LEVERAGING HUMAN INSIGHTS BY COMBINING MULTI-OBJECTIVE OPTIMIZATION WITH INTERACTIVE EVOLUTION

THESIS

Joshua R. Christman, Second Lieutenant, USAF

AFIT-ENG-MS-15-M-060

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-15-M-060

LEVERAGING HUMAN INSIGHTS BY COMBINING MULTI-OBJECTIVE OPTIMIZATION WITH INTERACTIVE EVOLUTION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Joshua R. Christman, B.S.C.S.

Second Lieutenant, USAF

March 2015

AFIT-ENG-MS-15-M-060

LEVERAGING HUMAN INSIGHTS BY COMBINING MULTI-OBJECTIVE OPTIMIZATION WITH INTERACTIVE EVOLUTION

THESIS

Joshua R. Christman, B.S.C.S.
Second Lieutenant, USAF

Committee Membership:

Maj Brian G. Woolley, PhD
Chair

Dr. Gilbert L. Peterson
Member

Maj John M. Pecarina, PhD
Member

AFIT-ENG-MS-15-M-060

Abstract

Deceptive fitness landscapes are a growing concern for evolutionary computation. Recent work has shown that combining human insights with short-term evolution has a synergistic effect that accelerates the discovery of solutions. While humans provide rich insights, they fatigue easily. Previous work reduced the number of human evaluations by evolving a diverse set of candidates via intermittent searches for novelty. While successful at evolving solutions for a deceptive maze domain, this approach lacks the ability to measure what the human evaluator identifies as important. The key insight here is that multi-objective evolutionary algorithms foster diversity, serving as a surrogate for novelty, while measuring user preferences. This approach, called Pareto Optimality-Assisted Interactive Evolutionary Computation (POA-IEC), allows users to identify candidates that they feel are promising. Experimental results reveal that POA-IEC finds solutions in fewer evaluations than previous approaches, and that the non-dominated set is significantly more novel than the dominated set. In this way, POA-IEC simultaneously leverages human insights while quantifying their preferences.

Acknowledgements

First and foremost, I thank my Lord and Savior, Jesus Christ, that he has given me this opportunity to come to AFIT and learn. I am beyond blessed to be able to continue my education and I am so thankful.

Next, I thank my wife for her unending support. Without her, my life would be incredibly dull and I am thankful every day that I am blessed to have her in my life. I look forward to the many years ahead of us.

I would also like to thank my advisor for his enthusiasm and passion. It has been a joy to work with him on this endeavor (even the writing part). Without him, this work would have never been created.

Lastly, I would like to thank my instructors and classmates for everything they have done over the past 18 months. It has been a fun journey and I would do it again if given the chance.

Joshua R. Christman

Table of Contents

Page

List of Figures

LEVERAGING HUMAN INSIGHTS BY COMBINING
MULTI-OBJECTIVE OPTIMIZATION WITH INTERACTIVE EVOLUTION

I. Introduction

A recent dialog in Evolutionary Computation (EC) has begun to address how traditional objective-based evolutionary methodologies fail when presented with a *deceptive* problem domain. Traditionally, this problem is addressed through diversity maintenance techniques, namely speciation, fitness proportional selection, and multi-objective algorithms. The goal of such techniques is to promote *behavioral diversity*, though none of them reward unique behaviors explicitly. Instead, they attempt to replicate behavioral diversity through genotypic diversity approaches that maintain unique genomes. Speciation and fitness proportional selection each artificially maintain a specific number of different genotype categories in an effort to prevent non-fit, possibly crucial behaviors from being eliminated from the population [46]. Fitness diversity has also been applied to encourage behavioral diversity (multi-objective algorithms). Multi-Objective Evolutionary Algorithms (MOEAs) use multiple, possibly competing objective functions simultaneously, maintaining any non-dominated behaviors (i.e. any individual such that no other individual is more fit in every objective) [6]. In this way, MOEAs create a diversity of fitness characteristics that maintains sufficient behavior diversity. This thesis demonstrates, through the use of multi-objective algorithms and human-computer collaboration, that fitness diversity can in fact provide the necessary behavioral diversity to overcome a deceptive domain.

More recent work with deceptive domains has demonstrated the effectiveness of ignoring the overall objective and instead rewarding unique behaviors to create behav-

ioral diversity. This approach, called *novelty search* [30], focuses solely on uniqueness when ranking solutions, ignoring their objective fitness at a given task. By calculating the sparsity of solution behaviors, novelty search rewards individuals that express new, interesting behaviors that break from the norm. This approach does not require that solutions be adept at a task, only that they are novel. The key concept of novelty search is that it does not eliminate essential *stepping stones* that are crucial to evolutionary success. In other words, novelty provides a mechanism for rewarding behaviors that score poorly in terms of fitness in order to escape local optima. Novelty, with its inherent ability to avoid deception, was shown to consistently provide solutions in a deceptive maze-navigation domain [30, 32] that was designed to thwart a fitness-based search.

The deceptive maze, introduced by Lehman and Stanley [30, 32], is specifically designed to be have local optima that intuitive objective functions are unable to escape. The maze contains a cul-de-sac that approaches near the goal, providing a very high fitness score for an objective function that measures a solution's "endpoint distance to the goal." Such objective functions rarely escape the local optimum. The same work demonstrates, however, that novelty search in the same deceptive domain consistently finds solutions, effectively overcoming the deception. It is in this domain that the technique introduced by this thesis will show that MOEAs in combination with human-computer collaboration can provide behavioral diversity that is subjectively similar to that of novelty search.

Searching for behavioral novelty as a way of avoiding deception has proven to be effective in a wide variety of domains: Gomes et al. [16] applied novelty search to the evolution of robotic swarms, Naredo and Trujillo [35] applied novelty search to data clustering, as well as many other domains [12, 29, 36]. Novelty search has consistently demonstrated itself as an effective method of maintaining behavioral diversity. This

deception-avoiding quality of novelty search has enabled algorithms to solve complex problems with EC in a much shorter amount of time and with fewer computational resources than objective-based search.

Interestingly, another approach that is effective at finding solutions to difficult problems is Interactive Evolutionary Computation (IEC) [48]. Under IEC, humans guide the evolutionary process through selective breeding. Dawkins [7] introduced the concept of IEC via the *Biomorphs* application, a program that used human selections to guide the evolution of insect-like images. He was able to demonstrate that humans provide key insights in evolution of *creative* domains that are difficult to capture with objective functions. Additionally, he found that the human insights provided a key mechanism for escaping local optima, as human intuitions are not subject to the same fixed limitations as a predefined objective function. Applying this technique, Sims [42] used IEC to evolve two-dimensional images of plants and, more recently, IEC has been used to evolve drumbeats [17] and even harmonies [18] to existing songs. Such subjective tasks are exceedingly difficult to characterize objectively. IEC, however, provides a mechanism which allows a human user to guide evolutionary processes using intuition and subjectivity.

However, Takagi [48] demonstrated that the limiting factor in every IEC instance is human fatigue. He explains that a typical IEC session lasts only about 20 generations, which may not be long enough to make meaningful progress in an evolution. This roadblock has motivated research into fatigue reduction. A concept of distributing workload between many users has been particularly successful in reducing individual fatigue. Sims [43] created an interactive art exhibit where users were able to "vote" on two-dimensional images they liked. In this way, many users helped evolve interesting artwork over the course of a longer period of time, with no one individual becoming fatigued. Sims [45] later performed a similar experiment to evolve three-dimensional

sea creatures, again utilizing a many-user interactive exhibit to generate a variety of unique sea creatures. More recently, Draves [13] introduced a many-user screensaver IEC project called *Electric Sheep*. This project lets users vote on their favorite screensavers, using a collaborative system to evolve a variety of screensavers. Another collaborative IEC effort recently demonstrated the ability for a community of users to evolve unique images using the Picbreeder online service [39, 40].

While successful, the many-user approach to IEC has certain pitfalls that reduce its efficacy as, in certain complex problem domains, it becomes infeasible to distribute workload over many users. In these domains, a single user has a more focused concept of a final solution and is more effective at guiding to that particular solution than a community of users. Fatigue distribution cannot help in these situations, creating the need for different fatigue reduction techniques.

A method of reducing fatigue while limiting the interaction to a single user is a fusion of two approaches, combining short-term search with interactive computing. Schmidt and Lipson [38] first introduced the concept, modeling user selections to guess which individual the IEC user would select next, attempting to preempt the user with useful suggestions. Hornby and Bongard [19] improved on this idea by running a fully automated fitness-based evolutionary search in the background, with human intervention at certain points to guide overall evolution. This concept worked well, but the short-term search still exhibited the deceptive tendencies to which objective-based approaches fall prey. Most recently, Woolley and Stanley [51] showed that an automated novelty search could greatly reduce the human fatigue inherent in IEC by generating important stepping stones, a capability well-known in novelty search. This approach, called *novelty-assisted interactive evolutionary computation* (NA-IEC) uses short-term automated novelty search as a way to accelerate IEC and reduce fatigue by presenting the human evaluator with a diverse set of candidate

behaviors. This diversity of behaviors is key to enabling the human user to effectively select and make progress quickly. NA-IEC evolved neurocontrollers for robots in deceptive mazes [30, 32] in significantly fewer evaluations (and less time overall) than any previous methodology [3, 19, 30, 32].

With a main solution to the fatigue problem being an automated approach, one might suppose that eliminating the human entirely is the best search solution. However, Woolley and Stanley [50] showed that images evolved using the Picbreeder online service [39, 40] could not be re-evolved in an automated manner with the same algorithms, supporting the findings of Dawkins [7] and Sims [42] and indicating that the unique insights a human provides serve an important role with regard to what can be discovered in a solution space. It is important, then, to try and better model human selections in an attempt to predict the next step in evolution. Better predictive models will accelerate IEC, reducing fatigue and enhancing the capabilities of the system.

1.1 Motivation

The concept of *autonomy* (i.e. the next step beyond automation into behavioral independence) is of interest to the Air Force, receiving a significant quantity of focus from Air Force Research Labs (AFRL). Deceptive problem domains are many of the most interesting and perhaps useful that exist. This research is motivated around an attempt to sidestep deception in such domains in an attempt to evolve higher functioning autonomous systems. While the problem domain used in this research is not directly related to any useful systems, the concepts introduced by this thesis can be used in arbitrarily complex domains to synthesize autonomous behaviors.

1.2 Problem Statement

This thesis builds on the work of Woolley and Stanley [51] by attempting to replicate the diversity maintenance mechanism of NA-IEC (i.e. novelty search) with a multi-objective algorithm that inherently contains a measurable heuristic that can be used to build such a predictive model. The result is a new approach called *pareto optimality-assisted interactive evolutionary computation* (POA-IEC), which is compared against NA-IEC and pure novelty search in its ability to evolve neurocontrollers. Under POA-IEC, a short-term search based on the *non-dominated sorting genetic algorithm* (NSGA) [11] progresses evolution in the background, with the non-dominated set serving as the diversity mechanism. Intuitively, members of the non-dominated set are shown to be more novel than the remainder of the population. Since each member of the dominated set lies along similar fitness vectors as the non-dominated set, they have similar fitness diversity. As such, because the non-dominated set is more fit than the remainder of the population, they will lie further along the path to perfect fitness (i.e. closer to a true solution). As few individuals have progressed as far along the fitness path, the non-dominated set necessarily has a more sparse distribution than the remainder of the population (i.e. the non-dominated set is more novel).

