAUSAL-H fails to ever reach any of the listed error measures.

Further examination of these results, particularly with regard to the At-

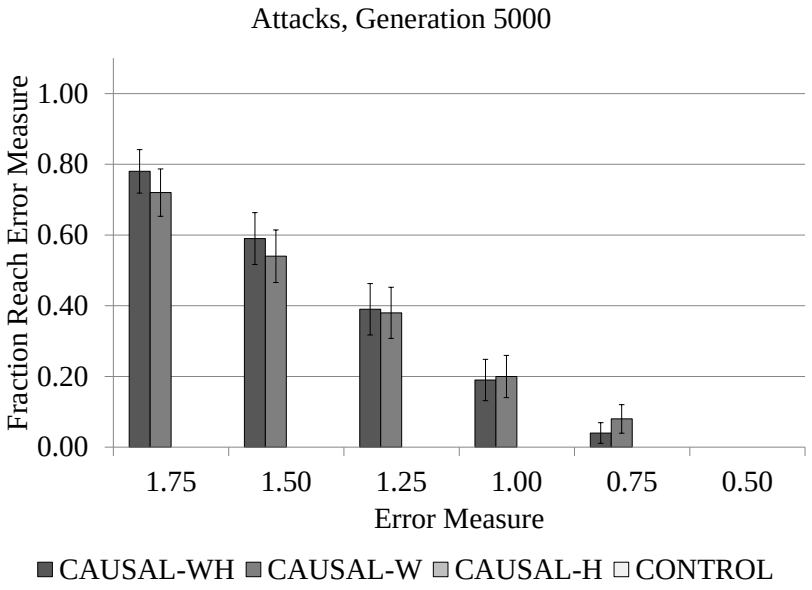

Figure 5.14: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach various error measures by generation 5000, when applied to the Attacks target social network. Error bars indicate 99% confidence intervals.

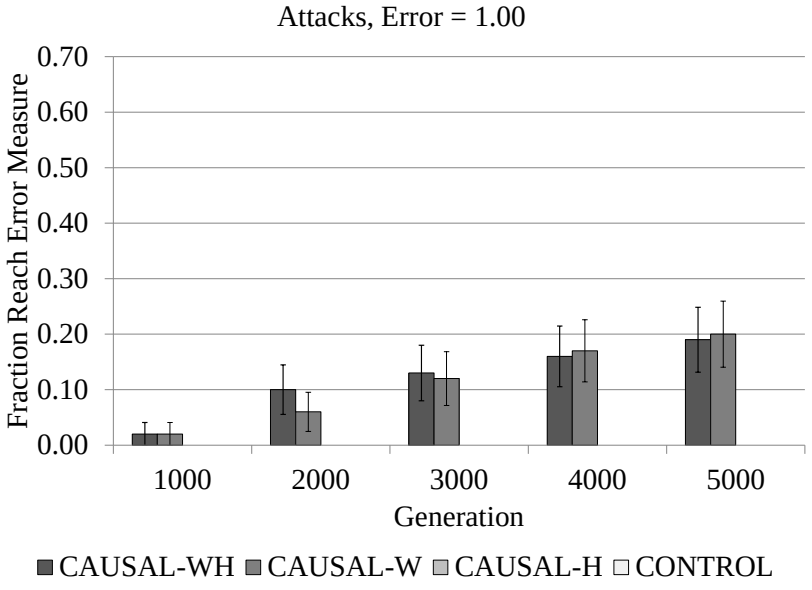

Figure 5.15: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-W, CAUSAL-W-H systems to reach specific error measures by various generations, when applied to the Attacks target social network. Error bars indicate 99% confidence intervals.

tacks target data set, reveals some reasons why "how" guidance of mutation is not effective in its current form, and suggests some possible improvements. The best social networks produced by each of the 300 trials of the CAUSAL-H system were examined, the best overall network (i.e., the one with the lowest error measure) was identified, and the effects of applying CAUSAL-H mutation to this network were examined. In this manner, the best evolved network that was discovered by the CAUSAL-H system was analyzed in an effort to determine why the CAUSAL-H evolutionary process did not make any improvements to this network.

The best CAUSAL-H evolved network has an error measure of over 1.90, which is substantially higher than error measures typically reached by the CAUSAL-W and CAUSAL-W-H systems (see Figure 5.14). Figure 5.16 shows the histogram of clustering coefficients in this evolved network as compared to the target histogram derived for Attacks target social network. The evolved network has too few nodes in the Pendant and [0.9 - 1.0] bins, and too many nodes in the [0.0 - 0.1), [0.1 - 0.2), [0.2 - 0.3), and [0.3 - 0.4) bins. The CAUSAL-H mutation operator was applied to this network 1000 distinct times, and each time the utility scores that were calculated for each mutation type were collected. It was discovered that in all 1000 applications of CAUSAL-H, the calculated utility score of the ADD_E mutation type was 0.0. Examination of the histogram data in Figure 5.16 reveals why this is the case. If any node in the [0.0 - 0.4) range is selected for mutation, the expected effect of applying the ADD_E mutation will be to decrease the

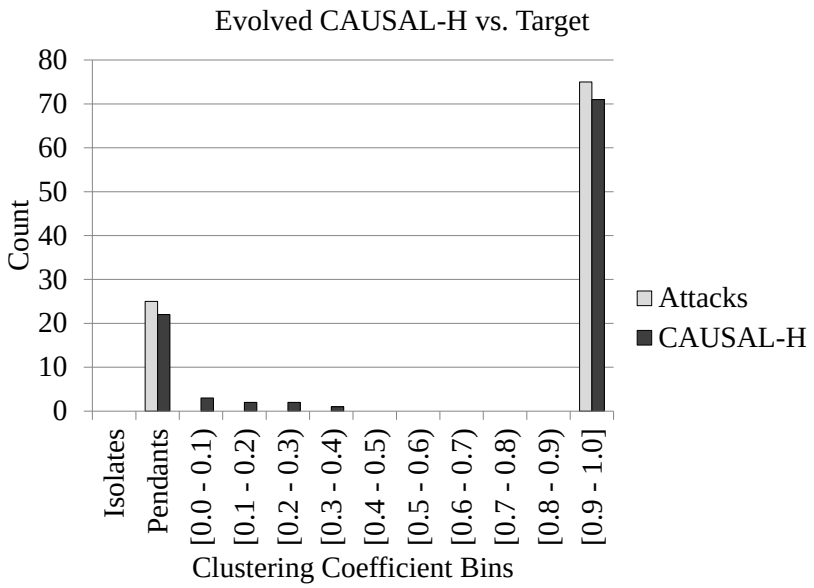

Figure 5.16: Histogram data for the clustering coefficients in Attacks target social network and the most fit social network produced by any CAUSAL-H trial.

clustering coefficient of the node, effectively moving the node to a bin in which there are already too many nodes. It is not possible to move a node from the [0.0 - 0.4) range to a bin that is desired (the Pendant or [0.9 - 1.0) bin) through a single application of the ADD_E mutation type. Similarly, applying ADD_E to a node that is in the Pendant or [0.9 - 1.0] bin will result in that node being placed in a bin in which there are already too many nodes. In short, applying the ADD_E mutation to any node in the network will not result in any of the observed performance symptoms being addressed, and accordingly the utility is assessed as being 0.

This analysis does not suggest that the causal knowledge being employed is inaccurate. In fact, for each of the observed mutations it was confirmed that applying the ADD_E mutation does in fact increase the error

of the individual. However, it does highlight the problematic way in which "how" guidance is applied. While the ADD_E mutation type may not result in an *immediate* improvement to the network, it is possible that it is a necessary first step in a series of mutations that may ultimately improve the network. By definition, a local optima is a solution that is located in an area of solution space such that all nearby solutions are inferior. As such, the only way for the evolutionary process to move an individual from a local optima to a global optima is by traversing an area of solution space that is characterized by higher error (poorer fitness). Guiding how mutation is applied based solely on the immediate effects of applying each mutation type may actually be *preventing* the search process from temporarily moving through an area of solution space that has higher error, resulting in the unintended effect of the search process getting stuck at local optima.

Consider a scenario in which a node with 4 nodes in its open neighborhood and 4 connections between those neighbors (a clustering measure of $\left\{{4 \atop 4}\right\}$) has been selected for mutation. Figure 5.17 illustrates clustering measure of this node (top center of the figure), the expected immediate effects of applying a mutation type to this node (middle row of clustering measures in the figure), as well as the expected effects of applying a second subsequent mutation (bottom row of clustering measures in the figure), based on our ExpCofMtoC() calculations. For example, the expected result of applying the ADD_E mutation to $\text{n}\left\{{4 \atop 4}\right\}$ is $\text{n}\left\{{4 \atop 5}\right\}$, and the expected effect of subsequently applying DEL_T is $\text{n}\left\{{3 \atop 5}\right\}$. In the context of this figure, the cur-

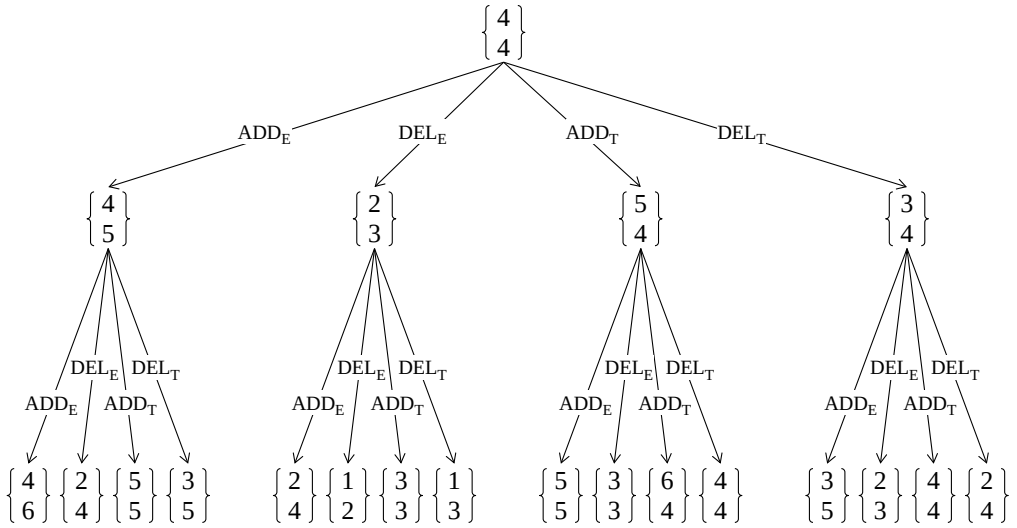

Figure 5.17: The expected immediate effects of applying a mutation type to a node with $\left\{{4 \atop 4}\right\}$ clustering, as well as the expected effect of subsequent mutations.

rent form of "how" guidance only considers the immediate effect of applying mutation types (middle row), and does not consider the effects of subsequent mutations (bottom row). For example, in a situation where $\left\{{4 \atop 5}\right\}$ maps to a histogram bin that has too many nodes, and $\left\{{3 \atop 5}\right\}$ maps to a histogram bin that has too few, the calculated utility of applying $\mathrm{DEL_E}$ to $\mathrm{n}\left\{{4 \atop 4}\right\}$ clustering will be quite low, despite the fact that this mutation type is a step toward producing a $\mathrm{n}\left\{{3 \atop 5}\right\}$, which is highly desired.

Fortunately, this line of thinking also suggests ways in which "how" guidance of mutation occurs may be improved: rather than considering only the immediate effects of each mutation, a more long-term view must be taken in which the effects of *subsequent* mutations are also considered. The utility of a particular mutation type must depend not only the performance

symptoms that it may immediately addresses, but also on the performance symptoms that may be addressed by subsequent mutations. To that end, the Utility(m,n(C)) function described earlier is extended to also take into account the number of mutations to look ahead, as follows:

Utility(m,n(C),x)= max(

Utility(m,n(C)),

d * Utility(ADD_E, ExpMtoN(m,n(C)), x-1),

d * Utility(DEL_E, ExpMtoN(m,n(C)), x-1),

d * Utility(ADD_T, ExpMtoN(m,n(C)), x-1),

d * Utility(DEL_T, ExpMtoN(m,n(C)), x-1))

where x is the number of mutations to look ahead. Additionally, the Utility function is defined as evaluating to 0 whenever $x \leq 0$. In this manner, all of the clustering measures that are "reachable" within x mutation steps are examined and considered. As in Section 5.6, the utility of each of reaching each clustering measure is calculated based on the number of instances there are of any performance symptoms that this clustering measure addresses. Multiplying each recursive call by a constant d (≤ 1.0) has the effect of penalizing the utility scores for clustering measures that require a long sequence of mutations to reach. For example, the utility of a clustering measure that is reachable by 3 mutations will be discounted by a factor of d^2.

Note that the Utility(m,n(C)) function defined in Section 5.6 is equivalent to Utility(m,n(C),1) as defined here. In this sense, the CAUSAL-H and

CAUSAL-W-H operators previously defined and evaluated can be seen as using multi-step "how" guidance presented here, but with a multi-step look ahead of only 1. As before, when applying multi-step "how" guidance a single mutation type is selected stochastically with probability proportional to its utility score, and a mitigation factor of 0.10 is used to dampen the effects of the guidance.