1.3 Research Objectives

This research will aim to illustrate the efficacy of POA-IEC in a deceptive maze domain. The goal is to avoid deception through diversity maintenance via the mechanism of the Pareto Front. By comparing the performance of POA-IEC with NA-IEC, which significantly outperformed all previous methodologies in the deceptive maze domain, a comparative analysis can be made to demonstrate the strengths of POA-IEC. Specifically, the measures of performance used to evaluate POA-IEC are, from

most to least important, evaluation count, wall clock time, and artificial neural network topological complexity minimization. Additionally, a comparison of the novelty of the individuals presented to the user (i.e. the Pareto Front) and the novelty of the rest of the population will show the similarities between the fitness diversity of the Pareto Front and behavioral diversity.

1.4 Results Contributed

Intuitively, using a multi-objective model to emulate behavioral diversity would not generate the pivotal stepping stones as well as a novelty-assisted approach. However, POA-IEC surprisingly outperforms NA-IEC, pure novelty search, and an automated *multi-objective evolutionary algorithm* (MOEA) that uses the same fitness vector, finding solutions in significantly fewer evaluations than any previous method to date. This supports the idea that fitness diversity resembles behavioral diversity such that a multi-objective approach can generate appropriate stepping stones in a deceptive domain.

1.5 Thesis Overview

This thesis will, over the next 5 chapters, cover relevant background (Chapter II), introduce the POA-IEC Framework (Chapter III), describe the experiment and its setup (Chapter IV), show the results (Chapter V), and discuss what can be concluded from the results (Chapter VI). Finally, final remarks and possible future work are both included in Chapter VI.

II. Background

This chapter reviews the concepts behind Evolutionary Computing (EC), Artificial Neural Networks (ANN) and their applications to EC, non-objective and single-objective methods for Evolutionary Computing, and an overview of multi-objective evolutionary algorithms (MOEA), including a look at some of the more advanced MOEA techniques.

2.1 Evolutionary Computing

The field of *Evolutionary Computing* (EC) has been applied to a wide variety of applications, finding particular success in optimization problems [23]. They are adept at solving ill-defined problems with non-obvious optimization criteria and are widely accepted as solvers for difficult problems.

EC can be broadly divided into several closely related categories [1], the most popular being *genetic algorithms* and *evolutionary algorithms*. In general, most EC strategies work by a process that mimics natural evolution. By choosing a representation of a problem space, a solution can be *evolved* through the application of a variety of *evolutionary operators*, namely *mutation*, *recombination*, and *selection* [1]. The evolutionary operators all modify the *genotype* (i.e. the "genetics") of the solution, with the resulting *phenotype* (i.e. the expression of the genotype) representing the domain-specific behaviors [6]. The operators each affect the genotype in a specific way.

2.1.1 Mutation.

In biological systems, mutation is inherent in every day life. When a cell's DNA mutates, new and unexpected behaviors may be expressed. Whether the phenotypic

expression of the mutations is beneficial is unknown prior to the mutation. Similarly, mutation in EC causes new behaviors to emerge in a solution space. For example, in a maze domain, a mutation of a robot controller may cause the robot to spin in circles instead of drive in a straight line. This expression of the changed genotype may be less desirable than the original behavior, in which case the mutation was not beneficial. The stochastic nature of mutations allows for serendipitous discovery of beneficial behaviors, however, and is key to escaping local optima in any EC domain [6].

2.1.2 Recombination.

Similar to mutation, recombination directly imitates natural life. Recombination is simply combining the genetic code of two individuals into a new genotype. In EC, this is typically implemented using a crossover, where half of the genetic material from one individual is concatenated with the opposite half of the genetic material from another individual. The recombination operator allows for useful behaviors from one individual to be combined with useful behaviors in another individual. This allows for a "divide and conquer" approach in certain domains, where a successful solution need not evolve multiple useful behaviors by serendipitous mutation [6].

2.1.3 Selection.

Unlike mutation and recombination, which are completely unaware of the phenotype of the individual, the selection operator is purely based on how "fit" an individual is. In any problem domain, an evaluation of each individual's expressed behavior must be completed to assess the *fitness* of each individual. This value represents how capable an individual is at carrying out the goal of the domain (e.g. a robot who stands still would receive a low fitness in a race). There exist many specific selection algorithms, such as *fitness proportional selection*, *rank-based selection*, and *tournament*

selection, each of which uses the fitness metric in some way to fill a new population with individuals that can be mutated and recombined. This follows the general idea of "survival of the fittest", where only the successful individuals are allowed to pass on their genetics to the next generation [6].

2.1.4 Behavioral Diversity.

Many evolutionary algorithms require a diversity maintenance mechanism. Because the selection operator uses fitness to create the next generation of an EC instance, over time, the population will converge to a single, dominant behavior. This behavior may be the global or a local optimum. Unless mutation is able to overcome the deception of the local optimum, EC will stall because fitness will cause the population to be dominated by individuals with more fit, locally optimal behaviors. The problem of deception has spurred research into diversity maintenance mechanisms, to include *speciation* and *multi-objective algorithms*, which will be discussed later.

2.1.4.1 Speciation.

Speciation is a concept that is closely related to real situations. Consider the canine family—there are many species of canine such as dogs, wolves, coyotes, and so on. However, these species do not breed with each other, though they all belong to the canine family. Similarly, speciation in EC separates genotypes by various characteristics into species. By setting minimum selection rates by species, it is possible to maintain behaviors that may be currently unfit, but are crucial behaviors to achieving the global optimum. The overall goal of speciation is to guard innovation [46]. Speciation is, however, not true behavioral diversity in that it does not take the diversity of the phenotypic behaviors into account. Rather, it protects certain genotypes from being extinguished due to poor behaviors. This particular technique is a geno-

typic diversity mechanism, which may or may not correlate to phenotypic behavioral diversity. That is, several unique genotypes may express similar phenotypes.

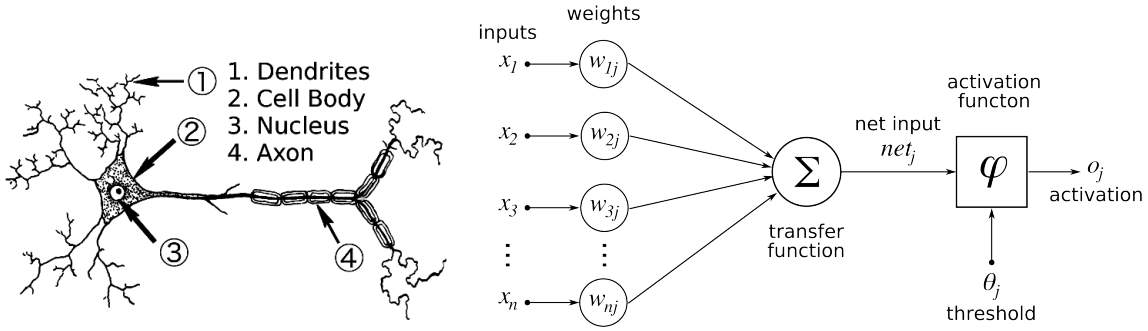
2.2 Artificial Neural Networks

Looking to nature as an inspiration for algorithms and techniques for solving complex problem has had much success [15]. Just as humans are particularly adept at information processing, Artificial Neural Networks (ANN), which simulate the function of biological neurons in a human brain, have been demonstrably successful in a wide variety of applications including pattern classification, clustering, optimization, and control [2, 20]. The human brain, being a particularly complex biological entity, has provided a mechanism to emulate in the form of a neuron.

2.2.1 Artificial and Biological Neurons.

Biological neurons (Figure 1a [41]) are cells that process input data, activating their outputs if the inputs match certain criteria. A neuron functions by receiving impulses along *dendrites* from other neurons. These impulses are processed in the cell body and a signal is transmitter through the *axon* to other neurons. *Synapses* exist at the ends of the dendrites and axons, which either *inhibit* or *enhance* the signal, depending on *learned* behavior [2]. The synapse is a chemically-based learning system that can modify the signals being transmitted from neuron to neuron. Since a synapse can be modified, it provides a memory mechanism to neurons. All of this behavior is mimicked by artificial neurons to create a learning system that can be broken down into atomic units.

An artificial neuron (Figure 1b [4]) mimics the behaviors of a biological neuron. A neuron has *inputs*, which are typically real numbers scaled to be between 0.0 and 1.0 or -1.0 and 1.0. Each input is scaled by a *weight*, which is usually a number in



(a) Biological Neuron (b) Artificial Neuron

Figure 1. Biological and Artificial Neurons [4, 41] - Artificial neurons are roughly analagous to biological neurons. Figure 1a shows a biological neuron and 1b shows an artificial neuron. In the diagram, the *inputs* of 1b correlate to the *axons* of other neurons in 1a, the *weights* correlate to the *synapses* at the end of the *dendrites*, the *transfer function* and *activation function* both happen in the *cell body* and *nucleus*, and the *output* (i.e. *activation*) correlates to the *axon*.

the same range as the inputs. A *transfer function* is applied to the *weight* $\times$ *input* calculations, typically a normalizing sum function that transforms all of the inputs to a meaningful range of numbers. The *activation function* is then applied to the result of the transfer function. The activation function is usually a sigmoid function, though linear, threshold, and Gaussian activation functions have applications in certain domains. Figure 1b shows a neuron with a threshold activation function (i.e. once the transfer function's output reaches the threshold θ_j, the output will toggle from a "low" state to a "high" state). The result of the activation function is the neuron's *output*, which feeds another neuron's input.

Artificial neuron structures are simple analogies that match well to a biological neuron. The parts of the two neurons are roughly comparable, as shown by Basheer and Hajmeer [2]: the *inputs* of the artificial neuron are mapped to the *axons* of other neurons, the *weights* multipliers of the artificial neuron correspond to the *synapses* that connect the axons of other neurons to the *dendrites* of the biological neuron, the *transfer* and *activation* functions happen in the *cell body* and *nucleus* of the biological neuron, and the *activation* (or output) of the artificial neuron correspond to the *axon*

of the biological neuron. In this way, an artificial neuron provides behaviors similar to those of a biological neuron.

2.2.2 Neural Networks.

When many neuron are chained together, they form a *neural network*. In biological neural networks, the axons of one neuron are connected to the dendrites of another neuron via an impulse-modifying synapse. An ANN has a similar construct by scaling the output of one neuron by a weight, and feeding the result to the next neuron. In an ANN, this is called a *connection* and the weight on the connection is called the *connection weight*. In a biological network, *learning* happens when the synapse is modified, changing the amount by which the impulse generated a neuron is enhanced or inhibited. In an ANN, the connection weights are modified to perform a similar function. In this manner, an ANN learns in much the same way as a biological neural network.