To evaluate multi-step "how" guidance, two additional evolutionary systems were developed: CAUSAL-W-HX and CAUSAL-HX. The CAUSAL-W-HX system uses "where" guidance and multi-step "how" guidance with an X step look-ahead. The CAUSAL-HX system is identical, but does not use "where" guidance. Note that the X is CAUSAL-W-HX and CAUSAL-HX is actually a parameter value that is replaced for any specific instance. For example, a CAUSAL-W-HX evolutionary system with 5-step "how" guidance is referred to as CAUSAL-W-H5. Thus, for clarify and consistency, the CAUSAL-H and CAUSAL-W-H evolutionary systems previously evaluated in this chapter are referred to below as CAUSAL-H1 and CAUSAL-W-H1.

5.7.1 Experimental Methods

Two new evolutionary systems that use the CAUSAL-W-HX and CAUSAL-HX operators were developed and used to evolve synthetic social networks that match the clustering characteristics of the four target social networks (Academy, Attacks, Terrorist, and Gang). Three hundred trials of each evo-

lutionary system were conducted, with parameter settings of d = 0.8 and x = 5. All other parameter settings were the same as in the previous section. The same types of analysis as in the previous study (fraction of each evolutionary systems trials to reach various fitness levels) were performed, allowing for direct comparison of performance with the CONTROL, CAUSAL-W, CAUSAL-H1, and CAUSAL-W-H1 systems previously presented and analyzed. The main goal of this experimentation is to evaluate the new methods for causally-guiding how mutation occurs as compared to the previously presented short-sighted methods as well as to the control systems that do not employ causal guidance. Accordingly, the performance of CONTROL, CAUSAL-H1, and CAUSAL-H5 systems were compared to each other, as were the CAUSAL-W, CAUSAL-W-H1, and CAUSAL-W-H5.

5.7.2 Results

Figure 5.18 shows the fraction of CONTROL, CAUSAL-H1, and CAUSAL-H5 trials that reached various error measures by the 5000th generation when applied to the Academy target data set. Both the CAUSAL-H1 and CAUSAL-H5 systems outperformed the CONTROL system to a statistically significant level. However, there is no discernible difference between the CAUSAL-H1 and CAUSAL-H5 systems. Figure 5.19 shows a similar analysis for the CAUSAL-W, CAUSAL-W-H1, and CAUSAL-W-H5 systems. While a larger fraction of CAUSAL-W-H5 trials than CAUSAL-W-H1 trials reach each of

the various fitness levels, this difference in performance is not statistically significant to a 99% confidence level.

Figure 5.20 shows the fraction of CONTROL, CAUSAL-H1, and CAUSAL-H5 trials that reached various error measures, while Figure 5.21 shows the same for the CAUSAL-W, CAUSAL-W-H1, and CAUSAL-W-H5 trials when applied to the Attacks data set. The difference in performance between the evolutionary systems that employ causal guidance for where mutation occurs (shown in Figure 5.21) and those that do not (shown in Figure 5.20) was quite large, and so different ranges of error measures were used in each analysis. As seen in Figure 5.20, the CAUSAL-H5 clearly outperforms the CAUSAL-H1 system at all listed error measures, such as 77% versus 36% of trials that reach error measure 3.25. The difference between CAUSAL-W-H1 and CAUSAL-W-H5 is equally dramatic, as seen in Figure 5.21 with 66% of CAUSAL-W-H5 versus 19% of CAUSAL-W-H1 trials reaching an error measure of 1.00, and 44% versus 4% reaching an error measure of 0.75. The difference in performance between these two systems was statistically significant to a 99% confidence level at all examined error measures.

Figure 5.22 and 5.23 show similar analysis of the evolutionary systems when applied to the Gang data set. There is little difference in performance between the CAUSAL-H1 and CAUSAL-H5 systems, as seen in Figure 5.22. A higher fraction of CAUSAL-H5 trials than CAUSAL-H1 trials reach various error measures, but never to a statistically significant level. All of the

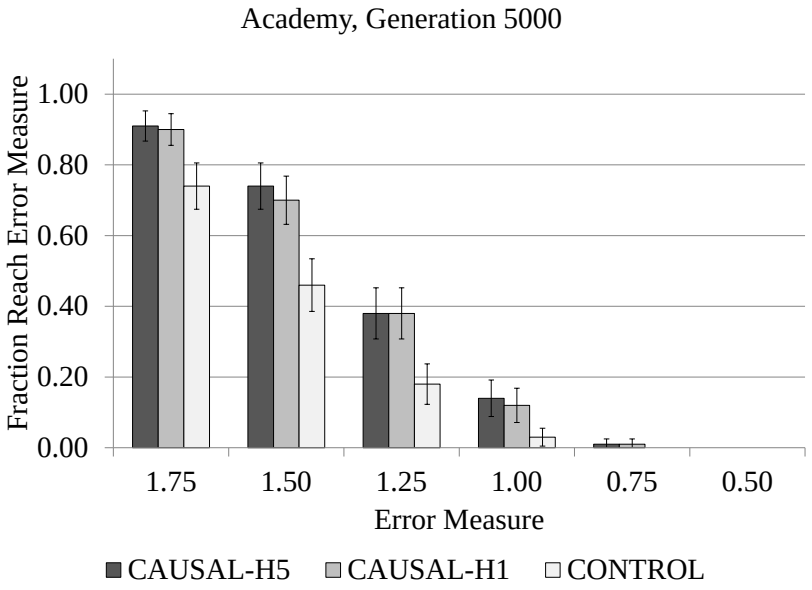

Figure 5.18: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-H5 systems to reach various error measures by generation 5000, when applied to the Academy target social network. Error bars indicate 99% confidence intervals.

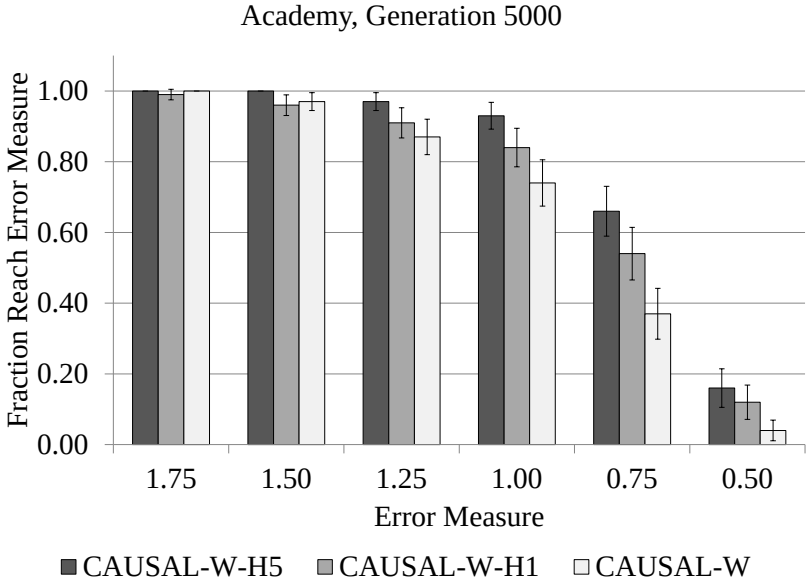

Figure 5.19: The fraction of trials of the CAUSAL-W, CAUSAL-W-H, CAUSAL-W-H5 systems to reach various error measures by generation 5000, when applied to the Academy target social network. Error bars indicate 99% confidence intervals.

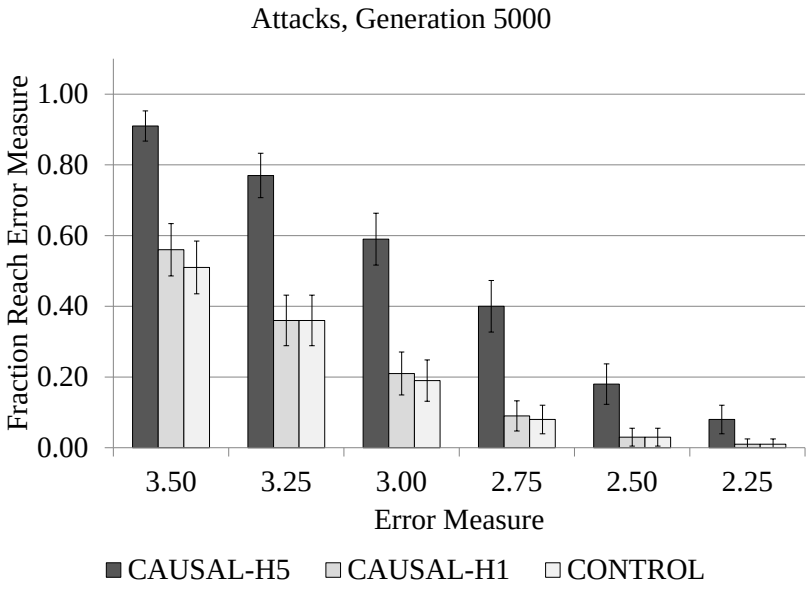

Figure 5.20: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-H5 systems to reach various error measures by generation 5000, when applied to the Attacks target social network. Error bars indicate 99% confidence intervals.

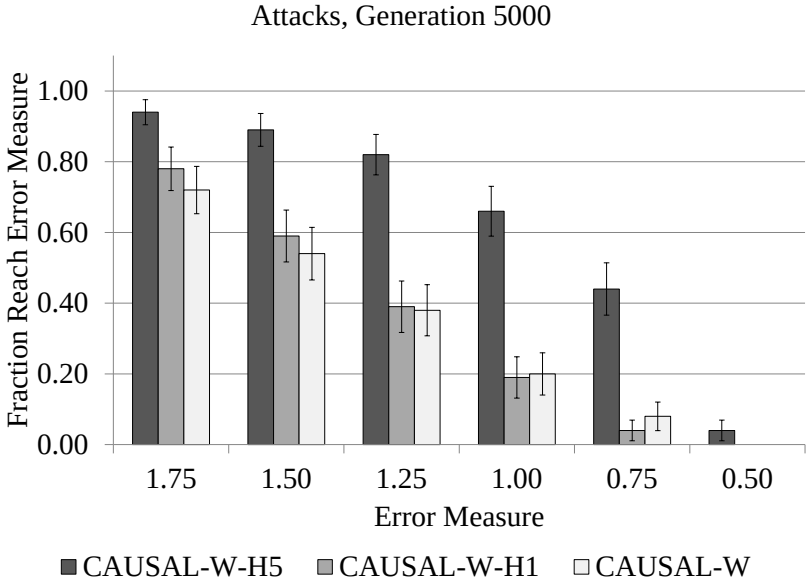

Figure 5.21: The fraction of trials of the CAUSAL-W, CAUSAL-W-H, CAUSAL-W-H5 systems to reach various error measures by generation 5000, when applied to the Attacks target social network. Error bars indicate 99% confidence intervals.

evolutionary systems in Figure 5.23 performed so well that it is difficult to characterize any difference between CAUSAL-W-H1 and CAUSAL-W-H5.

The same analysis of the evolutionary systems when applied to the Terrorist data set if illustrated in Figure 5.24 and 5.25. The CAUSAL-H5 system outperforms the CAUSAL-H1 system at many error measures, including 53% versus 35% of trials that reached an error measure of 3.25. The difference between CAUSAL-W-H1 and CAUSAL-W-H5 was clearer, as seen in Figure 5.25. More than twice as many CAUSAL-W-H5 trials (47%) as CAUSAL-W-H1 trials (21%) reached an error measure of 2.00. The CAUSAL-W-H5 system outperformed CAUSAL-W-H1 to a statistically significant level at all but the highest error measures listed (3.00 and 3.25).

5.8 Discussion

Some general findings can be distilled from the above analysis and results. These findings and their implications for causally-guided evolutionary computation in general are discussed here.

It was observed that across all target data sets there was some high error measure which all trials of all evolutionary systems reached. Conversely, for all target data sets there was some very low error measure that none of the trials of any evolutionary system reached (with the exception of the Gang data set, for which some trials of all evolutionary systems were able to reach a 0.0 error measure). It is between these two extremes in error measure goals

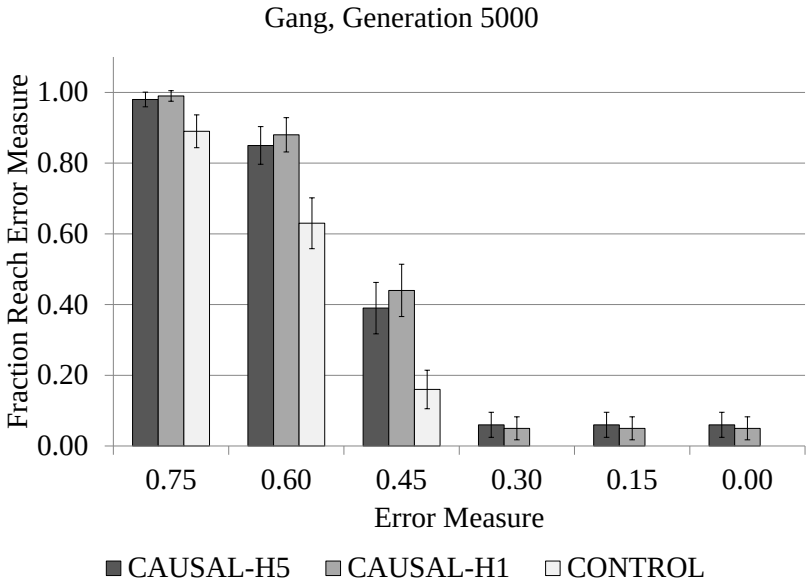

Figure 5.22: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-H5 systems to reach various error measures by generation 5000, when applied to the Gang target social network. Error bars indicate 99% confidence intervals.