Another important aspect of an ANN, that will impact its effectiveness, is the *topology* of the network (i.e. the layout of the network) [14]. All ANNs have an "input layer" and an "output layer". In the input layer, each neuron has what is, essentially, pass-through functionality. That is, the neuron's input and output are the same. This allows an external agent to feed data directly into the neural network. The output layer is made of typical neurons and the outputs are generally readable by the same external agent as the feeds the input. The result is a "black box", where an agent feeds data to the ANN and reads the result of the calculations. This makes neural networks ideal for real-time operations, such as controllers for robots. In such a scenario, the inputs are the readings from various sensors (e.g. range finders, compasses, etc) and the outputs are mapped directly to motor controls. The

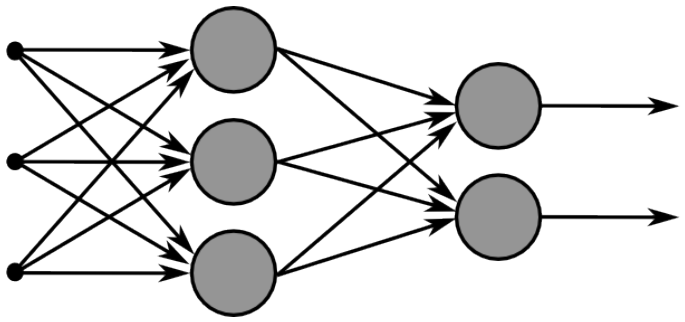


Figure 2. Multilayer Perceptrons [5] Multilayer perceptrons are a special class of multi-layer topological ANN. They have unidirectional connections that feed forward from the inputs, to the hidden layers, to the outputs. There are many other classes and modified versions of the multilayer perceptron that include features such as connections that feed backwards and loops.

ANN processes the data from the sensors and makes a real-time decision to affect the motors.

Many domains are too complex for a single layer (i.e. an input and output layer) topology to make effective decision boundaries. In this case a multi-layer topology must be used, as shown in Figure 2. In these architectures, at least one "hidden layer" processes the inputs and sends the results to the next hidden layer or to the output layer. This creates a situation in which arbitrarily complex decision boundaries can be drawn by creating a sufficiently complex topology.

There are many algorithms that are effective at learning through modification of connection weights. The back-propagation algorithm made multi-layer perceptrons (a specific class of multi-layer topological ANN) very popular in research. Back-propagation ANNs (BPANN) are the most popular ANN type in use due [20] to their effectiveness at converging to a solution. Another popular algorithm for ANN training is the Levenberg-Marquadt (LM) algorithm which, in many cases, is preferred to back propagation due to the faster convergence of LM [52]. One of the major drawbacks of the BPANN (and most ANN algorithms until 2003), is the static nature of its topology. In the human brain, connections between different neurons are constantly

being created and destroyed to suit the need of the brain. This aspect of the biological neural network was not imitated until much later.

2.3 NeuroEvolution of Augmenting Topologies

Introduced by Stanley and Miikkulainen [46, 47], the NeuroEvolution of Augmenting Topologies (NEAT) algorithm opened a new realm of possibilities for ANNs by creating a system that could evolve topologies in addition to connection weights. This eliminated the need to define an arbitrary, possibly overly complex, network topology prior to solving connection weights, instead allowing for the evolution of small, more capable networks through standard EC operators. Starting with a minimal network topology, NEAT complexifies the network via topological mutations that add and remove nodes and connections to the ANN, while traditional connection weight mutation occurs. In this way, non-intuitive, powerful network topologies can be created that perform tasks at a higher level than large, overly complex, arbitrary ANNs using traditional methods [46].

2.3.1 Neurocontroller Representation.

The neurocontrollers in this thesis are evolved using NEAT. A minimal topology used as a starting point for evolution in this thesis is shown in Figure 3b. The starting point is a fully connected ANN with no hidden nodes. That is, every input (all of which are sensory data) are connected to every output (which are read as direction and velocity controls). In this way, every sensor has an impact on the direction and velocity outputs. The manner in which the ANN evolves will affect the impact of each input on the control outputs. As the network complexifies, new behaviors are expressed that could not be expressed by simpler networks.

2.3.2 Evolutionary Frameworks.

The underlying implementation of NEAT used for this thesis is called *Another NEAT Java Implementation* (ANJI) [22]. ANJI is an open source project built on the *Java Genetic Algorithms Package* (JGAP) [33], another open source project that provides the underlying evolutionary mechanisms to ANJI. A version of ANJI and JGAP modified to support multi-objective and interactive evolution was created for this thesis.

2.4 Fitness Search

Fitness search is a central concept to EC that involves assigning a value to a solution that enumerates a solution's propensity for satisfying one or more predefined objectives. The more similar a solution to the objective, the higher the score assigned. The field of EC has traditionally used fitness functions as a driving force for improvement in a population's objective ability [1]. Single-objective functions can provide a good metric of a solution's comparative success to other solutions within the population. In this way, the fitness function points the way toward the goal such that individuals with "better" fitness are selected, propagating their genetics and driving the population as a whole towards better fitness.

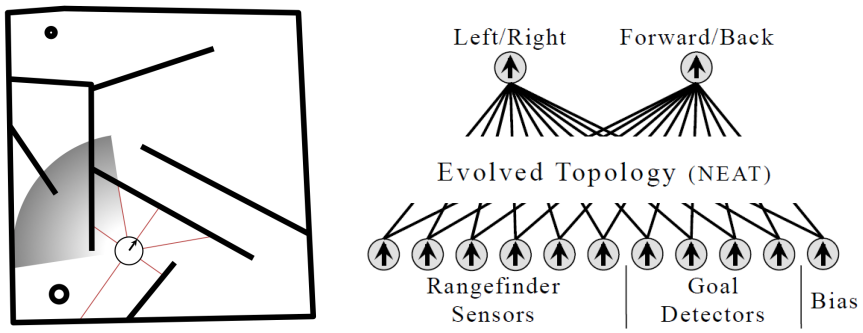
An excellent example of a fitness search was introduced by Sims [44] by evolving virtual creatures in a three dimensional world. He created a system in which swimming, walking, and jumping behaviors are evolved for different creature morphologies. That is, he evolved the shape and characteristics of the creatures to discover new creatures. Using traditional EC mutation, recombination, and selection, a wide variety of creature morphologies were evolved, including some structures that look strikingly similar to some natural behaviors. The success of this experiment, however, is overshadowed by more complex problem domains.

This classical approach to EC has two issues. One is that different solutions with similar fitness scores are conflated. The other is that objective-based search fails in deceptive landscapes, primarily due to a lack of behavioral diversity. In complex domains, which are inherently deceptive, it performs especially poorly as objective-based search is effectively a hill climbing algorithm that is unable to escape the local optima [32].

2.5 Deceptive Maze Domain

Lehman and Stanley [32] define a deceptive objective function as one that "will *deceive* search by actively pointing the wrong way." If the solution to a problem domain requires a path through certain intermediate points (i.e. *stepping stones*), then the objective function must reward these stepping stones appropriately in order for evolution to proceed along that path. However, in deceptive domains, the necessary stepping stones may have low fitness scores, thus causing evolution to avoid the critical path. These domains have inescapable local optima unless the algorithm can generate the important stepping stones to overcome hill-climber behaviors. As an example of such a situation, Lehman and Stanley [30, 32] introduced the deceptive maze domain as a metaphore for search (Figure 3a).

Here we can see that a simple objective function for this domain is to minimize Euclidean distance measured from the endpoint of the agent's path to the goal (i.e. assign a higher fitness to a low endpoint distance to goal). This fitness function, however, will spend a significant quantity of time exploring the cul-de-sac directly above the starting point, as it has a small distance to the goal. Such deceptive elements significantly hinder fitness-based search to the point that it may not successfully find a solution in a reasonable amount of time [30, 32]. If the critical stepping stones are known, it is possible to construct a fitness function that rewards according to reaching



(a) Robot Configuration (b) ANN Representation

Figure 3. Deceptive Maze and ANN Representation [51] - Figure 3a shows the deceptive maze domain introduced by Lehman and Stanley [30, 32] and used by Woolley and Stanley [51]. It is designed to have local optima such that a fitness based solution will struggle to evolve a solution. The maze is not an meaningful problem domain, but it servers as a metaphor for any problem that has local optima and deceptive structure. The image also shows the robot configuration used in this paper, as well as by Woolley and Stanley [51]. The robot includes six laser range-finders that detect the distance to walls and four pie-slice sensors that activate if the goal is in the quadrant indicated by that pie-slice. The image shows the active quadrant, indicating a direction to the goal. These sensors are interpreted as direct inputs to an ANN, shown in (b), that maps the sensory inputs to motor actions. Though the maze is not visible to the robot, the sensory input allows the evolution of a control policy to navigate the maze.

the stepping stones. In many problem domains, however, the precise stepping stones are unknown.

To address the issues inherent in deceptive domains, Lehman and Stanley [30, 32] presented a technique called novelty search, wherein the objective gradient is abandoned in favor of solution uniqueness as a heuristic for guiding evolution. This technique is described next.

2.6 Novelty Search

Searching a solution space without regard to an objective is less intuitive, but has been shown to outperform a fitness based search in deceptive domains [28, 31]. Novelty search assigns fitness based on how unique an individual's expressed behavior is with regard to the rest of the population and its ancestors. The more unique a solution, the higher the score assigned to the solution. This approach addresses the

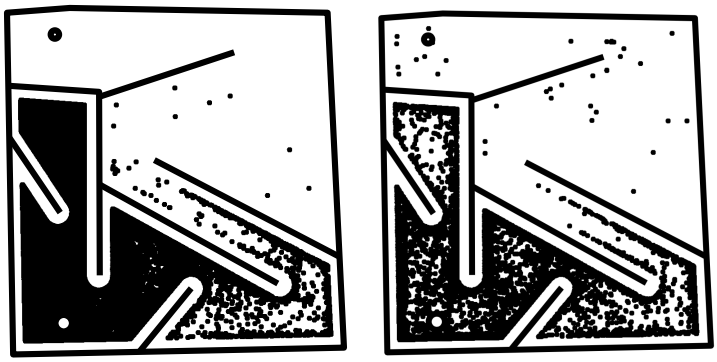
main downfall of fitness search by deliberately avoiding local optima through encouraging exploration rather than exploitation. The novelty of a solution is calculated by measuring the *sparseness* of solutions, defined as

$$\rho(x) = 1/k \sum_{i=0}^{k} dist(x, \mu_i) \tag{1}$$

where the sparseness of solution x is defined by the distance (in the behavior space) to its k nearest neighbors, such that μ_i is the ith nearest neighbor of x in the *behavior space* [50]. It is important to note that novelty is measured in the behavior space, as this will make each sparseness metric unique to the problem domain. In the maze domain in Figure 3a, the endpoint of a solution defines its behavior. By rewarding ending in unique (i.e. sparse) areas of the maze, the evolutionary pressure is to explore the maze. That is, if a robot ends its run in a different part of the maze than its predecessors, the relative density of endpoints in that location will be lower, meaning that robot has a more novel behavior. Similarly, as more individuals explore a certain area of the maze, the behavior that reaches that part of the maze becomes less fit as it becomes less unique, encouraging behaviors that reach new parts of the maze.

Woolley and Stanley [51] demonstrated that novelty search significantly outperforms fitness in the deceptive maze domain. Figure 4 shows the distribution of endpoints for a typical run of novelty and fitness-based searches. A dot is placed at each endpoint for a robot during the run, with the density of the dots showing the exploration pattern of the algorithms. A fitness-based search (Figure 4a) spends a majority of its evaluations searching the local optimum cul-de-sac, falling prey to the deceptive nature of the domain. A novelty based search (Figure 4b) explores the maze more evenly, escaping the deceptive cul-de-sac.

What is remarkable about novelty search is that while early solutions naturally achieve high novelty scores, such rudimentary behaviors quickly become common-



(a) Fitness (b) Novelty

Figure 4. Fitness and Novelty Endpoint Distributions [51] - The mazes show the distribution of endpoints for typical runs of a fitness based search and a novelty based search in the maze domain introduced by Lehman and Stanley [30, 32]. Each point represents an endpoint for one of the evolved control policies for a robot during the run. The distributions show that novelty search explores the maze more evenly than does fitness search. Work by Woolley and Stanley [51] validated that novelty search significantly outperforms fitness in this deceptive domain.

place, thus generating evolutionary pressures that lead to new behaviors. By rewarding only unique endpoints, evolutionary pressure toward novelty will eventually drive a solution that achieves the goal, even when it was not the objective of evolution.

2.6.1 Issues with Novelty Search.

In certain domains, particularly those that are unbounded, novelty search has been shown to fail [25]. The behavior space becomes infinitely large and the evolutionary pressure to do something unique is diffused into the space of trivial solutions. For example, if the outer boundaries of the deceptive maze domain were removed, the space of unique behaviors would become infinitely large. That is, a robot driving in any direction that ends somewhere new, would be rewarded for being novel, thus creating evolutionary pressure for useless behaviors. In these situations, it becomes necessary to supplement EC with more knowledge in order to better guide evolution.

Another issue with novelty search is the lack of solution quality measures. Other than the idea that a solution is unique, nothing else is measured about the solutions.

While useful in EC, it is not possible to infer more information about a problem domain by examining the novelty scores of the individuals. Another method must be used if we are to learn about a problem domain by examining the EC metrics.

2.7 Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms (MOEAs) are another method for avoiding deception [26]. They are based around the concept of looking at multiple (i.e. more than one), possibly competing, objectives simultaneously in a problem domain. A main motivation for this is that, even when a domain is deceptive, it is more difficult to deceive multiple objectives that just one. While it is possible to construct domains and constraints that cause deception for MOEAs [8], it is more difficult than it is for a single objective. Perhaps this is because individual solutions have many ways to distinguish themselves.

2.7.1 Aggregate Multi-Objective Functions.

MOEAs should not be confused with an aggregate single-objective built from multiple objectives. An aggregate single-objective function may take the form of a weighted multiplication, where multiple aspects of a problem domain are measured and then aggregated into a single number that represents the fitness of a solution. This technique, while somewhat more useful than looking at only a single aspect of a problem domain, is still subject to single-objective deception due to the fact that unfit, critical stepping stones are lost in the selection scheme. That is, local optima are still inherent when multiple objectives are aggregated in a single number. Additionally, the methods used of combining the objective functions together require precise steps. In many cases, a slight change in the combination will result in vastly

different solutions generated [27]. MOEAs are built on the idea of looking at multiple objective functions individually and measuring their *Pareto Dominance*.

2.7.2 The Pareto Front.

MOEAs generally leverage a concept called the Pareto Front (PF) (which is made up of Pareto Dominant solutions) to drive evolution. The PF is the set of all individuals that are *non-dominated* [24]. That is, a solution is in the PF (or the *pareto optimal set*) if it is not *dominated* by any other solution. A solution is non-dominated and in the PF if it satisfies

$$x \in PF \iff \neg\exists y(y_0 > x_0 \wedge y_1 > x_1 \wedge ... \wedge y_i > x_i) \tag{2}$$

where, for any solution, no solution exists that is more fit in every objective [6]. The PF, then contains a diverse set of solutions in the fitness space, which may or may not be equivalent to the behavior space. For example, if a solution is only fit in one aspect, but no other solution dominates it, than it will be in the PF. The PF, then, generates fitness diversity by preserving behaviors that may be unfit in certain objectives, but provide crucial stepping stones.

2.7.3 Non-dominated Sorting Genetic Algorithm.

The *Non-dominated Sorting Genetic Algorithm* (NSGA) [11] is one of the premier algorithms in the field of MOEAs [6]. At its core, it leverages the concept of *Pareto Optimality* (PO) as a selection mechanism. NSGA uses a tournament selector where individuals in better Pareto fronts are selected more frequently. In fact, the first PF plays a key role in the approach introduced by this thesis, wherein retaining non-dominated solutions provides an intrinsic method of maintaining diversity in a population. Along the line of maintaining population diversity, it should be noted

that the non-dominated set follows a logistic growth pattern with respect to the number of objective functions [9]. As more objective functions are introduced, the number of individuals belonging to the non-dominated set becomes an increasingly large portion of the population. Thus, a balance must be struck between diversity maintenance and the number of objectives.

2.7.4 Many-Objective Evolutionary Algorithms.

Recently, work in the area of *Many-Objective Evolutionary Algorithms* has attempted to address the issue of logistical growth in the non-dominated set size with respect to the number of objective functions. Many-objective algorithms are those in which there are more than three objectives. Deb [9] demonstrated that once four objective functions were used, about 20% of the population was in the non-dominated set, six objective functions placed about 50% of the population in the non-dominated set, and ten objective functions placed about 90% of the population in the non-dominated set. To address this, Deb and Jain [10] describe a method of dividing the objective space with reference points to discretize the objective space into fixed spaces [10, 21, 53]. These techniques allowed for the use of more than three objectives without disproportionately increasing the size of the non-dominated set. While not used in this thesis, this concept could be applied in future work to further improve the quality of solutions generated by the technique used in this thesis.

2.7.5 Balancing Fitness and Novelty.

As it is the unique ability of MOEAs to pursue not just multiple, but often competing objectives, recent work has shown that an MOEA utilizing novelty *and* fitness as objectives can be useful [34]. The insight here was that the novelty objective would serve to guide evolution past deceptive periods without totally abandoning the desire

to pursue optimality. The results of the experiment, which was conducted in the same deceptive maze domain, suggest that the use of novelty and fitness together does not provide sufficient improvement in performance. What then is needed to appropriately balance the need to mitigate deception with the desire to achieve an objective or a stated goal? Interestingly, human-computer collaboration with evolution has shown potential for addressing this limitation.

2.8 Interactive Evolutionary Computation

Interactive evolutionary computation (IEC) replaces traditional evolutionary components of an evolutionary algorithm (EA) with user selection [48]. In domains where a fitness function is difficult to define, whether because of the subjective nature of the domain or because of its intricacies, IEC becomes particularly effective as it leverages a human's intuition as to what is "good" about a solution. IEC systems tend to succeed in situations where the human adds a subjective evaluation that is difficult to quantify. In creative domains, IEC has created drum beats [17, 49], music [18], art [37, 43, 45], and images [39, 40]. In these problem domains, evaluation is subjective, wherein a human evaluator proves beneficial in providing intuition to the evolutionary process.

As IEC systems are typically constructed on existing EA frameworks (such as ANJI and JGAP [22, 33]), the underlying mechanisms are largely the same. Evolution begins with a random population and evolution proceeds through recombination and mutation operators. Unlike automated EAs, IEC delegates the fitness evaluation and selection routines to a human evaluator. In this way, the human evaluates candidate solutions and selects solutions that serve as the parents that will form the next generation. This process typically proceeds with a harsher selection rate, driving evolution towards particular behaviors very quickly. Unlike fully automated fitness-

based evolution, which follows a strict performance gradient, IEC allows for the idea of *serendipitous discovery*, where important stepping stones for a solution are effectively identified, even when they would have been selected against by a fitness-based search.

Many IEC systems interface with the human evaluator through a system similar to the canonical *Biomorphs* application introduced by Dawkins [7]. The interface presents a human with a series of individual solutions from which the user selects the parents of the next generation. Once the selections are made, the IEC system performs the EC steps of recombination and mutation to create the new generation. This process repeats until a stopping criteria is met (i.e. the user is satisfied with the solution).

The key drawback to human-computer interaction is fatigue [48]. That is, as mutational changes may be small and even imperceptible, humans fatigue quickly from looking at large quantities of individuals with little progress being made between generations. According to Takagi [48], typical IEC sessions last fewer than 20 generations. It becomes infeasible for a single human to evolve a solution in complex domains in a single IEC session. It is to address this issue that several fatigue-reduction techniques have been created.

2.8.1 Multi-user IEC.

First conceptualized by Sims [43], multi-user IEC distributes the evolutionary workload between many individuals. Rather than a single person evolving the individuals to the goal, a community of users is used to reduce fatigue on a single individual. Sims [43, 45] used an interactive art exhibit to showcase this capability. In the art exhibits, humans voted on their favorite art and, in doing so, subjectively evaluated the quality of the art. In his first work, Sims [43] used the community to

evolve abstract art. The second exhibit was similar in its implementation, having many users evolve interesting sea creatures [45].

In a more recent effort in multi-user IEC, Secretan et al. [39, 40] demonstrated this capability using the Picbreeder online service. The results show that a many-human IEC system can evolve interesting, abstract solutions over a period of time with minimal fatigue.

In many situations, however, a multi-user IEC system is infeasible, particularly in non-subjective domains with specific goals. In these scenarios, a single user is important to maintaining continuity and driving the solution to the goal. To address fatigue in these systems, short-term automated searches have been used [3, 19, 51] to reduce the number of human interactions and reduce fatigue.

2.8.2 Fitness Assistance in IEC.

Work has been done to show that fitness in combination with IEC can augment the human's capabilities, speeding evolution towards the goal and reducing fatigue by reducing the number of steps required to reach the goal [3, 19]. In these works, the IEC system tries to simulate what a human would pick via modeling algorithms. The short-term searches significantly decreased the number of steps required to find solutions, but this methodology is subject to deception in the short-term search.

2.8.3 Novelty Assistance in IEC.

Woolley and Stanley [51] demonstrated that novelty is a better aid than fitness for IEC. The Novelty-Assisted IEC (NA-IEC) framework, wherein the human is only presented with solutions that are above a novelty threshold (i.e. solutions that are different from what has been seen previously) from which the user can select candidates as in a traditional IEC system, outperformed every previous methodology by a

significant margin. Where the deceptive maze domain required more than 1000 user operations by a human in traditional IEC, NA-IEC reduced the number of operations to 32. Additionally, the number of evaluations (i.e. the number of individuals created), was reduced from more than 24,000 to about 7,500. The success of NA-IEC is in its ability to present the human evaluator with *behavioral diversity*, thus reducing fatigue by automating the tedious search for potential stepping stones, while leveraging human insights about what is important in a given domain.