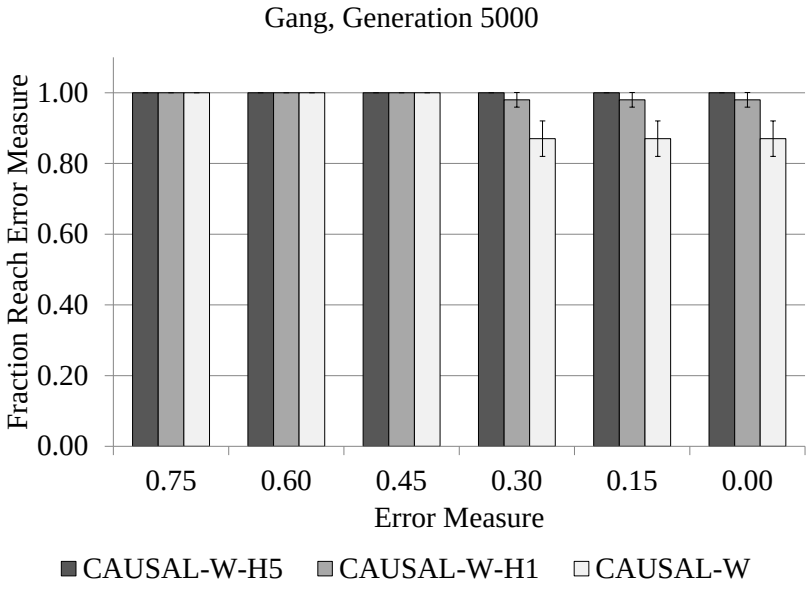

Figure 5.23: The fraction of trials of the CAUSAL-W, CAUSAL-W-H, CAUSAL-W-H5 systems to reach various error measures by generation 5000, when applied to the Gang target social network. Error bars indicate 99% confidence intervals.

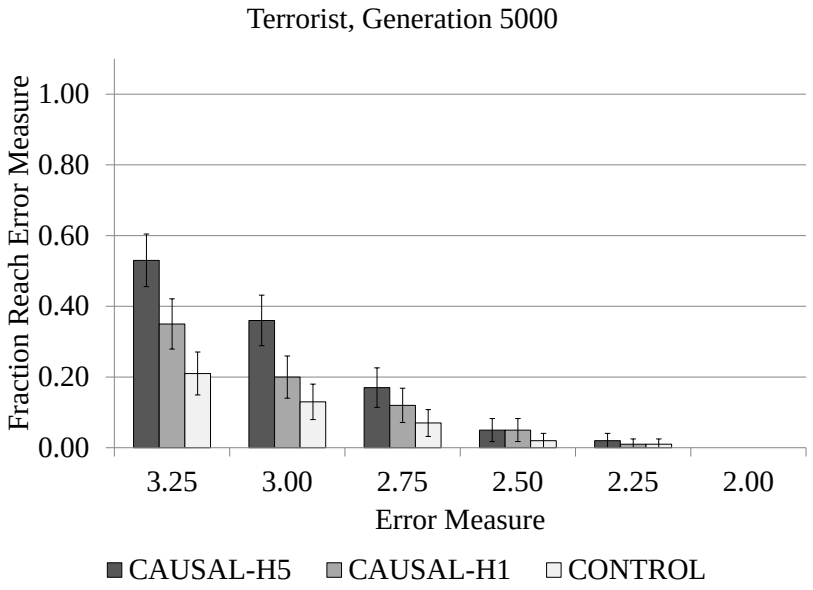

Figure 5.24: The fraction of trials of the CONTROL, CAUSAL-H, CAUSAL-H5 systems to reach various error measures by generation 5000, when applied to the Terrorist target social network. Error bars indicate 99% confidence intervals.

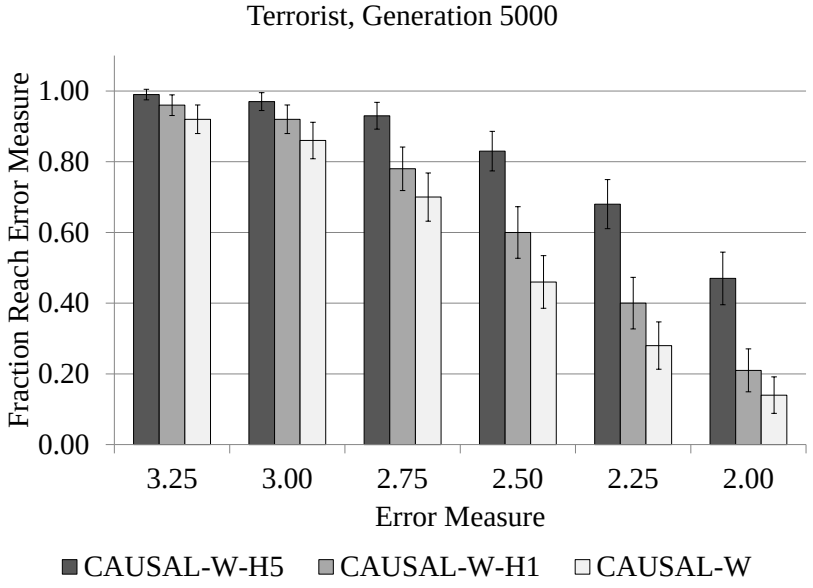

Figure 5.25: The fraction of trials of the CAUSAL-W, CAUSAL-W-H, CAUSAL-W-H5 systems to reach various error measures by generation 5000, when applied to the Terrorist target social network. Error bars indicate 99% confidence intervals.

that the differences between the performance of the evolutionary systems is clear, with causally-guided systems generally and clearly outperforming the control system. This suggests that a key benefit of causally-guided evolution over conventional evolution is a capacity for pushing the boundaries of the types of problems that can be solved and the types of solution that can be discovered. When problems are too "easy" there is little benefit to to including causal-guidance, and when problems are too "hard" even including causal guidance may not be enough for evolution to solve the problem. However, as this study suggests, there are likely numerous problems that exist in between these two extremes, and for which the benefits of causal guidance may be significant.

Evolutionary systems that employ causal guidance to influence where mutation occurs and those that employ causal guidance to influence how mutation occurs both outperformed the control evolutionary system. The difference between CAUSAL-H5 and CONTROL systems was statistically significant at a 99% confident level at most error measures across all data sets. This difference in performance was often quite substantial, such as the nearly 60% of CAUSAL-H5 trials that reached an error measure of 3.00 compared to less than 20% of CONTROL trials on the Attacks data set. The difference between CAUSAL-W and CONTROL was more dramatic, with high fractions of CAUSAL-W trials reaching error measures that none of the CONTROL trials did. For example, 87% of CAUSAL-W trials reached an error measure of 0.0 on the Gang data set, but none of the CONTROL trials did. This

clearly demonstrates the effectiveness of causally-guiding where mutation occurs and causally-guiding how mutation occurs, in isolation (i.e., causally-guiding where or how, but not both).

Furthermore, the benefits of casually-guiding where and how mutation occurs appear to be synergistic. This can be seen in that the CAUSAL-W-H5 system outperforms all other evolutionary systems in which only one form of causal guidance is used. Generally, the CAUSAL-W-H5 systems outperformed the CAUSAL-W systems, which in turn outperformed the CAUSAL-H5 systems. Higher fractions of CAUSAL-W-H5 trials than CAUSAL-W trials reached various error measures, by margins such as 68% to 28% (Terrorist, 2.25 RMSE), 66% to 20% (Attacks, 2.25 RMSE), 66% to 37% (Academy, 0.75 RMSE). The difference between CAUSAL-W-H5 and CONTROL is often quite substantial, such at 93% of CAUSAL-W-H5 trials reaching an error measure of 1.00 on the Academy data set, compared to less than 5% of the CONTROL trials. This qualitative difference in performance clearly demonstrates the synergistic power of causally-guiding where and how mutation occurs.

In addition to demonstrating the effectiveness of causally guiding where and how mutation occurs, these results also provide some lessons learned regarding the best way to implement these methods. Specifically, causal guidance to influence how mutation occurs proved to be more complicated than initially anticipated. The initially explored form of causally-guiding how mutation occurs, which only examines the immediate impact of each mutation

type, resulted in various dubious performance benefits. While the CAUSAL-H system did generally outperform the CONTROL, and the CAUSAL-W-H system did generally outperform the CAUSAL-W, these differences in performance were rarely statistically significant and there were some instances where the reverse was true (i.e., it was better to not causally guide how mutation occurs). Evolutionary systems that employed causal guidance for how mutation occurs performed particularly poorly when applied to the Attacks data set.

Deeper analysis of the causal guidance of mutation, particularly as applied to the Attacks data set, revealed a potential problem. As defined, the causal guidance of how mutation occurs examined only the immediate effect of any potential mutation type, and did not consider the effects of any subsequent mutations. In this sense, the causal guidance was "short-sighted" and it was hypothesized that this could be causing the evolutionary process to become stuck in local optima. Accordingly, the causal guidance of how mutation occurs was extended to consider the effects of multiple sequential mutations. Additional experimental analysis support the effectiveness of this approach. Across all data sets, there are numerous instances in which CAUSAL-H5 clearly outperforms CAUSAL-H1, but none where the reverse is true. Furthermore, the benefits of H5 guidance over H1 appears to be more dramatic when combined with causal guidance of where mutation occurs. For all target data sets and for most error measures, the difference in performance between the CAUSAL-W-H1 and CAUSAL-W-H5 systems was

larger than the difference between CAUSAL-H1 and CAUSAL-H5. While the synergy between H1 guidance and W guidance was doubtful (see Section 5.6.5), the results presented here suggest that H5 and W guidance are complementary and synergistic. This clearly demonstrates the effectiveness of causally guiding how mutation occurs, but also suggests the following lesson: when influencing how mutation is applied to individuals, it is important to consider not only the short term effects of applying that mutation, but also the longer term effects of subsequent mutations.

The importance of considering the long term effects of sequences of mutations appears to be particularly important when applied to problems in which many local optima exist. The benefits of H5 versus H1 guidance appears to be most dramatic when applied to the Attacks and Terrorist data sets, as opposed to the Academy and Gang data set. Examination of the distribution of clustering coefficients in these four data sets (see Figure 5.4 through 5.7) reveal that in both the Attacks and Terrorist data sets most nodes are focused into a small number of bins, whereas in Academy and Gang the nodes are distributed more evenly across bins. For the latter type of data set, any single mutation operation may not be sufficient for moving a node from an undesired bin to a desired one. Instead, only sequences of mutations may accomplish this. Accordingly, using a mechanism for causally-guiding mutation that examines multi-step mutations may be particularly important for these particular data sets. More generally, this multi-step analysis may be

particularly important when applying causally-guided evolution to solution spaces in which local optima are surrounded by large sub-optimal regions.

Chapter 6

Acquiring Causal Relations

6.1 Introduction

The goal of this chapter is to evaluate the feasibility of applying causally-guided evolution to problem domains in which causal knowledge is not available a priori and must instead be acquired and used for causal guidance during the evolutionary process. The chapter begins with a more detailed discussion of the motivation for this work. Next, a general approach for acquiring mechanistic causal relations through observation of the evolutionary process is described. As an initial step in evaluating these ideas, an evolutionary system that acquires mechanistic causal relations through observation of the evolutionary process and then uses those relations as the basis for "how" guidance is applied to the task of designing synthetic social networks. This is the same application problem as presented in Chapter 4, but here a priori causal knowledge is not used. The results of this experimental analysis are presented, and the implications for the feasibility of using causally-guided evolution when causal relations are not available a priori is discussed. While it may be possible to acquire both diagnostic and mechanistic causal relations in this manner, in this initial study we focus on the acquisition of mechanistic

causal relations, while the acquisition of diagnostic relations is left as an area for future work.

6.1.1 Motivations

In previous chapters, causally-guided evolutionary computation methods were evaluated in application domains for which at least some knowledge of cause-effect relations was available a priori. In the work presented in the previous three chapters, each subsequent study extended and evaluated the ways in which causal relations are used to guide the evolutionary process. Additionally, the types of application problems against which these methods were evaluated was varied. In the first study (Chapter 3) causally-guided evolution was applied to the task of optimizing weights in a neural network; an application problem for which cause-effect knowledge is both readily available and has been thoroughly vetted through extensive research in neural network learning. In the second and third study (Chapters 3 and 4), causally-guided evolution was applied to the task of antenna array design and synthetic social network design, respectively. In both of these application problems knowledge about causal relations is available from human experts, but is potentially incomplete or inaccurate in some instances. However, in both studies it was demonstrated that these relations can be an effective basis for causally-guiding the evolutionary process.

In this context, the work presented in this chapter represents the next

logical step in extending the types of application problems to which causally-guided evolution may be applied: those in which causal knowledge is not available a priori. Instead, causal relations are acquired during the evolutionary process by observing the interplay between individuals, mutations, and changes in performance. These relations are then used as the basis for causal guidance within the remainder of the same evolutionary process. If successful, the benefits of causally-guided evolutionary computation that have been previously demonstrated (more effective and efficient discovery of solutions) may also be obtained when applied to a wide range of application problems for which a priori causal knowledge is not available.