2.8.4 Multi-Objective Assistance in IEC.

No work has been done previously in combining MOEAs with IEC. It is the key insight of this thesis that MOEA techniques can be leveraged as a mechanism to preserve behavioral diversity in much the same way as novelty search, thus allowing access to user's intuition through a multi-objective framework. This will enable user's preferences to be measured and quantified by examining his or her choices in the fitness space. The measurement of these preferences and selections could allow the development of a metaheuristic that profiles human intuitions about a complex problem domain.

III. POA-IEC Framework

The ability of a human to identify what is important in a domain is an important aspect of IEC. NA-IEC proved effective for its ability to collect novel stepping stones that a human could identify during the selection process, thus enabling serendipitous discovery. This section introduces a new IEC approach, called *Pareto Optimality Assisted Interactive Evolutionary Computation* (POA-IEC), which encourages behavioral diversity through MOEA techniques, rather than novelty, because it provides a way to measure what aspects of a solution human evaluators see as important for a given domain.

In this approach, a user is asked to perform selection from a pool of candidates and then apply an evolutionary operator: a multi-objective fitness optimization or a short-term pareto optimality based search. By selecting a mode of evolution, the user has fine-grained control over the evolutionary direction towards a goal. Inspired by NA-IEC's contribution of short-term automated searches for novelty, POA-IEC employs a multi-objective algorithm to maintain behavioral diversity during IEC.

3.1 Software Architecture

Before discussing the interface and evolutionary method, a brief overview of the software architecture (Figure 5) is helpful in distinguishing between problem domain and POA-IEC functionality. This diagram presents an overall architecture for the POA-IEC framework. Inside the framework, JGAP provides the genetic algorithm underpinnings for ANJI, a NEAT implementation for neural networks. An IEC system built on top of ANJI allows the user to control evolution. The user interacts with this system via the interface, viewing the phenotype graphical representations and performing the selection process. When the user has selected promising candidate

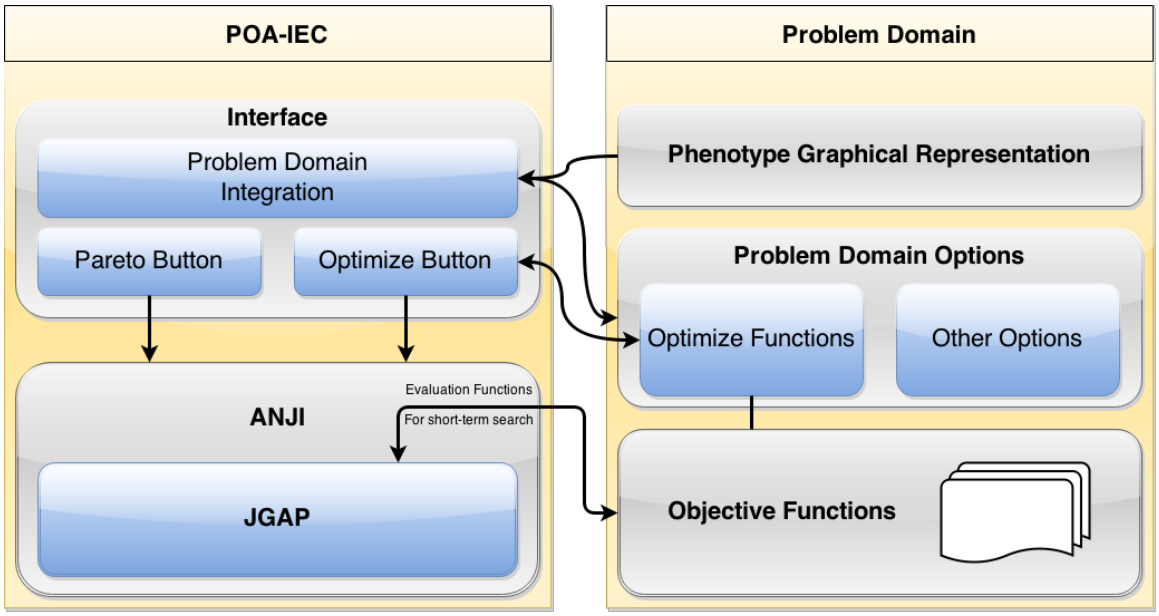


Figure 5. POA-IEC Software Architecture - The yellow boxes represent the separation of the problem domain and the POA-IEC framework, while each subcomponent illustrates how the various software pieces work together. The key takeaway is that POA-IEC is problem domain agnostic (i.e. the evaluation functions for short-term search and the graphical representation of phenotypes is problem domain specific).

behaviors, he or she can start the short-term evolution by pressing the *Pareto* or *Optimize* button, launching an evolutionary process within ANJI.

The *Pareto* button launches a short-term multi-objective search, while the *Optimize* button performs a single or multiple objective optimization. In both cases, calls are made from the interface to perform prescribed evolutionary tasks within ANJI and JGAP. In each case, the problem domain must implement evaluation functions for JGAP to use in its short-term search. These are implemented outside of POA-IEC, as they are specific to each domain. Similarly, the visual representation and domain customization options are provided by the domain implementation, not POA-IEC.

3.2 Software Implementation

POA-IEC is implemented on a version of the open-source ANJI [22] package, modified to support multi-objective algorithms. The modification was implemented

specifically for POA-IEC and implements an algorithm that is based loosely on the NSGA-II [11] algorithm, relying strictly on the Pareto Front ranking for its selection mechanism. POA-IEC provides the user a way of interacting with the multi-objective ANJI, allowing for control of evolution.

3.3 User Interface

The main interface for POA-IEC is graphical and implements the evolutionary modes described (Figure 6). Once the user has selected one or more candidates from the on-screen population, they can elect to generate a new population of pareto optimal candidates or optimize a particular individual.

While the NA-IEC implementation also had an optimize function, based on reducing distance to goal, POA-IEC allows users to select (under Evaluation Options — Optimize Functions) which fitness functions they feel are most relevant at a particular moment. This allows the user to select the parts of the solution they would like to optimize and let an automated process perform the optimization. For example, if a path through the maze were very jagged, a user could select the "smoothness" objective function (if one exists). The automated process would then perform single-objective optimization on that function, with the parent being the currently selected individual, and return a new individual that was optimized to that objective. Such a capability allows the user to direct evolution away from the path of fitness in search of critical stepping stones that can be optimized later. Similarly, this capability supports optimizing two or more objectives simultaneously, providing an automated process with which to perform a custom optimization most appropriate for the current state of evolution.

Choosing the *Pareto* button starts a short-term Pareto Dominant search that runs until a new non-dominated solution is found, or until an evaluation limit is reached.

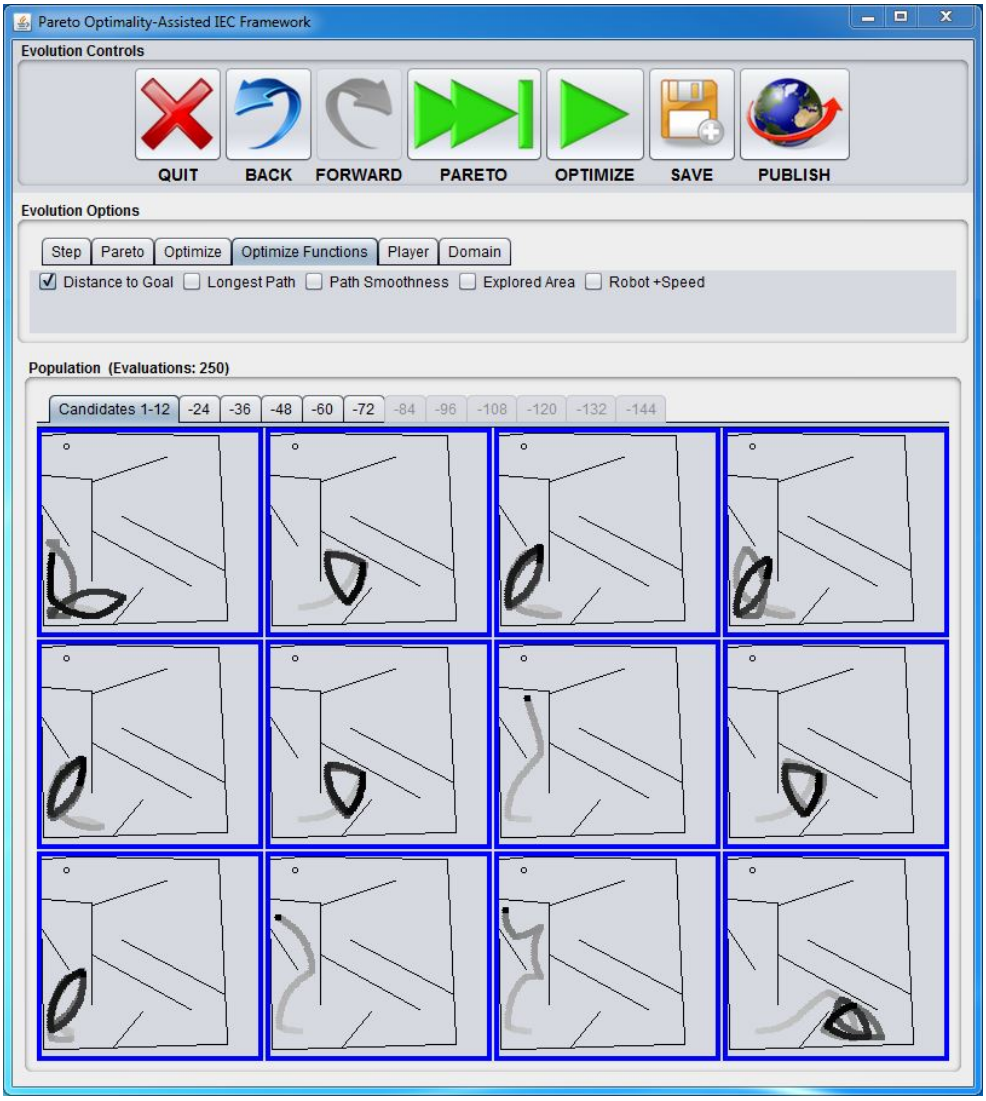


Figure 6. Main POA-IEC Interface - The user interface for the POA-IEC framework has several parts: *Evolution Controls*, *Evolution Options*, and the *Evaluation Population* (from top to bottom). Solutions are presented as gradient trails through the maze, with darker gradients showing the later points in the robot's path. The Evaluation Population is presented in order from most to least fit (from top left to bottom right). Shown in the Evolution Options, the *Optimize Functions* interface presents the list of objectives that can be optimized in the form of a series of checkboxes. The user selects the objective functions they would like to optimize and then clicks the *Optimize* button.