Furthermore, it may be desirable to use these methods for acquiring causal relations even when a priori causal knowledge is available. There are significant start-up costs associated with implementing causally-guided evolutionary methods, including specification of causal relations, causal guidance, and integration with genetic operators. The ability to acquire causal relations and use them for causal guidance without requiring design or input of causal relations from a human developer is an important step in reducing the burden of working with causally-guided evolution, and greatly improves the attractiveness of these methods for real-world use. In this study, while domain knowledge is used to inform the structure of causal relations and the mechanisms for causal guidance, initial steps are taken to explore the possibility of acquiring causal knowledge.

6.2 Acquiring Causal Relations from the Evolutionary Process

In the previous chapters, diagnostic causal relations were used as the basis for influencing where mutation and crossover occurs, and mechanistic causal relations were used as the basis for influencing how mutation is done. In some problem domains, these types of causal relations may not be available a priori. However, the evolutionary process itself contains a wealth of data from which causal relations may potentially be acquired. Throughout the course of an evolutionary process, individuals with known performance characteristics are modified by mutation operations, resulting in new individuals with altered performance characteristics. Collectively these occurrences comprise a data set from which cause-effect relations may be distilled.

While there are likely many ways that one could attempt to learn causal relations from the evolutionary process, one particular approach is employed here to learn mechanistic relations. Recall that mechanistic causal relations take the following form:

Mutation Type x Design Component $\rightarrow$ Mitigation of Symptoms

in which a design component is any type of "building block" that the evolutionary process arranges to construct a solution, a mutation type is any particular type of change that may be made to or "with" a design component, and the mitigated symptoms are the expected effect of applying the mutation to the design component.

Over the course of some predefined number of *learning generations*, every single application of a mutation operator to an individual and change that results is observed and recorded. After the predefined number of learning generations are complete, the collected data is analyzed and consolidated into a set of mechanistic relations as follows. For every design component c and mutation type m, all observed instances of m being applied to c are analyzed. Each of these instances results in some change in the performance of the individual, which may or may not mitigate some performance symptom. Analysis is performed to determine, on average, what performance symptom s is mitigated by applying m to c, and the following mechanistic causal relation is established:

$$m \times c \rightarrow s$$

During the rest of the evolutionary process these acquired mechanistic causal relations are used as the basis for influencing how mutation is applied to individuals. Evolutionary systems that acquire or "learn" about causal relations by observing the evolutionary process, and then use those acquired relations to causally-guide the remainder of that evolutionary process are referred to as *learned casually-guided evolutionary systems*.

6.3 Evaluation on Synthetic Social Network Design Problem

To evaluate these ideas, the methods for learned causally-guided evolution described above are applied to the task of designing synthetic social

networks that match the distribution of clustering coefficients of various target real-world social networks. This is the same task that was studied in Chapter 5. However in the study presented here, the mechanistic causal relations are not defined a priori by human experts, and must instead be acquired during the evolutionary process. In this sense, the work presented here "pretends" that the causal relation are not available even though, as evident in the previous chapter, they are. Conducting this initial evaluation on an application problem for which causal relations are available and have been successfully used as the basis for causal guidance was a strategic decision. It was important to conduct this initial evaluation on such an application problem in order to be sure that there are in fact causal relations to be acquired.

The goal of these experiments is to evaluate the feasibility of applying causally-guided evolutionary computation in situations where causal knowledge is not available a priori. Is it possible to acquire causal relations during the evolutionary process? Will this causal knowledge be sufficiently accurate to effectively guide the evolutionary process? How well do these methods perform relative to a causally-guided evolutionary system in which a prior causal knowledge is available? All of these questions are explored through this experimental evaluation.

6.4 Experimental Methods

The methods for learned causally-guided evolution described above were evaluated when applied to the task of designing synthetic social networks that match the clustering of target real-world networks, under the hypothetical situation in which no causal knowledge is available a priori.

Recall that mechanistic causal relations describe the cause-effect relationship between the application of a particular type of mutation to particular design components, and the performance symptoms that are expected to be mitigated as a result. As in Chapter 5, the design components here are nodes with particular clustering characteristics. The notation n(C) is used to represent a node with clustering of C. Alternatively, the notation $\mathrm{n}\left\{{e \atop k}\right\}$ is used to represent a node with clustering $\left\{{e \atop k}\right\}$. The mutation types are $\mathrm{ADD_E}$, $\mathrm{DEL_E}$, $\mathrm{ADD_T}$, and $\mathrm{DEL_T}$. Mechanistic causal relations were defined in the previous chapter as follows:

Mutation m x Node n(C) $\rightarrow$ Mitigation of insufficient(b)

where b=Bin(Coeff(ExpMtoN(m,n(C)))). Note that the core piece of causal knowledge here is actually the ExpMtoN(m,n(C)) function, which captures the clustering measure that is expected to result when a mutation m is applied to a node with clustering of C. It is this key piece of knowledge that allows for the establishment of the mechanistic causal relations described above. In the previous chapter, the ExpMtoN function was defined a priori, as follows:

$$\text{ExpMtoN}(\text{ADD}_\text{E},\ \text{n}\left\{{e \atop k}\right\}) = \text{n}\left\{{e \atop k+1}\right\}$$

$$\text{ExpMtoN}(\text{DEL}_\text{E},\ \text{n}\left\{{e \atop k}\right\}) = \text{n}\left\{{\text{round}(e-\frac{2e}{k}) \atop k-1}\right\}$$

$$\text{ExpMtoN}(\text{ADD}_\text{T},\ \text{n}\left\{{e \atop k}\right\}) = \text{n}\left\{{\min(e+1, k*(k-1)*\frac{1}{2}) \atop k}\right\}$$

$$\text{ExpMtoN}(\text{DEL}_\text{T},\ \text{n}\left\{{e \atop k}\right\}) = \text{n}\left\{{\max(e-1, 0) \atop k}\right\}$$

In this chapter, the above definition of the ExpMtoN function is referred to as the "a priori ExpMtoN function." In contrast, in this chapter this definition of ExpMtoN is not available a prioir, and instead values of the function must be learned through observation of the evolutionary process. As values of the ExpMtoN function are learned, they are used to construct the mechanistic causal relations described above.

To learn values of the ExpMtoN function, the learned causally-guided evolutionary system analyzes every instance of mutations being applied to a node and the change in the clustering measure of that node that results. For each distinct mutation type and clustering measure, the various outcomes are gathered and averaged. For example, there may be five instances in which the ADD_E mutation type is applied to node with clustering of $\left\{{4 \atop 4}\right\}$, resulting in clustering measure of $\left\{{4 \atop 5}\right\}$, $\left\{{4 \atop 5}\right\}$, $\left\{{4 \atop 5}\right\}$, $\left\{{6 \atop 5}\right\}$, and $\left\{{8 \atop 5}\right\}$. The average outcome ($\left\{{5.2 \atop 4}\right\}$) is calculated and rounded ($\left\{{5 \atop 4}\right\}$), thereby establishing a single value of ExpMtoN function:

$$\text{ExpMtoN}(\text{ADD}_\text{E},\ \left\{{4 \atop 4}\right\}) = \left\{{5 \atop 4}\right\}$$

Now that this value of the ExpNtoM value is known, the following mechanistic causal relation is established:

ADD_E x n$\left\{{4 \atop 4}\right\}$ $\rightarrow$ Mitigation of insufficient(8)

based on the fact that Bin(Coeff(ExpMtoN(ADD_E, $\left\{{4 \atop 4}\right\}$))) can now be evaluated, since we now know this particular ExpMtoN value. In this manner, the learned causally-guided evolutionary system learns the values of the ExpMtoN function rather than having it specified a priori, and each learned value is used to establish a mechanistic causal relation.

A learned causally-guided evolutionary system that uses these methods was developed and is referred to as Causal-H5L, where H5 indicates that 5-step "how" guidance is used, and the L indicates that learning is being used to acquire mechanistic causal relations. Three hundred trials of this evolutionary system were applied to the same four target data sets as in Chapter 5, yielding 1200 total trials. Through a small number of trial-and-error evaluations, it was found that 100 learning generations is effective for the Attacks and Terrorists data sets, while 500 is effective for the Academy and Gang data sets. All other parameter values are the same as in Chapter 5. The performance of these trials was compared to those of the Control system and the Causal-H5 system from Chapter 5. The Control system serves as baseline in which causal knowledge is neither available nor acquired, and Causal-H5 serves as a comparable system in which causal knowledge is available a prior and therefore does not need to be acquired.

The mutations that were observed by the Causal-H5L system, and the ExpMtoN values that were acquired as a result are analyzed, and the

degree of their agreement with the a priori ExpMtoN function from Chapter 5 are assessed. This analysis is performed in order to answer two similar but distinct questions: 1) To what extent does the observed effect of mutations match what is expected according to the a prior knowledge? and; 2) To what extent do the ExpMtoN values that are acquired by the CAUSAL-H5L system through observation of these mutations match the a priori ExpMtoN function from Chapter 5? The overall performance of the CAUSAL-H5L system was assessed using the same analyses as in the previous chapter (fraction of each evolutionary systems trials to reach various fitness levels), allowing for direct comparison with all evolutionary systems presented previously. Z-tests with 99% confidence levels were used to evaluate the statistical significance of the reported results.

6.5 Results

Across 300 trials, the learned causally-guided evolutionary systems observed an average of 50,800 mutations when applied to the Academy and Gang data sets, and 10,160 when applied to the Attacks and Terrorist data sets. The divergent numbers are due to the different numbers of learning generations that were used on the different data sets (500 vs. 100). From the observed mutations, the expected value of the ExpMtoN(m,n(C)) was determined for numerous values of m and n(C). On average, 459, 369, 117, and 137 distinct ExpNtoM() values were acquired when applied to the Academy,

Gang, Attacks, and Terrorist data sets, respectively. In the remainder of this section, the effectiveness of the CAUSAL-H5L system compared to the CAUSAL-H5 and CONTROL system is examined, followed by an analysis of the ExpNtoM() values that were acquired by the CAUSAL-H5L system.

Figure 6.1 illustrates the fraction of trials from the CONTROL, CAUSAL-H5L, and CAUSAL-H5 systems that reached various error measures by generation 5000 when applied to the Academy target data set. As seen in the figure, while the CAUSAL-H5L system was not as effective as the CAUSAL-H5 system, it did outperform the CONTROL system in which no causal knowledge was used. For example, 53% of the CAUSAL-H5L trials reached an error measure of 1.50, compared to only 19% of the CONTROL trials. The difference in performance was statistically significant to a 99% confidence level at error measures between 1.75 and 1.00. While a higher fraction of CAUSAL-H5 than CAUSAL-H5L trials reached various fitness levels, this difference was small and never statistically significant.

The fraction of trials that reached various fitness levels when applied to the Attacks data set are illustrated in Figure 6.2. The difference in performance between the CAUSAL-H5L and CONTROL systems are quite dramatic. Approximately 53% of CAUSAL-H5L trials reached an error measure of 3.00, compared to only 19% of CONTROL trials. The difference in performance between CAUSAL-H5 and CAUSAL-H5L trials was not statistically significant at any error measure.

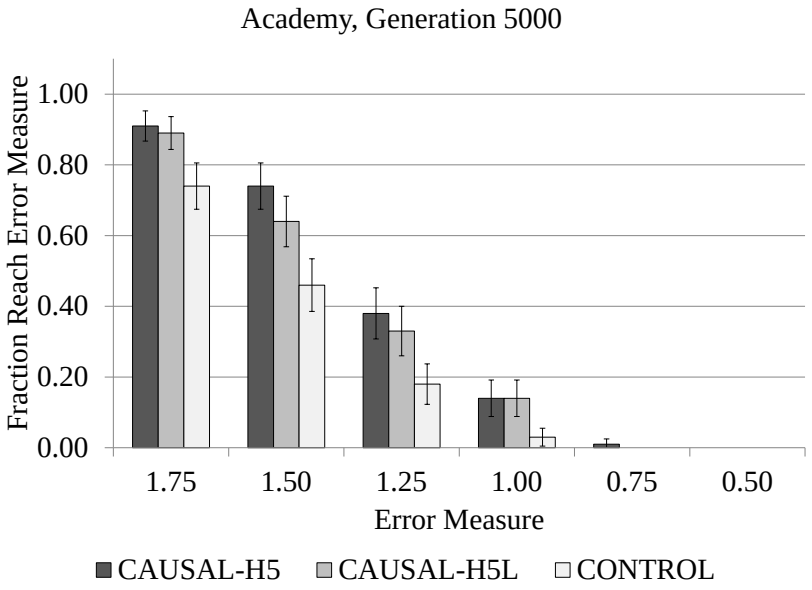

Figure 6.1: The fraction of trials of the CONTROL, CAUSAL-H5, CAUSAL-H5L systems to reach various error measures by generation 5000, when applied to the Academy target social network. Error bars indicate 99% confidence intervals.