This automated process, which replaces the short-term novelty search function in NA-IEC, provides the diversity in the evaluation set needed to reduce fatigue in the human evaluator. To accomplish this, a larger background population is seeded with mutations of the user selected individual. A non-dominated sorting evolutionary algo-

rithm (similar to NSGA) is then applied until a solution that dominates the selected individual is found or until an evaluation limit is reached, whichever is first. In this way, behavioral diversity across the Pareto Front (PF) is maintained by the diversity of objectives. Thus, the next on-screen population is the set of non-dominated solutions, where at least one individual dominates the user's previous selection. The user selection and background operation cycle is repeated until the user is satisfied, the overall goal is met, or evolution is abandoned.

It should be noted here that only the first PF is presented to the user because each subsequent PF necessarily contains solutions that are less fit than those in the first PF. This insight is supported via the experimental results, which demonstrate that the first PF is all that is necessary to adequately present diversity to the user.

3.4 Typical Evolutionary Sequence

A typical evolutionary sequence is shown in Figures 7, 8, and 9. Initially, a random population of 250 individuals is generated, with only the non-dominated set being presented to the user (Figure 7a). By hovering over an individual, information is presented including the fitness scores for each objective function, the number of other solutions the individual dominates, the number of others that dominate the individual, and the novelty score of that individual. This functionality is shown in Figure 7b—note that the top left individual dominates 22 others and is dominated by 0 others. This agrees with the concept of the non-dominated set; there are no individuals that dominate the individuals presented to the user.

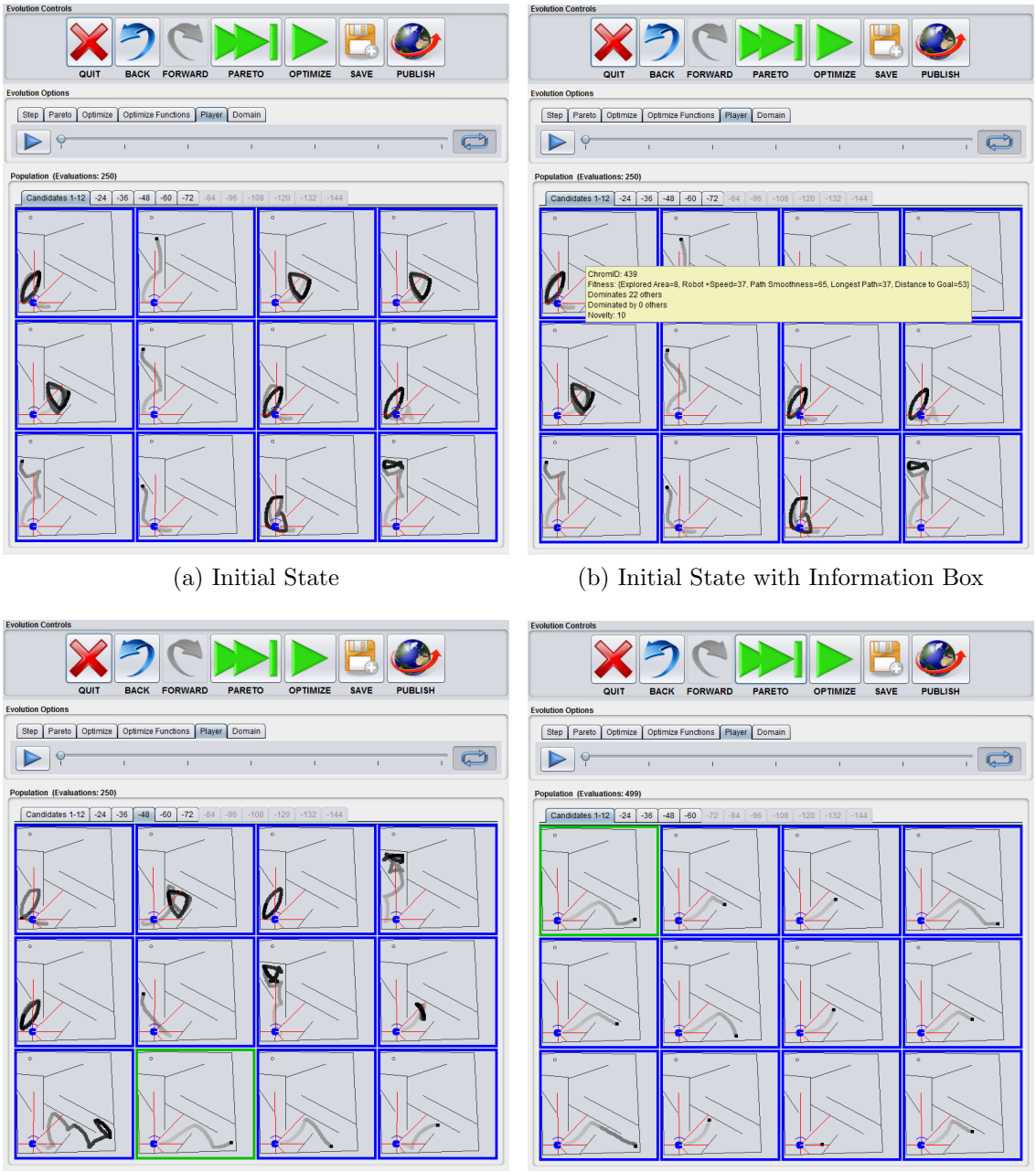
Once the user examines the individuals in the non-dominated set (in this case about 72 individuals), they can select one or more individuals that they feel provide progress towards the goal. This is shown in Figure 7c as the user has selected an individual on the 4th page of results. Once the user selects this individual, he or she

clicks the *Pareto* button and a new generation is created using a short-term MOEA search (Figure 7d). At this point, the non-dominated set is less than 60 individuals, sorted into 5 pages of individuals for the human to examine.

The user once again examines the individuals available and selects an individual on the 2nd page of results (Figure 8a). This individual has made progress over the first by navigating around the bottom right corner of the maze. Once again, the user presses the *Pareto* button to run a short-term MOEA search with the selected path as the parent to the new generation, which is presented in Figure 8b. This step provided (in this evolution) a serendipitous discovery. The user now has three options that have navigated the narrow passage in the middle of the maze and have reached the large chamber of the maze. One individual even navigated into the chamber of the maze with the solution. From here, the user selects the individual that navigated into the solution chamber (Figure 8c). At this point, either the *Pareto* or the *Optimize* buttons will likely solve the maze. In this particular run, the user chooses the *Pareto* button and the population in Figure 8d is generated using the short-term MOEA search. A solution is in this new population, which the user selects in Figure 9. If satisfied with the solution, the user can choose to *Publish* the solution. If not, he or she can *Optimize* the solution, or continue to search the space for a different solution using the *Pareto* button.

3.5 Pareto Optimality Pointing Vector

Another feature of POA-IEC is that, once the next on-screen population is ready, the solutions are sorted by their distance from a vector called the *Pareto Optimality Pointing Vector* (POPV). The POPV is a weight vector that begins equally weighted (e.g. $[0.5, ..., 0.5]$ for an n-objective space) and shifts in the fitness space based on previous user choices. Each time a user makes a selection, that solution's POPV is



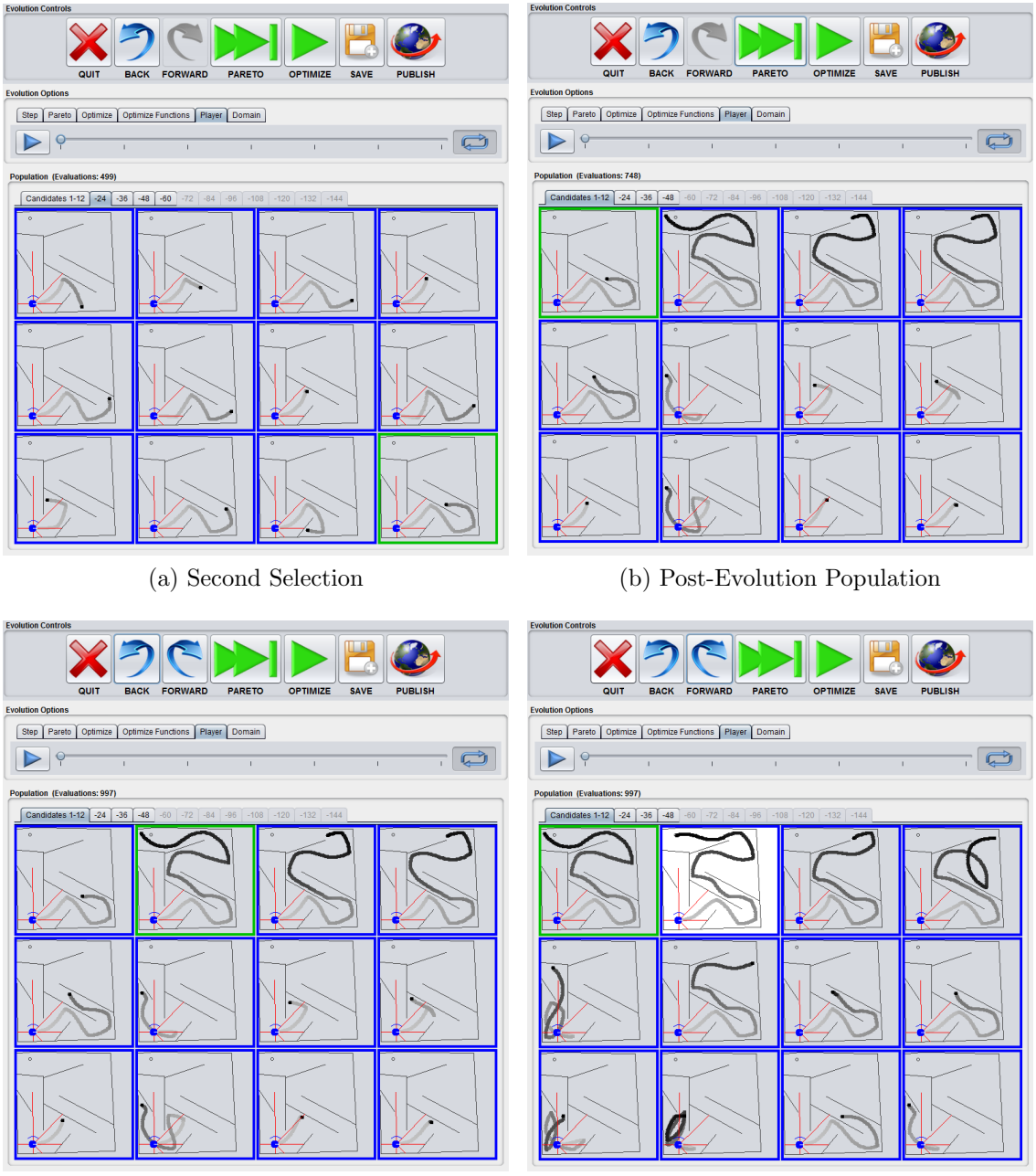
(a) Initial State

(b) Initial State with Information Box

(c) First Selection

(d) Post-Evolution Population

Figure 7. Evolutionary Sequence: Part 1 - A typical evolutionary sequence is shown in this figure. Part 1 of the evolutionary sequence has four images. Figure 7a shows the initial population after the program starts. At this point there are six pages of paths (up to 72 individuals) for the human to search through. 7b shows the result of hovering over a square—the data shows the fitness of the solution in all five objective functions as well as how many solutions it dominates, how many dominate it, and the novelty of the path. 7c shows the first user selection of an individual on the 4th page and 7d shows the results of evolution using that individual as the seed. Note that the new population has only 5 pages of paths (up to 60 individuals) for the human to search through.



(a) Second Selection

(b) Post-Evolution Population

(c) Third Selection

(d) Post-Evolution Population

Figure 8. Evolutionary Sequence: Part 2 - A typical evolutionary sequence is shown in this figure. Part 2 of the evolutionary sequence has four images. Figure 8a shows the second user selection, an individual on the 2nd page, and 8b shows the results of evolution using that individual as the seed. Note that, at this point, there are only 4 pages of results to search and that a serendipitous behavioral discovery has been made—a solution path has made it close to the goal. 8c shows the third user selection and 8d shows the results of that selection. The maze has been solved at this point.

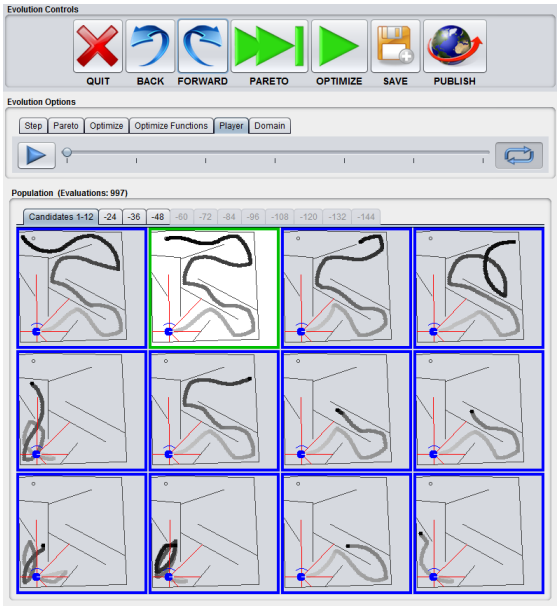


Figure 9. Evolutionary Sequence: Solution - A typical evolutionary sequence is shown in this figure. Figure 9 shows the solution to the maze selected. At this point, the user can *Optimize* the solution or they can publish the solution if they are satisfied with it.

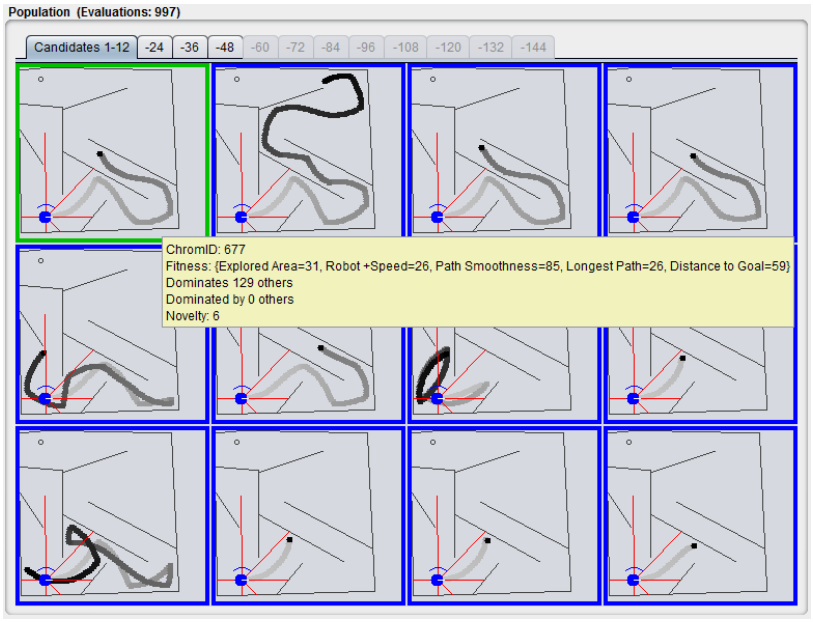
calculated and the POPV is set to that new vector. In this way, the set of non-dominated solutions is presented to the user in order of distance from the POPV, thus sorting the new evaluation population by what is *most similar* to what he or she selected previously. While some phenotypically different solutions could have a similar POPV, this mechanism provides a way of logically sorting the PF for presentation to the user. This has no influence on the underlying MOEA as it is intended to reduce fatigue on the user because, as the trial progresses, the user should increasingly be presented with solutions that are most similar to previously chosen solutions. This is apparent in the evolutionary sequence in Figures 7, 8, and 9. Initially, the user in that sequence chose from the 4th page of paths (POPV is evenly weighted), then from the 2nd page (POPV influences the sort), then chooses the top sorted path the final two times.

To further demonstrate this sorting mechanism, Figures 10 and 11 show the 4 pages of a population at a midpoint of a typical run. Figure 10a shows the first page

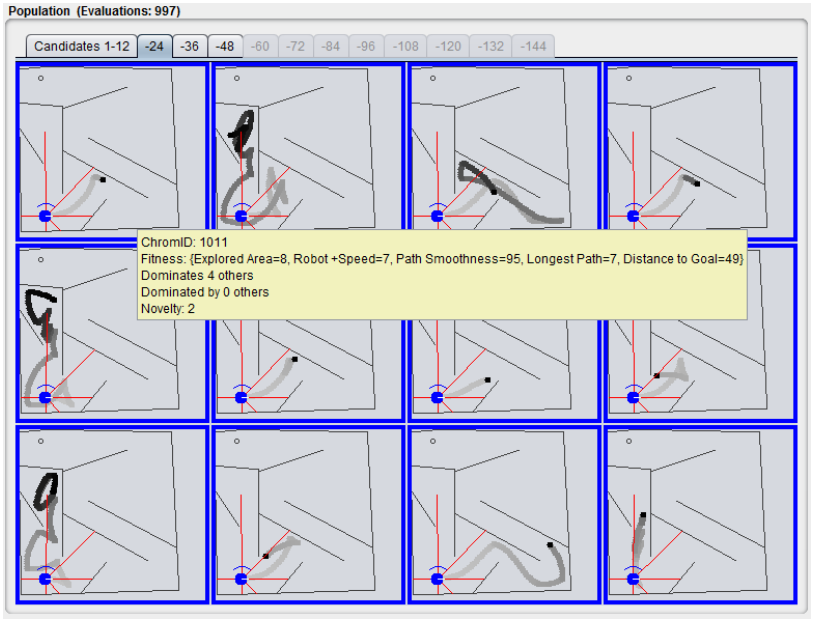
of individuals, where the top left was the previously selected path and the rest are its children. Note that the path sorted to the top of the population seems to be the next logical selection for evolution. This implies that the POPV sorting should help fatigue significantly. Additionally, the next four behaviors (from left to right, top to bottom) are all phenotypically very similar to the chosen individual, further supporting the concept of the POPV sorting mechanism by presenting similar behaviors to those chosen previously.

On the second page of results (Figure 10b), the paths are phenotypically different from their ancestor. The POPV has sorted them out of the first page of results onto the second one. On the third page of results (Figure 11a), there are some individuals that are clearly related to the chosen parent, but are degenerative versions of the parent (i.e. they do not travel as far or exhibit strange behaviors). Of note is that there is a path in this population that reaches the top right chamber of the maze, which is significant in progress to the goal. However, this path exhibits very strange behavior at the end of its run that impacts at least one of the objective functions, causing its undesirable behaviors to drop it down the POPV sorting rankings. The last page of results (Figure 11b), has only 4 results, each of which exhibit extremely undesirable behaviors.

The POPV sorting mechanism, which does not influence the short-term MOEA, should help reduce fatigue by bringing the most phenotypically similar behaviors to the front of the evaluation population if, in fact, fitness behavior is equatable to phenotypic behavior. Most importantly, the POPV provides a quantitative measurement of what phenotypic attributes the human evaluator is intuitively selecting, a capability that other deception-avoiding techniques such as novelty and NA-IEC simply do not have. Thus, it is a significant insight that the POPV can be used to track



(a) Page 1



(b) Page 2

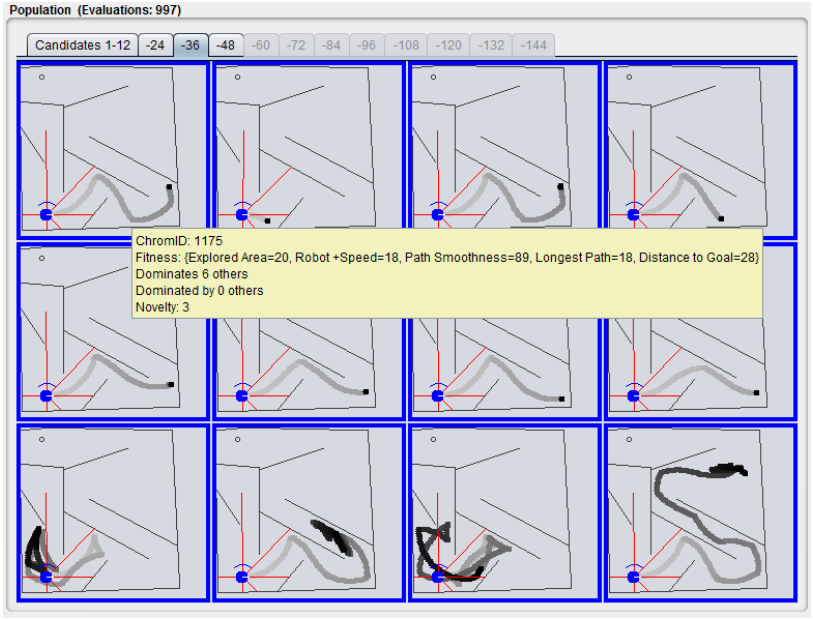
Figure 10. POPV Sorting: Part 1 - This set of images shows the robot paths as they are sorted by the POPV. In each image, the fitness information shown is the hoverover text from the top left individual. Figure 10a shows the fitness information for the parent of the generation, as well as the 11 closest individuals to the line that extends from its POPV vector. 10b shows the second page of results, which are the next 12 closest individuals as sorted by distance from the POPV. Note that the behaviors shown in 10a are subjectively more similar to the parent than are the individuals in 10b.

user selections and provides insight into the domain that is difficult to capture with classical approaches.