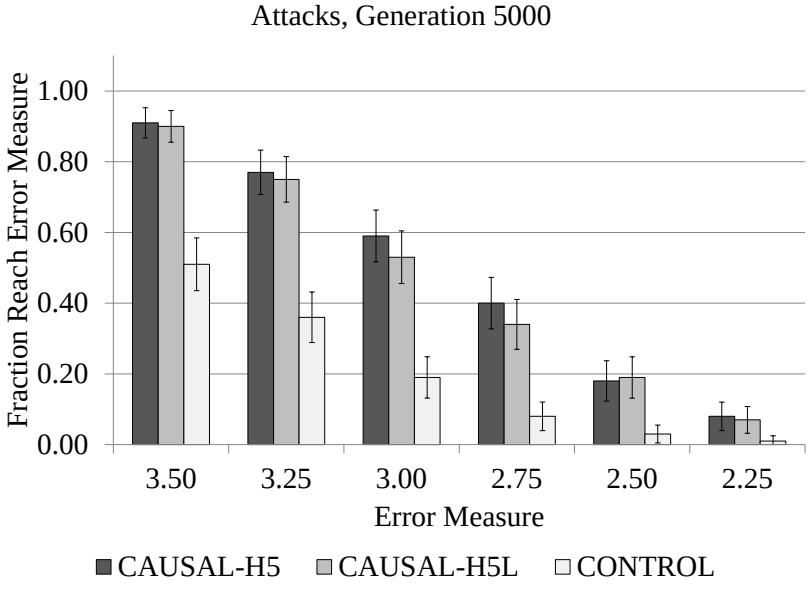

Figure 6.2: The fraction of trials of the CONTROL, CAUSAL-H5, CAUSAL-H5L systems to reach various error measures by generation 5000, when applied to the Attacks target social network. Error bars indicate 99% confidence intervals.

Figure 6.3 shows the fraction of trials from the evolutionary systems that reached various error levels when applied to the Gang data set. For higher error measures (above 0.80, not listed) there is little difference in performance between the systems. However, at error measures between 0.75 and 0.45, there is a clear difference between the CAUSAL-H5L and CONTROL systems. 87% of the CAUSAL-H5L trials reached an error measure of 0.60, compared to only 63% of the CONTROL trials. At an error measure of 0.45, the difference was 38% of CAUSAL-H5L trials compared to 16% of CONTROL trials. The difference in performance between CONTROL and CAUSAL-H5L was statistically significant to a 99% confidence level at all examined error measures. In contrast, there was no statistically significant difference between CAUSAL-H5L and CAUSAL-H5 systems.

Figure 6.4 shows similar analysis when the evolutionary systems were applied to the Terrorist data sets. As seen in Figure 6.4, while the CAUSAL-H5L system outperformed the CONTROL system at all error measures, this difference was never statistically significant, nor was the difference in performance between the CAUSAL-H5 and CAUSAL-H5L systems.

Analysis of the mutations that were observed by the CAUSAL-H5L systems and the ExpMtoN values that were learned from them is best understood by examining each mutation type separately. Analysis of the ADD_T and DEL_T mutation types are trivial. Across all 1200 trials of the CAUSAL-H5L system (300 trials * 4 data sets), more than 5 million ADD_T and 7

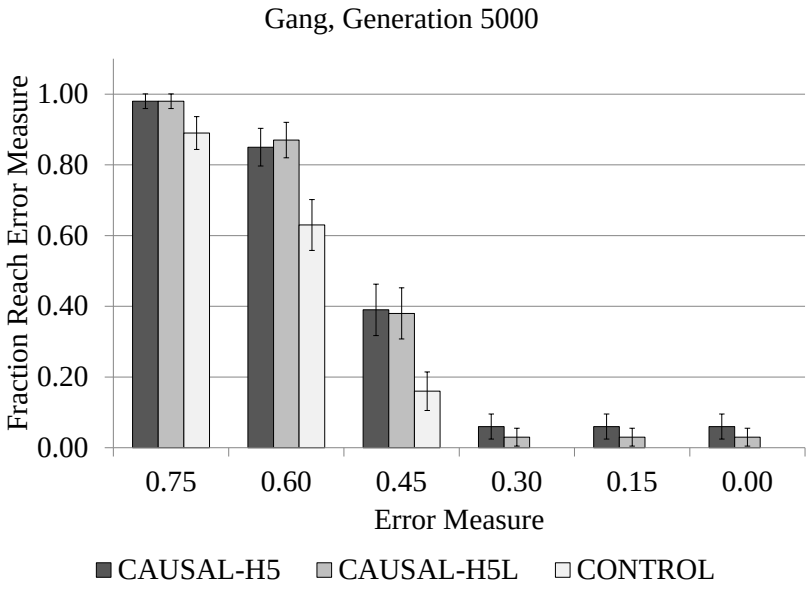

Figure 6.3: The fraction of trials of the CONTROL, CAUSAL-H5, CAUSAL-H5L systems to reach various error measures by generation 5000, when applied to the Gang target social network. Error bars indicate 99% confidence intervals.

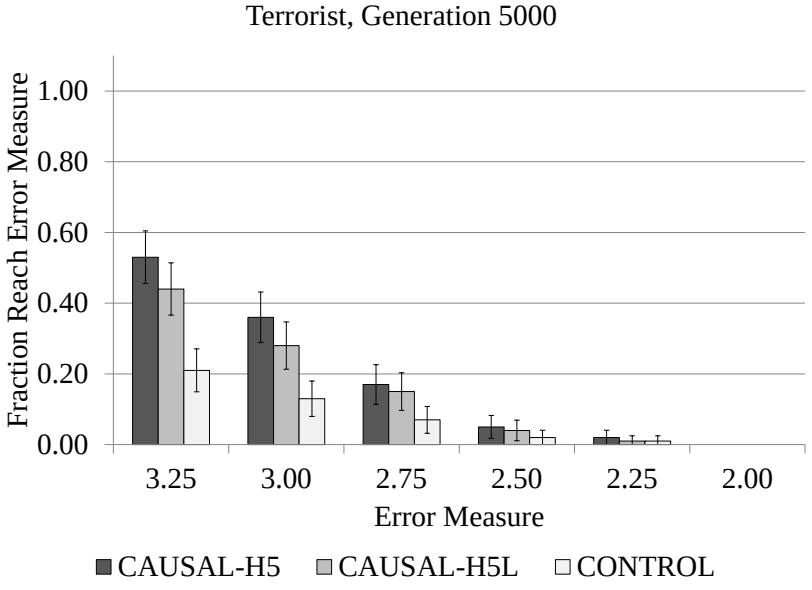

Figure 6.4: The fraction of trials of the CONTROL, CAUSAL-H5, CAUSAL-H5L systems to reach various error measures by generation 5000, when applied to the Terrorist target social network. Error bars indicate 99% confidence intervals.

million DEL_T mutations were observed. Every single instance of these mutations resulted in a clustering measure that matches what is expected according to the a priori ExpMtoN function. This is a trivial result, as adding or removing a triad from a node's open neighborhood definitively determines the resulting clustering of that node (i.e., there is no way for the a priori knowledge to be wrong). Accordingly, all of the ExpMtoN() values that the CAUSAL-H5L systems acquires based on these observed mutations match the a priori values exactly.

Table 6.1: Observed ADD_E Mutations, Acquired Knowledge, and Agreement with A Priori Knowledge

	Academy	Gang	Attacks	Terrorist
# Observed ADD_E Mutations	17248	17193	4243	4440
Fraction Match A Priori	0.93	0.94	0.96	0.96
# Acquired ExpMtoN Values	118	96	32	38
Fraction Match A Priori	0.92	0.95	0.97	0.96
# Acquired ExpMtoN Values (30)	79	66	26	22
Fraction Match A Priori	0.99	1.00	0.99	1.00

Table 6.1 summarizes the analysis of ADD_E mutation type. In the vast majority of instances, the ADD_E mutation resulted in the mutated node having a clustering measure that exactly matches what is expected according to the a priori ExpMtoN function. However, in roughly 5% of instances, the resulting clustering measure differed from the a priori ExpMtoN function. These instances occur when an edge is added from the mutated node n to a non-neighbor that happens to already be connected to one of n's neighbors. In such a circumstance, the number of neighbors of n will increase by one

(as expected in the a priori definition) but the number of edges in n's open neighborhood will also increase by one (or more), which is not expected according to the a priori ExpMtoN function. Because of these instances, roughly 5% of the acquired ExpMtoN values do not match a priori values. This tends to occur when ExpMtoN values are acquired based on a very small number of observed instances of the ADD_E mutation being applied to a particular clustering type. For example, if only 2 instances of ADD_E being applied to a node with clustering of $\left\{{3 \atop 4}\right\}$ are observed and both of these instances result in an edge being added to a neighbor of the node, then the acquired value for ExpMtoN(ADD_E, $\left\{{3 \atop 4}\right\}$) will be $\left\{{4 \atop 5}\right\}$, rather than $\left\{{3 \atop 5}\right\}$ as expected according to the a priori ExpMtoN function. Among those acquired values of ExpMtoN that are based on at least 5 observations, over 99% match the a priori ExpMtoN function.

Figures 6.5 illustrates the effect of applying DEL_E mutations as observed by the CAUSAL-H5L system compared with what is expected according to the a priori ExpMtoN function. Specifically, each point in these figures represents the application of DEL_E mutations to nodes with a particular clustering measure. Each point is plotted on the horizontal axis based on how many edges the mutated node is expected to have in its open neighborhood (the e in $\left\{{e \atop k}\right\}$), on average, according to the a priori ExpMtoN function. Each point is plotted on the vertical axis based on how many edges the mutated node has in its open neighborhood, on average, as observed in all 300 trials of the CAUSAL-H5L systems. All data points that are based on less than 30

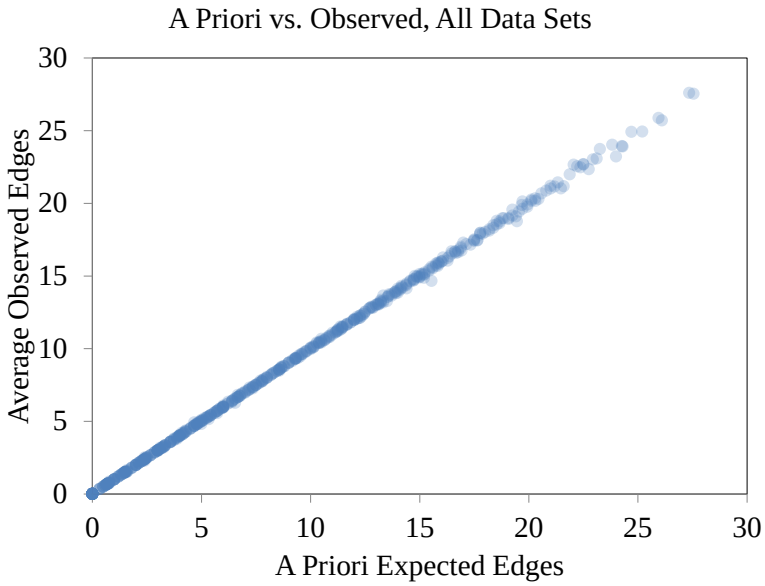

Figure 6.5: The observed average number of edges in a node's open neighborhood after DEL_E mutation has been applied (vertical axis), compared against the average number of edges that is expected according to a priori knowledge (horizontal axis). Each data point represents the application of DEL_E mutation to a node with a distinct clustering measure.

observed mutations are omitted from these figures (thereby removing spurious results based on a few data points). As can be seen in Figure 6.5, there is wide agreement between the expected effect of applying DEL_E mutations and the effect that is actually observed.

Table 6.2 summarizes the analysis of the DEL_E mutation types. The majority of instances of the DEL_E mutation resulted in the mutated node having a clustering measure that matches what is expected according to the a priori ExpMtoN function. However, significant fractions of these DEL_E mutations result in clustering measures that do not match a priori knowledge, including 24% of the observed DEL_E mutations on the Academy data set. Similar fractions of the acquired ExpMtoN values, which are learned through

Table 6.2: Observed DEL_E Mutations, Acquired Knowledge, and Agreement with A Priori Knowledge

	Academy	Gang	Attacks	Terrorist
# Observed DEL_E Mutations	14064	15514	3217	2911
Fraction Match A Priori	0.76	0.81	0.93	0.87
# Acquired ExpMtoN Values	117	95	31	36
Fraction Match A Priori	0.72	0.75	0.83	0.81
# Acquired ExpMtoN Values (30)	78	65	21	25
Fraction Match A Priori	0.84	0.86	0.92	0.90

analysis of the observed mutations, differ from the a priori ExpMtoN function. When ExpMtoN values that are based on less than 5 instances of an observed mutation are removed, large fractions of acquired ExpMtoN values that differ from the a priori values still remain, including 16% of ExpMtoN values on the Academy data set.

Further analysis reveals some potential reasons for these deviations. It is important to note that unlike DEL_T and ADD_T mutations, there are often many outcomes that can result from applying the DEL_E mutation to a node. For example, consider the scenario in which the DEL_E mutation is applied to a node n with three neighbors and one connection between those neighbors (a clustering of $\left\{{1 \atop 3}\right\}$). In a DEL_E mutation, a connection between the node n and one of its neighbors o is removed. If o had been party to the single connection in n's open neighborhood, then that single connection will be removed from n's open neighborhood as well, resulting in a clustering of $\left\{{0 \atop 2}\right\}$. Otherwise, the resulting clustering of n will be $\left\{{1 \atop 2}\right\}$. For this reason, it is expected that some instances of the DEL_E mutation being applied will

not result in a clustering measure that matches the *average* outcome, which is what the a priori ExpMtoN function specifies.

While this explains the existence of DEL_E mutations that result in clustering measures that do not match the a priori ExpMtoN function, it does not explain the differences between the ExpMtoN values acquired by the Causal-H5L system and the a priori ExpMtoN function. Indeed, if the observed average effect of applying DEL_E mutations closely matches the a prioir ExpMtoN function, as seen in Figure 6.5, why do so many acquired ExpMtoN values deviate from the a priori function? Closer examination of these differing acquired ExpMtoN values reveal that in all instances the deviation appears to be due to rounding error. For example, when applied to the Academy data set, the Causal-H5L system observed over 525,000 instances of DEL_E mutation being applied to a node with clustering of $\left\{{3 \atop 4}\right\}$. After the DEL_E mutation was applied, the nodes had an average of 1.4958 edges in their open neighborhoods. When constructing an ExpMtoN value from this data, the Causal-H5L system rounds this number, yielding the acquired ExpMtoN value: $\text{ExpMtoN}(\text{DEL}_\text{E}, \left\{{3 \atop 4}\right\}) = \left\{{1 \atop 3}\right\}$. In contrast, the expected number of edges according to the a priori knowledge is 1.5, which is rounded up to 2, resulting in the a priori ExpMtoN value: $\text{ExpMtoN}(\text{DEL}_\text{E}, \left\{{3 \atop 4}\right\}) = \left\{{2 \atop 3}\right\}$. Note that despite the fact that the observed and expected number of edges are almost exactly the same (1.50 compared to 1.4958), the use of rounding results in two distinct clustering measures for the acquired and a priori knowledge.

6.6 Discussion

With the exception of the Terrorist data set, the CAUSAL-H5L system clearly outperformed the CONTROL system when applied to all of the target data sets. This difference in performance was statistically significant to a 99% confidence level at most error measures across all three data sets. In some instances, the difference in performance was quite dramatic, with nearly twice as many CAUSAL-H5L trials as CONTROL trials reaching an error measure of 3.00 on the Attacks data set, 0.50 on the Gang data set, and 1.25 on the Academy data set. This clearly demonstrates the benefits of learned causally-guided evolution, and shows that it is possible to acquire and apply causal knowledge during the evolutionary process even when that knowledge is not available a priori.

Interestingly, the difference in performance between the CAUSAL-H5 and CAUSAL-H5L systems was not statistically significant for any error measure on any target data set. Indeed there are a number of instance in which the CAUSAL-H5L system actually performed better than the CAUSAL-H5 system, though again this was not statistically significant. This suggests that, at least for this particular application problem, there is little benefit to having mechanistic causal relations a prior as opposed to acquiring and applying them online. While this finding is too limited to necessarily have any general applicability across problem domains, it is still encouraging in that it suggests effective causal knowledge can be collected and applied during

an evolutionary process. Due to the expensive nature of specifying a priori causal relations and causal guidance, the ability of learned causally-guided evolution to successfully acquire and use causal relations is particularly attractive. By building upon these methods, it may be possible to further reduce the start-up costs of causally-guided evolution, which in turn makes these methods applicable to a wider range of application problems.

Analysis of the causal relations that are acquired by the CAUSAL-H5L systems reveal some interesting directions for future research. The a priori knowledge from Chapter 5 did not match the effects of *all* observed instances of the ADD_E or DEL_E mutation operator. In particular, large fractions of observed DEL_E mutations and the acquired relations that involve DEL_E mutations do not match a priori knowledge. This was shown to be due to two distinct factors. First, applying a DEL_E mutation can result in numerous different clustering outcomes, thus in many instances the result of applying DEL_E does not match the *average* result of applying DEL_E. Second, because there is a range of possible outcomes when DEL_E is applied, the average result of clustering can involve a non-integer number of edges in the open neighborhoods and rounding is used to produce clustering measures that are valid. This rounding can thus produce acquired causal relations that differ from the a priori ones, even when the observed effects of applying DEL_E are very close to what was expected according to the a priori ExpMtoN function. This raises an interesting question regarding the way in which mechanistic causal relations have been structured and used thus far: Given

that the application of a mutation type to a design component can result in a range of different outcomes, would it be more effective to construct and use causal relations that explicitly recognize that range of outcomes, rather than basing such relations on the average outcome, as has been done thus far? This remains an important area for future research. Interestingly, such an approach would require specification of more detail in the causal relations (a range of outcomes rather than an average outcome) which is potentially more burdensome on human users. In this context, the techniques that were investigated in this chapter for automatically acquiring causal relations may be even more valuable.

Chapter 7

Conclusion and Future Work

This chapter concludes this dissertation by summarizing the work and highlighting the contributions that were made to the study of causally-guided evolutionary computation. It also discusses the limitations of this work, as well as some possible directions for future work.

7.1 Summary and Contributions

The use of evolutionary computation methods as a design tool has been in part encouraged by the incredible innovativeness of biological evolutionary processes (e.g., the "invention" of optical lenses, sonar, pumps, valves, winged flight, neural computation, and many other things long before they were thought of by people (Bentley, 1999)), and in part by the growing number of successful applications discussed in Section 2.5. While evolutionary computation appears to be a promising tools for supporting the design process, in order for the evolutionary process to remain computationally tractable when applied to increasingly complex problems, new techniques must be developed that increase the efficiency and effectiveness with which evolutionary systems produce optimal designs.

This dissertation focuses on one such possible extension: causally-guided evolutionary computation. In contrast to conventional forms of evolutionary computation, in causally-guided evolutionary computation the application of genetic operators to an individual is driven in part by observing that individual's performance characteristics and performing causal reasoning based on explicit cause-effect relations in the domain. As such, genetic operators can be influenced to address the specific design issues that are present in the individual to which they are applied, and not are not blind and random as in conventional evolutionary methods. The central hypothesis that guided this research is that ultimately causal guidance will make the evolutionary process more effective by allowing it to explore a much larger number of good designs while still exploring novel solutions that initially appear unpromising, and more computationally efficient by decreasing the number of poorly fit individuals that do contribute useful information to the search process. To evaluate this hypothesis, various forms of causally-guided evolutionary systems were designed and developed, applied to a range of application problems, and their performance was evaluated against carefully matched control evolutionary systems that do not employ causal guidance but are otherwise identical.

The first major contribution of this dissertation is the demonstration, for the first time, of the feasibility of a casually-guided genetic operator. As presented in Chapter 3, the form of one type of causal knowledge that is used by causally-guided evolution was defined, in which diagnostic cause-

effect relationships between genotypic flaws and phenotypic problems in an individual may be described using a simple formalism. The generic form of causally-guided genetic operators was defined, consisting of three steps: 1) the performance characteristics of individuals is examined and phenotypic flaws are assessed; 2) using supplied causal knowledge, inferences are made about the relative likelihood of various genotypic flaws, and; 3) the application of the genetic operator is biased accordingly. In this initial study, one such genetic operator was defined as follows: *Causally-guided mutation operations are biased such that those parts of the genotype with higher relative likelihoods of being flawed are made more likely to be mutated.*

These ideas were evaluated by applying a causally-guided evolutionary system to the task of optimizing a set of connection weights in fixed-architecture neural network in order to produce a network that recognizes mirror symmetry of input patterns. The performance of this system was compared to a carefully matched control system in which causal guidance was not used but was otherwise identical. When applied to the 8-input symmetry problem, the causally-guided system discovered optimal network nearly twice as frequently as the control system. When applied to a ten-input symmetry problem, the causally-guided system found optimal networks 20% of the time, while the control system was unable to find an optimal network in any of the 100 trials. A small increase (less than 4%) in computational time per generation was found to be required by the causally-guided system as compared to the control. This marginal increase is far outweighed by the

fewer number of generations required to find optimal solutions. This work demonstrates for the first time the feasibility of using causal relations to guide the evolutionary process.

The second major contribution of this dissertation is the development of a second causally-guided genetic operator, demonstrating the feasibility of combining multiple causally-guided genetic operators together in a single evolutionary process, and validating their effectiveness when applied to a task in which causal knowledge is present but incomplete. In addition to causally-guided mutation, the causally-guided form of the crossover operator was defined as follows: *Causally-guided crossover operations are biased such that those parts of parent individuals' genotypes that have lower relative likelihoods of being flawed are made more likely to be combined together when creating offspring.* To evaluate the effectiveness of this second causally-guided genetic operator and the feasibility of applying multiple causally-guided operators together, these methods were applied to the real-world task of designing antenna arrays that meet pre-specified performance criteria. In contrast to neural network design task in the previous chapter, knowledge of cause-effect relations in the antenna design domain is less complete and comprehensive, and a central goal of this chapter was to evaluate the feasibility of causally-guided evolutionary computation when applied to such problems. This causal knowledge as well as the specific forms of causally-guided mutation and crossover were defined, and an evolutionary system that uses them

was developed, applied to the antenna design task, and its performance was compared to a control evolutionary system that does not use causal guidance.

It was found that, at various generations, the causal systems found the fittest antennas with significantly greater frequency than the control system. On average, the causally-guided systems also required significantly fewer generations to find antenna designs with various fitness scores. The causally-guided systems found optimal antenna designs much more frequently largely by avoiding specific sub-optimal designs. Interestingly, these sub-optimal designs were characterized by dipoles that are longer than optimal and have high VSWR values, factors that relate directly to the specific cause-effect relations that were employed. In each result discussed, it was found that the systems that use only causal mutation or causal crossover outperformed the control system, but that the system that uses both causal mutation and causal crossover performed even better, indicating that these causally-guided operators are synergistic/complementary. Additional experimentation in which design aspects of the fittest antenna were systematically varied and changes in performance were measures validated the causal knowledge that was employed, revealed some more complex cause-effect relationships in the domain, and suggested some possible extensions for the ways in which causal knowledge may be described.

The third major contribution of this dissertation is the development of an extended form of casually-guided evolution for design construction, in

which causal guidance is used to influence both where and how mutation is applied to individuals, as well as discovering some important guidelines for effective design of methods for causally-guiding where mutation is applied. Chapter 5 presents the extension of causally-guided evolution to a design construction problem, in contrast to the design optimization problems in Chapter 3 and 4. To this end, a new casually-guided mutation operator is defined in which both where and how mutation is applied to an individual are guided by causal reasoning. To influence how mutation is applied, a second type (mechanistic) of causal knowledge is defined that relates the application of mutation operators to genotypic components and the resulting change in phenotype, and a formalism for describing this type of causal relation is defined. This extended form of causally-guided mutation is defined as follows: *Causally-guided mutation operators are biased such that those components in the genotype with higher relative likelihoods of being flawed are made more likely to be mutated, AND are biased such that the mutations that are more likely to address those flaws are made more likely to be applied.*

These methods are evaluated by applying them to the task of designing synthetic social networks with characteristics that match real-world data sets. The specific diagnostic and mechanistic causal relations that are employed, as well as the application specific forms of causally-guiding where and how mutation occurs are described. Evolutionary systems that employ causally-guidance to influence where and how mutation occurs were developed and applied to the synthetic network design problem, as were systems

that only guide where mutation occurs, systems that only guide how mutation occurs, and control systems that do not use any causal guidance. It was found that across a number of target data sets and a number of goal error measures, the evolutionary system that uses causal guidance dramatically outperform the control system. For example, across the Academy, Attacks, Gang, and Terrorist data sets, there were error measures that none of the control system trials reached, compared to 66%, 94%, 100% and 47% of the causally-guided evolutionary system trials. Furthermore, it was discovered that employing causal guidance to influence both where and how mutation is applied clearly outperforms only influencing only one or the other. Lastly, the work in this chapter reveal some principles for proper design of causally-guiding how mutation is applied based on mechanistic relations. Specifically, it was discovered that when influencing how mutation is applied, it is important to consider not only the immediate effects of that mutation, but also the potential effects of subsequent mutations. Failing to do so can result in an evolutionary process that is short-sighted and tends to get stuck at local optima.

Finally, the fourth major contribution of this dissertation is the demonstration that causally-guided evolution may be applied in application domains where a priori causal knowledge is not available, and must instead be acquired and applied during the execution of the evolutionary process. Because such methods do not require as much "start-up" costs as causally-guided evolution (i.e., there is no need for the a priori specification of causal

relations, etc.,) they may even be desirable in application domains in which a priori knowledge is available. While the work presented in Chapters 3, 4, and 5 all rely upon causal knowledge that is supplied a priori, Chapter 6 examines situations in which no such knowledge is available. A general approach was outlined in which each application of a mutation to a genotypic component and the resulting change in phenotype is recorded and used to construct causal relations. These relations are then used to influence how mutation occurs, as described in Chapter 5. These methods were evaluated by applying them to same synthetic social network design problem as in Chapter 5, but in this case without the use of a priori causal knowledge. It was revealed that the learned causally-guided evolutionary system clearly outperformed a control evolutionary system in which causal knowledge was not supplied a priori, acquired, or used to guide the evolutionary process. Furthermore, it was determined that there was little difference in performance between the learned causally-guided system and the causally-guided system (from Chapter 5) in which causal knowledge was supplied a priori. This striking result clearly demonstrates the potential for acquiring and applying causal knowledge, particularly in application domains for which causal knowledge may not be available.

7.2 Limitations and Future Directions

There are a number of important limitations on the work presented in this dissertation, which also suggest some important directions for continued research.

The first limitation is the formalisms by which causal knowledge may be described by human experts. Using the current formalisms, domain experts may indicate the existence of causal relationships in the domain, but there is no way for them to describe the strength/conditional probability of those relationships. This lack of detail may obscure a great deal of information about cause-effect dynamics in some domains, which could be used to more effectively guide the evolutionary process. The causal strength (conditional probabilities) associated with causal relations have been shown to be a natural and intuitive way for humans to describe causal relationships (Peng and Reggia, 1990). This extension would be particularly valuable in situations where multiple causal relations have an impact on the same performance characteristics with varying degrees of influence, as it would provide a means to bias the search process by an appropriate amount depending on the strengths of various relations.

With this extension, the causal knowledge that is supplied by domain experts would implicitly define a causal Bayesian network, and this would permit the posterior probability of genotypic flaws given observed symptoms to be computed using standard Bayesian inference methods. These more

principled methods may produce more accurate measures of the relative likelihoods of flaws than the methods I've used, potentially resulting in a more effective evolutionary process. Further, using a causal Bayesian network would allow not only for the expression of noisy-OR relationships between flaws and symptoms, but also for noisy-AND relationships or even the conditional probability tables of symptom nodes, allowing domain experts to describe more complex causal relationships, such as when two genetic flaws together cause a phenotypic symptom that they would not cause independently. These principled methods for performing causal inference also support reasoning about multi-disorder explanations, in situations where there are many-to-many relationships between flaws and symptoms, a complex situation that is not examined in this dissertation.

There are considerable start-up costs associated with using causally-guided evolutionary computation, including defining the design flaws, performance symptoms, mutation types, causal relations, and the specific forms of causally-guided operators. While much of this work is application specific, an important area for future research is to explore methods for facilitating or even automating this process. For example, numerous evolutionary computation software packages exist in which developers can select from a set preprogrammed representation types, mutation operators, crossover operators, fitness functions, etc., in order to rapidly assemble an evolutionary system for the problem they wish to solve. It should be possible to develop similar software frameworks that can be used to rapidly assemble causally-guided

evolutionary systems to solve problems. When coupled with the ability to automatically acquire causal relations, such a software framework could greatly enhance the usability of causally-guided methods.

Aside from the work presented in Chapter 6 (in which causal relations are learned), a limitation of the causally-guided systems presented in this dissertation is their reliance on the ability of human experts to supply accurate information about causal relationships in the problem domain. In many applications, it may be difficult for domain experts to supply anything more than very limited information about causal relationships. Furthermore, it is possible that the supplied causal knowledge may even be inaccurate. In such circumstances, far from helping to guide the evolutionary process, causal reasoning may in fact influence the evolutionary process away from fruitful areas of the search space. While such an issue would require extensive study to clarify, I conducted a few preliminary simulations in which intentionally incorrect causal knowledge was supplied to the evolutionary process. Specifically, the causal relationship (Number of Dipoles $\rightarrow$ High VSWR), which is known by domain experts to be incorrect, was used. Indeed, as can be seen in Figure 4.10, the number of dipoles in an antenna appears to have very little effect on the VSWR of the antenna. Fifty trials of an evolutionary system, using both causally-guided mutation and causally-guided crossover while supplied with this erroneous causal knowledge, were applied to solve the antenna design problem. In 1000 generations, only 20 out of the 50 trials resulted in the discovery of an optimal antenna design. This 40% success rate

is significantly lower than the 60% success rate of the CONTROL system, and dramatically inferior to the 99% success rate enjoyed by the $\text{CAUSAL}_{\text{CM}}$ system.

These results clearly demonstrate the potential for incorrect causal knowledge to negatively impact the evolutionary process. Situations in which supplied causal knowledge is only valid in part of the solutions space could present similar problems. The results presented above suggest that when the causally-guided evolutionary process moves into areas of the solution space for which the supplied causal knowledge is invalid, the search process may be misled resulting in a negative impact on performance. An important area for future research is to examine this issue further in order to better understand the sensitivity of causally-guided evolution to incorrect knowledge.

The situation described above, in which supplied causal knowledge is incorrect, as well as the situation in which causal knowledge is simply not available a priori both suggest the need for automated methods to acquire and/or revise causal knowledge. While the work in Chapter 6 explores the feasibility of acquiring causal knowledge during the evolutionary process, it is quite limited in that it only addresses one type of causal knowledge and assumes a very specific form of that knowledge. More general purpose and powerful techniques for acquiring and refining causal knowledge through observation of the evolutionary process should be explored in order to overcome situations in which causal knowledge is unavailable or incorrect. Extending

these methods so that causal knowledge is represented as a Bayesian network, as described above, would also allow for the use of existing statistical relational learning methods to revise and improve network structure and values through observation of the evolutionary process. Additionally, these methods could be used to identify causal relationships that were previously unknown by the domain expert.

As discussed in Chapter 2, recent interest in evolutionary computation has been driven in part by the demonstrated capability of evolutionary systems to produce "innovative" designs that are qualitatively different than previously encountered solutions. In this context, an important area of future research into causally-guided evolutionary computation is to evaluate what impact causal guidance has on the discovery of truly novel solutions by the evolutionary process. In this dissertation, I have conducted some limited analysis of the types of solutions produced by the various evolutionary processes. However, a more rigorous evaluation should be performed in application domains where creativity is at a premium, such as the evolutionary design of art, music, etc. Does causally-guided evolution constrain the search process, preventing the formulation of truly novel solutions? Or does it actually facilitate novel design by focusing the search process on more fruitful areas of the search space?

There has also been much interest in recent years in the use of "developmental representations" in evolutionary computation. In these approaches,

rather than evolving a solution to a problem directly, a set of instructions or commands for creating a solution to a problem or created. For example, rather than evolving a neural structure directly using an adjacency matrix, a set of rules or a cell-growing grammar can be evolved which is then used to produce a neural architecture. All of the studies conducted in this dissertation employ a direct encoding. Accordingly, an important area of future research is to investigate how causally-guided evolutionary computation can be used with developmental representations and evaluating the impact of causal guidance on such an evolutionary system.

Lastly, as with any new technique, it is import to further evaluate the methods introduced here by applying causally-guided evolutionary computation to a wider range of challenging problems from a variety of domains, in order to assess the generality of the results presented here.

Bibliography

Abboud, K. and M. Schoenauer (2002). Surrogate deterministic mutation: Preliminary results. In *Artificial Evolution*, pp. 104–116. Springer.

Aleti, A., I. Moser, and S. Mostaghim (2012). Adaptive range parameter control. In *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8. IEEE.

Altshuler, E. and D. Linden (1997). Wire antenna design using a genetic algorithm. *IEEE Antenna and Propagation Society Magazine 39*, 33–43.

Altshuler, E. E. and T. H. O'Donnell (2011). An electrically small multi-frequency genetic antenna immersed in a dielectric powder. *IEEE Antennas and Propagation Magazine 53*(5), 33–40.

Angeline, P. and J. Pollack (1993). Evolutionary module acquisition. In *Proceedings of the Second Annual Conference on Evolutionary Programming*, La Jolla, CA, pp. 154–163. Evolutionary Programming Society.

Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computation. In M. Palaniswami, Y. Attikiouzel, R. J. Marks, D. Fogel, and T. Fukuda (Eds.), *Computational Intelligence: A Dynamic System Perspective*, pp. 152–161. IEEE Press.

Angeline, P. J. (1996). Two self-adaptive crossover operators for genetic programming. In P. J. Angeline and K. E. Kinnear (Eds.), *Advances in Genetic Programming 2*, pp. 89–109. Cambridge, MA, USA: MIT Press.

Back, T. (1992). Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, pp. 263–271. MIT Press.

Bailey, A., M. Ventresca, and B. Ombuki-Berman (2012). Automatic generation of graph models for complex networks by genetic programming. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pp. 711–718.

Banzhaf, W., P. Nordin, R. Keller, and F. Francone (1998). *Genetic Programming*. Morgan Kaufman.

Barabasi, A. and R. Albert (1999). Emergence of scaling in random networks. *Science 286*(5439), 509–512.

Barrett, C., R. Beckman, M. Khan, V. Kumar, M. Marathe, P. Stretz, T. Dutta, and B. Lewis (2009). Generation and analysis of large synthetic social contact networks. In *Winter Simulation Conference*, WSC '09, pp. 1003–1014.

Bearman, P. and J. Moody (2004). Suicide and friendships among american adolescents. *American Journal of Public Health 94*(1), 89–95.

Becerra, R. and C. C. Coello (2005). A cultural algorithm for job-shop scheduling. In Y. Jin (Ed.), *Knowledge Incorporation in Evolutionary Computation*, pp. 37–56. Springer.

Bentley, P. J. (1999). An introduction to evolutionary design by computers. In P. J. Bentley (Ed.), *Evolutionary Design by Computers*, pp. 1–73. Morgan Kaufmann.

Bentley, P. J. and D. W. Corne (2002). An introduction to creative evolutionary systems. In P. J. Bentley and D. W. Corne (Eds.), *Creative Evolutionary Systems*, pp. 1–55. Morgan Kaufmann.

Biles, J. (2002). Genjam: Evolution of a jazz improvisor. In P. Bentley and D. Corne (Eds.), *Creative Evolutionary Systems*, pp. 165–187. Academic.

Burke, G. J. and A. J. Poggio (1981). Numerical electromagnetics code nec : Method of moments. Technical Report UCID18834, Lawrence Livermore Lab.

Byrne, J., M. Fenton, E. Hemberg, J. McDermott, M. ONeill, E. Shotton, and C. Nally (2011). Combining structural analysis and multi-objective criteria for evolutionary architectural design. In *Applications of Evolutionary Computation*, pp. 204–213. Springer.

Charniak, E. (1991). Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated. *AI Magazine 12*(4), 50–63.

Chen, C.-H., T.-K. Yao, C.-M. Kuo, and C.-Y. Chen (2012). Evolutionary design of constructive multilayer feedforward neural network. *Journal of Vibration and Control*.

C.M. Coleman, E. R. and J. Ross (2004). Investigations of simulated annealing, ant-colony optimization, and genetic algorithms for self-structuring antennas. *IEEE Trans Antennas and Propagation 52*(4), 1007–1014.

Couclelis, H. (2009). The abduction of geographic information science: transporting spatial reasoning to the realm of purpose and design. In *Spatial Information Theory*, pp. 342–356. Springer.

Dawkins, R. (1996). *The Blind Watchmaker*. W. W. Norton.

De Jong, K. (1988). Learning with genetic algorithms. *Machine Learning*, 121–138.

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

De Jong, K., W. Spears, and D. Gordon (1993). Using genetic algorithms for concept learning. *Machine Learning 13*, 161–188.

Drabowitch, S., A. Papiemik, and H. G. et al (1998). *Modern Antennas*. Chapman and Hall.

Du, J. and R. Rada (2012). Knowledge-guided genetic algorithm for financial forecasting. In *Computational Intelligence for Financial Engineering & Economics*, pp. 1–8. IEEE Press.

Dupuis, J.-F., Z. Fan, and E. Goodman (2013). Evolutionary design of discrete controllers for hybrid mechatronic systems. *International Journal of Systems Science*, 1–14.

Easley, D. and J. Kleinberg (2010). *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press.

Eiben, A. and Z. Ruttkay (1996). Self-adaptivity for constraint satisfaction: learning penalty functions. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 258–261.

Eiben, A. E., R. Hinterding, and Z. Michaelwicz (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation 3*, 124–141.

Elliot, R. (2003). *Antenna Theory and Design*. IEEE Press.

Erdos, P. and A. Renyi (1959). On random graphs. *Publicationes Mathematicae Debrecen 6*, 290–297.

Firat, A., S. Chatterjee, and M. Yilmaz (2007). Genetic clustering of social networks using random walks. *Computational Statistics & Data Analysis 51*(12), 6285–6294.

Fogel, D. (2000). What is evolutionary computation? *Spectrum, IEEE 37*(2), 26–28.

Fogel, D. B. (1991). *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*. Ginn Press.

Fogel, D. B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley.

Funes, P. and J. Pollack (1999). Computer evolution of buildable objects. In P. J. Bentley (Ed.), *Evolutionary Design by Computers*, pp. 387–404. Morgan Kaufmann.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Gomez, F. (2012). Scalable neuroevolution for reinforcement learning. In *Theory and Practice of Natural Computing*, pp. 27–29. Springer.

Grannis, R. (2009). *From the Ground Up: Translating Geography into Community through Neighbor Networks*. Princeton University Press.

Grannis, R. (2010). Six degrees of who cares?. *American Journal of Sociology 115*(4), 991–1017.

Gruau, F. and K. Quatramaran (2001). Cellular encoding for interactive evolutionary robotics. In M. Patel, V. Honavar, and K. Balakrishnan (Eds.), *Advances in the Evolutionary Synthesis of Intelligent Agents*, pp. 29–63. MIT Press.

Hancock, P. (1992). Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specifications. In *Combinations of Genetic Algorithms and Neural Networks. 1992 COGANN-92*, pp. 108–122.

Haupt, R. L. (1995). An introduction to genetic algorithms for electromagnetics. *Antennas and Propagation Magazine, IEEE 37*(2), 7–15.

Haupt, R. L. and D. Werner (2007). *Genetic Algorithms in Electromagnetics*. IEEE Press (Wiley-Interscience).

He, L. and N. Mort (2000). Hybrid genetic algorithms for telecommunications network back-up routeing. *BT Technology Journal 18*, 42–50.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Hornby, G. S., A. Globus, D. S. Linden, and J. D. Lohn (2006). Automated antenna design with evolutionary algorithms. In *AIAA Space*, pp. 19–21.

Hu, J., E. d. Goodman, S. Li, and R. Rosenberg (2008). Automated synthesis of mechanical vibration absorbers using genetic programming. *Artificial Intelligence for Engineering Design Analysis Manufacturing 22*(3), 207–217.

Iba, H. and H. de Garis (1996). Extended genetic programming with recombinative guidance. In P. J. Angeline and K. E. Kinnear (Eds.), *Advances in Genetic Programming 2*. Cambridge, MA, USA: MIT Press.

Ishibuchi, H., T. Yoshida, and T. Murata (2002). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation 7*, 204–223.

Jin, E. M., M. Girvan, and M. E. Newman (2001). Structure of growing social networks. *Physical review E 64*(4), 046132.

Jin, Y. (2004). *Knowledge incorporation in evolutionary computation*, Volume 167. Springer Verlag.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing 9*(1), 3–12.

Josephson, J. R. (1994). Conceptual analysis of abduction. In J. R. Josephson and S. G. Josephson (Eds.), *Abductive Inference: Computation, Philosophy, Technology*, Chapter 1, pp. 5–30. Cambridge University Press.

Julstrom, B. A. (1995). What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. In *Proceedings of the 6th International Conference on Genetic Algorithms*, San Francisco, CA, USA, pp. 81–87. Morgan Kaufmann Publishers Inc.

Jung, J.-Y. and J. Reggia (2006). Evolutionary design of neural network architectures using a descriptive encoding language. *IEEE Transactions on Evolutionary Computation 10*(6), 676–688.

Jung, J.-Y. and J. A. Reggia (2009). Evolving an autonomous agent for non-markovian reinforcement learning. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, New York, NY, USA, pp. 971–978. ACM.

Knowles, J. and D. Corne (2005). Memetic algorithms for multiobjective optimization: Issues, methods and prospects. In W. Hart, J. Smith, and N. Krasnogor (Eds.), *Recent Advances in Memetic Algorithms*, Volume 166 of *Studies in Fuzziness and Soft Computing*, pp. 313–352. Springer Berlin / Heidelberg.

Korb, K. and A. Nicholson (2004). *Bayesian Artificial Intelligence*. Chapman and Hall.

Koza, J. R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press.

Koza, J. R. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Norwell, MA, USA: Kluwer Academic Publishers.

Koza, J. R., F. H. B. III, D. Andre, M. A, and F. Dunlap (1997). Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation 1*, 109–128.

Krasnogor, N. and J. Smith (2005). A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation 9*(5), 474–488.

Leung, S. W., S. Y. Yuen, and C. K. Chow (2012). Parameter control system of evolutionary algorithm that is aided by the entire search history. *Applied Soft Computing*, 3063–3078.

Lipson, H. (2008). Evolutionary synthesis of kinematic mechanisms. *Artif. Intell. Eng. Des. Anal. Manuf. 22*(3), 195–205.

Lis, J. (1996). Parallel genetic algorithm with the dynamic control parameter. In *Proc. of IEEE International Conference on Evolutionary Computation*, pp. 324–329.

Lohn, J., W. Kraus, and D. Linden (2002). Evolutionary optimization of a quadrifilar helical antenna. In *IEEE Antennas and Propagation Society International Symposium*, Volume 3, pp. 814–817.

Lohn, J., D. Linden, G. Hornby, and W. Kraus (2004). Evolutionary design of an x-band antenna for nasa's space technology 5 mission. In *Antennas and Propagation Society International Symposium*, Volume 3, pp. 2313–2316. IEEE.

Lohn, J. D., G. S. Hornby, and D. S. Linden (2008). Human-competitive evolved antennas. *Artificial Intelligence for Engineering Design Analysis and Manufacturing 22*(3), 235–247.

Lohn, J. D., W. F. Kraus, D. S. Linden, and S. Colombano (2001). Evolutionary optimization of yagi-uda antennas. In *Proc. of the 4th International Conference on Evolvable Systems*, London, UK, pp. 236–243. Springer-Verlag.

Luo, Z., X. Chen, and K. Huang (2010). A novel electrically-small microstrip genetic antenna. *Journal of Electromagnetic Waves and Applications 24*(4), 513–520.

Mehrotra, K. (1997). *Elements of Artificial Neural Networks*. MIT Press.

Meyer-Nieberg, S. and H. G. Beyer (2007). Self-adaptation in evolutionary algorithms. In F. Lobo, C. Lima, and Z. Michaelwicz (Eds.), *Parameter Setting in Evolutionary Algorithms*, pp. 47–75. Springer.

Michel, O. (2001). Evolutionary neurogenesis applied to mobile robotics. In M. Patel, V. Honavar, and K. Balakrishnan (Eds.), *Advances in the Evolutionary Synthesis of Intelligent Agents*, pp. 185–213. MIT Press.

Miikkulainen, R. (2010). Neuroevolution. In *Encyclopedia of Machine Learning*, pp. 716–720. Springer.

Minsky, M. and S. Papert (1969). Perceptron: an introduction to computational geometry. *The MIT Press 19*, 88.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press.

Moscato, P. and C. Cotta (2010). A modern introduction to memetic algorithms. In *Handbook of Metaheuristics*, pp. 141–183. Springer.

Nau, D. S., J. A. Reggia, and P. Y. Wang (1983). Knowledge-based problem solving without production rules. In *Proceedings of the IEEE Trends and Applications Conference*, pp. 105–108. IEEE Press.

Neri, C. and Moscato (2011). *Handbook of Memetic Algorithms*. Springer.

Newman, M. E. (2009). Random graphs with clustering. *Physical review letters 103*(5), 058701.

Pan, Z. and J. A. Reggia (2010). Computational discovery of instructionless self-replicating structures in cellular automata. *Artificial Life 16*, 39–63.

Panduro, M. A., C. A. Brizuela, L. I. Balderas, and D. A. Acosta (2009). A comparison of genetic algorithms, particle swarm optimization and the differential evolution method for the design of scannable circular antenna arrays. *Progress In Electromagnetics Research B 13*, 171–186.

Patokorpi, E. (2009). What could abductive reasoning contribute to human computer interaction? a technology domestication view. *Psychology Journal 7*, 113–131.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Peng, Y. and J. A. Reggia (1990). *Abductive inference models for diagnostic problem-solving*. New York, NY, USA: Springer-Verlag New York, Inc.

Poole, D. (1998). Learning, bayesian probability, graphical models, and abduction. In P. A. Flach and A. C. Kakas (Eds.), *Abduction and Induction: Essays on their Relation and Integration*, Chapter 10, pp. 153–168. Kluwer Academic Publishers.

Porto, V. W. (2000). Evolutionary programming. In T. Back, D. Fogel, and Z. Micahalewicz (Eds.), *Evolutionary Computation 1: Basic Algorithms and Operators*, pp. 89–100. Institute of Physics.

Rajo-Iglesias, E. and O. Quevedo-Teruel (2007). Linear array synthesis using an ant colony optimization based algorithm. *Antennas and Propagation Magazine 49*(2), 70–79.

Rasheed, K., X. Ni, and S. Vattam (2005). Methods for using surrogate models to speed up genetic algorithm optimization: Informed operators and genetic engineering. *Knowledge Incorporation in Evolutionary Computation*, 103.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzberg Verlag.

Reittu, H. and I. Norros (2012). Random graph models of communication network topologies. In *Unifying Themes in Complex Systems VII*, pp. 214–221. Springer.

Robinson, J. and Y. Rahmat-Samii (2004). Particle swarm optimization in electromagnetics. *IEEE Trans Antennas and Propagation 52*(2), 397–407.

Rocke, S. (2002). Eons of genetically evolved algorithmic images. In P. Bentley and D. Corne (Eds.), *Creative Evolutionary Systems*, pp. 229–265. Academic.

Rosenman, M. A. (1997). The generation of form using an evolutionary approach. In *Evolutionary Algorithms in Engineering Applications*, pp. 69–86. Springer-Verlag.

Rothenberg, R. (2002). From whole cloth: making up the terrorist network. *Connections 24*(3), 36–42.

Rubrecht, S., E. Singla, V. Padois, P. Bidaud, and M. De Broissia (2011). Evolutionary design of a robotic manipulator for a highly constrained environment. In *New Horizons in Evolutionary Robotics*, pp. 109–121. Springer.

Rudolph, G. (2000). Evolution strategies. In T. Back, D. Fogel, and Z. Micahalewicz (Eds.), *Evolutionary Computation 1: Basic Algorithms and Operators*, pp. 81–87. Institute of Physics.

Rumelhart, D. (1987). *Parallel Distributed Processing: Volume 1*. MIT Press.

Rumelhart, D. E., G. E. Hintont, and R. J. Williams (1986). Learning representations by back-propagating errors. *Nature 323*(6088), 533–536.

Sala, A., L. Cao, C. Wilson, R. Zablit, H. Zheng, , and B. Zhao (2010). Measurement-calibrated graph models for social network experiments. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 861–870. ACM.

Schaffer, J. D. and A. Morishima (1987). An adaptive crossover distribution mechanism for genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ, USA, pp. 36–40. Lawrence Erlbaum.

Schraudolph, N. N. and R. K. Belew (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning 9*(1), 9–21.

Schwefel, H. P. (1981). *Numerical Optimization of Computer Models*. Wiley.

Schwefel, H. P. (1995). *Evolution and Optimum Seeking*. John Wiley.

Setiean, L. (1998). *Practical Communication Antennas*. Prentice Hall.

Shaefer, C. G. (1987). The argot strategy: adaptive representation genetic optimizer technique. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, Hillsdale, NJ, USA, pp. 50–58. L. Erlbaum Associates Inc.

Shakarian, P. and V. Subrahmanian (2011). Region-based geospatial abduction. In *Geospatial Abduction*, pp. 57–92. Springer.

Siakavara, K. (2010). Novel fractal antenna arrays for satellite networks: Circular ring sierpinski carpet arrays optimized by genetic algorithms. *Progress In Electromagnetics Research 103*, 115–138.

Smith, A. E. and D. M. Tate (1993). Genetic optimization using a penalty function. In *Proceedings of the 5th International Conference on Genetic Algorithms*, San Francisco, CA, USA, pp. 499–505. Morgan Kaufmann Publishers Inc.

Smith, J. and T. Fogarty (1996). Self adaptation of mutation rates in a steady state genetic algorithm. In *Proc. IEEE Internatinal Conference on Evolutionary Computation*, pp. 318–323.

Spears, W. M. (1995). Adapting crossover in evolutionary algorithms. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pp. 367–384. MIT Press.

Spector, L. and J. Klein (2008). Machine invention of quantum computing circuits by means of genetic programming. *Artificial Intelligence for Engineering Design Analysis and Manufacturing 22*(3), 275–283.

Spirtes, P., C. Glymour, and R. Scheines (2000). *Causation, Prediction and Search*. MIT Press.

Stadelhofer, R., W. Banzhaf, and D. Suter (2008). Evolving blackbox quantum algorithms using genetic programming. *Artificial Intelligence for Engineering Design Analysis and Manufacturing 22*(3), 285–297.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Stonedahl, F., W. Rand, and U. Wilensky (2010). Evolving viral marketing strategies. In *Proc. of the 12th annual conference on Genetic and evolutionary computation*, pp. 1195–1202. ACM.

Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proc. of 8th Annual Conference of the Cognitive Science Society*, pp. 823–831. Lawrence Erlbaum Associates.

Teller, A. (1996). Evolving programmers: the co-evolution of intelligent recombination operators. In *Advances in genetic programming: volume 2*, pp. 45–68. Cambridge, MA, USA: MIT Press.

Tse, S.-M., Y. Liang, K.-S. Leung, K.-H. Lee, and T.-K. Mok (2007). A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. *37*(1), 84 –91.

Watts, D. and S. Strogatz (1998). Collective dynamics of small-world networks. *Nature 6684*(393), 440–442.

White, T. and F. Oppacher (1994). Adaptive crossover using automata. In *Proc. of the International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature*, London, UK, pp. 229–238. Springer-Verlag.

Whitley, D. (2000). Permutations. In T. Back, D. Fogel, and Z. Michalewicz (Eds.), *Evolutionary Computation 1*, pp. 139–150. IOP Press.

Whitley, D., K. Mathias, and P. Fitzhorn (1991). Delta coding: An iterative search strategy for genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 77–84. Morgan Kaufmann.

Whitley, L. D., V. S. Gordon, and K. E. Mathias (1994). Lamarckian evolution, the baldwin effect and function optimization. In *Proc. of the International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature*, London, UK, pp. 6–15. Springer-Verlag.

Yang, H., J. Shang, Y. Yang, and D. Dong (2010). An improved genetic optimization method for neural network. In *Proc. of International Conference on Natural Computation*, Volume 1, pp. 122–125. IEEE.

Yao, X. (1999). Evolving artificial neural networks. In *Proc. of the IEEE*, pp. 1423–1447.

Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research 33*(4), 452–473.

Zhao, B., P. Sen, and L. Getoor (2006). Event classification and relationship labeling in affiliation networks. In *Proc. of ICML Workshop on Statistical Network Analysis (SNA)*.