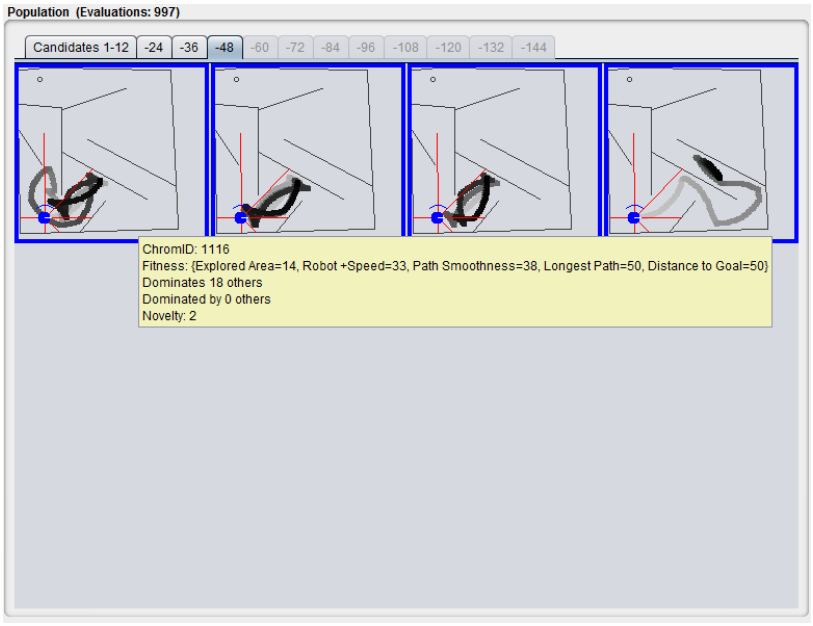
3.6 Objective Functions

Objective functions, being unique to the problem domain, must be implemented outside of the POA-IEC framework. It is recommended that there be at least three objective functions to adequately preserve behavioral diversity. This is shown in Figure 5, and must be implemented according to the specifications in the JGAP library used by POA-IEC.

In short, in an effort to reduce fatigue and accelerate the human-computer interaction, the new POA-IEC approach leverages multi-objective techniques to generate behavioral diversity and gain knowledge about user selection preferences at runtime.



(a) Page 3



(b) Page 4

Figure 11. POPV Sorting: Part 2 - This set of images shows the robot paths as they are sorted by the POPV. In each image, the fitness information shown is the hoverover text from the top left individual. Figure 11a shows behaviors that seem to be degenerative models of the parent. An interesting individual on this page is the bottom right, which successfully reaches the top right chamber of the maze. While this seems to be similar to the individual deemed closest to the POPV (top row, 2nd column of Figure 10a), the indecisive behavior at the end of the path severely impacted the its fitness such that it fell to the third page of results. 11b shows the last page of results, which are the furthest individuals as sorted by distance from the POPV. Note that the behaviors shown in 11b are subjectively useless behaviors and would seem to suggest that sorting by the POPV has indeed pushed the least desirable individuals to the last page of results.

IV. Experimental Setup

To demonstrate the effectiveness of POA-IEC, an experiment is performed in the deceptive maze domain introduced by Lehman and Stanley [30, 32] to allow for a direct comparison between POA-IEC, NA-IEC [51], a traditional MOEA, and pure novelty search. In this domain (Figure 3a), the maze is a metaphore for search in a space where the path to the goal is not known a priori. The goal is to evolve a controller for a robot to navigate from the start point to the end point of the maze. The evolutionary techniques evaluated here are measured against each other in terms of the number of evaluations required to reach a solution.

To evaluate the POA-IEC framework, six novice users (i.e. they were unfamiliar with the framework and had no experience with the field of EC) were selected to generate 30 solutions. The users were introduced to the POA-IEC framework and each was asked to evolve five solutions to the maze domain. The hope is that the performance of these individuals will correlate to a reasonable picture of how the POA-IEC framework will perform in real-world scenarios with experienced users. If a user felt their solution had become "stuck", he or she was permitted to restart the evolution, though the evaluations that occurred prior to the restart were included in that run's results.

To evaluate the degree to which the multi-objective component is aiding evolution, a fully automated MOEA was run with the same objective functions and evolutionary parameters as POA-IEC. The fully automated MOEA employed an NSGA-II-like [11] algorithm to perform selection, recombination, and mutation. A population of 250 individuals is doubled in size via tournament selection, mutation, and recombination. The 250 "most optimal" solutions were retained as the new population. The optimality comparator is the dominance operator, with non-dominated solutions receiving the highest score. In this implementation, the individuals are ranked according to

how many others they are dominated by, with the non-dominated set being dominated by 0 others. For example, an individual dominated by 3 others receives a higher dominance ranking than one dominated by 9 others. In this way, the population is divided into a ranking of Pareto Fronts, and selected according to that ranking.

4.1 Objective Functions

Five objective functions were chosen for this experiment: *endpoint distance to goal*, *path length*, *path smoothness*, *path area*, and *robot speed*. The number of objective functions was limited to five to keep the number of members in the first PF competitive. That is, since the size of the non-dominated set grows logistically with respect to the number of objective functions [9] and no many-objective algorithms were employed, five objective functions provided a good balance of fitness diversity and population size. Too few objective functions does not adequately preserve behavioral diversity while too many does not help reduce the size of the population adequately for the user.

Insights from evolving solutions in the maze domain [51] suggest that human users are selecting for some quality of a solution, even if they cannot articulate it themselves. By creating more objective functions, the POPV provides an insight into what the user's selection criteria might be. The following is a description of the objective functions introduced in this experiment.

4.1.1 Endpoint Distance to Goal.

The endpoint distance to goal heuristic calculates the Euclidean distance between the goal and the endpoint of a robot's path. This objective function is an inverted minimization function (i.e. a high fitness is assigned to a low Euclidean distance). A fitness of 100 is assigned to a distance of zero, while a fitness of zero is assigned to

the maximum distance, calculated as the diagonal of the bounded maze. The fitness is expressed by

$$f_d(x) = 100 * (1 - \frac{d}{d_{\max}}) \tag{3}$$

where d is the individual's endpoint distance to goal $d_{\max}$ is the length of the diagonal of the bounded maze. This objective function is designed, as it was originally introduced by Lehman and Stanley [30, 32], to apply an evolutionary pressure that moves robots closer to the goal.

4.1.2 Path Length.

The path length heuristic calculates the total length of the path in an effort to favor longer paths. This objective function is a maximization function, applying a fitness of 100 to the longest possible path and a fitness score of zero to the shortest possible path. The maximum path length is calculated as the maximum robot velocity multiplied by the time limit of the run, while the shortest possible path length is zero. This is expressed simply as

$$f_{length} = 100 * \frac{l}{l_{\max}} \tag{4}$$

where l represents the path length and $l_{\max}$ is the maximum path length. In this way, the fitness of a robot is directly proportional to its path length in this objective, encouraging robots to drive further.

4.1.3 Path Smoothness.

The path smoothness heuristic is designed to encourage "smooth" robot paths. While this is partly an aesthetic objective function, it also encourages robots to drive straight lines when possible and to take turns without hitting a wall. The function walks the path, selecting three consecutive points and samples the angle between those

three points. A sum of all of the angles along the path is taken and a fitness function is applied. A fitness of 100 is assigned to a straight line (which is the smoothest shape) and a fitness of zero is assigned to a robot that takes the maximum turn rate at every single step. This can be expressed by

$$f_s = 100 * (1 - \frac{\sum_0^{t_{max}} turn_t}{\sum_0^{t_{max}} turn_{max}}) \tag{5}$$

where t_{max} is the length of the simulation in seconds, $turn_t$ is the change in angle of movement at time t, and $turn_{max}$ is the maximum turn angle.

4.1.4 Path Area.

The path area heuristic encourages *exploration* by rewarding behaviors that end farthest from the starting point. It calculates the bounding box for the path and calculates its area. A higher score is assigned to high areas, making this a maximization function. The fitness is assigned by

$$f_A = 100 * \frac{A}{A_{max}} \tag{6}$$

where A is the area of the bounding box of the path and A_{max} is the area of the maze.

4.1.5 Robot Speed.

The robot speed heuristic rewards behaviors that drive in the positive direction (i.e. not in reverse) at high velocities. It is a maximization function that assigns a score of 100 to a robot who drives at the maximum speed for the entire duration of the run. This can be expressed with

$$f_V = 100 * \frac{V_{avg}}{V_{max}} \tag{7}$$

where V_{avg} is the average velocity and V_{max} is the maximum velocity. This is designed to encourage behaviors that drive forward and have high velocities (i.e. that reach the goal first). While backwards driving solutions are valid for this domain, this heuristic specifically punishes that behavior.

4.2 Experimental Parameters

The evolutionary parameters for this experiment are derived from the previous work by Woolley and Stanley [51], Lehman and Stanley [30, 32], and designated, optimal parameters for NEAT [46]. The experiments were run with a version of the ANJI [22] package, modified to support multi-objective algorithms. The parameters for NA-IEC were adopted for POA-IEC whenever possible: The IEC population size was 12, the search population sizes were 250, and each run was limited to 250,000 total evaluations. The speciation threshold, δ_t, was 0.2 and the compatibility modifier was 0.3. The NEAT parameters were standard: 5% chance of node addition, 10% chance of connection addition, 1% chance of connection deletion, and weight mutation power of 0.8. During optimization operations, the connection weight mutation power was lowered to 0.1 to allow for fine-grained changes. The ANN used an unsigned sigmoid activation function with recurrent connections, with a shifted output range of [-0.5,0.5].

The following parameters are specific to the new techniques used in this thesis. The POPV is initially set to an $n \times 1$ dimensional array, with each element initialized to 0.5. The POPV is adjusted during runtime and is reset at the beginning of each run. The *Optimize Functions* options are locked with only the *endpoint distance to goal* heuristic enabled in order to mimic the *Optimize* button operation available under NA-IEC.

V. Experimental Results

As in the original experiments by Lehman and Stanley [30, 32] and later by Woolley and Stanley [51], any behavior whose endpoint is within a distance of 5 from the goal is considered to be successful. The main result is that POA-IEC discovers successful solutions in significantly fewer evaluations than both the automated MOEA and NA-IEC, and thereby all methodologies that NA-IEC outperformed. POA-IEC also found solutions an order of magnitude faster than pure novelty search alone.

5.1 Evaluations

POA-IEC evolved 33 successful solutions in an average of just 2,562 (sd = 1,691) evaluations, which is a significant ($p < 10^{-3}$; Student's t-test) improvement over NA-IEC (7,481 evaluations, sd = 6,610). POA-IEC also shows a significant ($p < 10^{-11}$; Student's t-test) improvement over the automated Novelty search (34,207 evaluations, sd = 19,249) and automated MOEA (12,258 evaluations, sd = 6,146). These results are depicted graphically in Figure 12. In this figure, there are two boxes and many small circles. The inner box represents the standard error of the mean (SEM) and the outer box represents the first standard deviation while the circles each represent single run. Note that the evaluations scale is cut off at 80,000 for the purpose of readability of the POA-IEC results. The figure graphically demonstrates that POA-IEC clusters at a much lower evaluation count than does any methodology to date. Significantly, traditional automated fitness search rarely completes the deceptive maze [51] and is often forced to finish at the cutoff of 250,000 evaluations. In fact, only 4 out of 30 fitness searches found a solution in the alloted 250,000 evaluations limit, and only three did so in less than 80,000 evaluations. Figure 12 shows the benefit of exceptional performance of each deception avoidance technique. All data other than POA-IEC

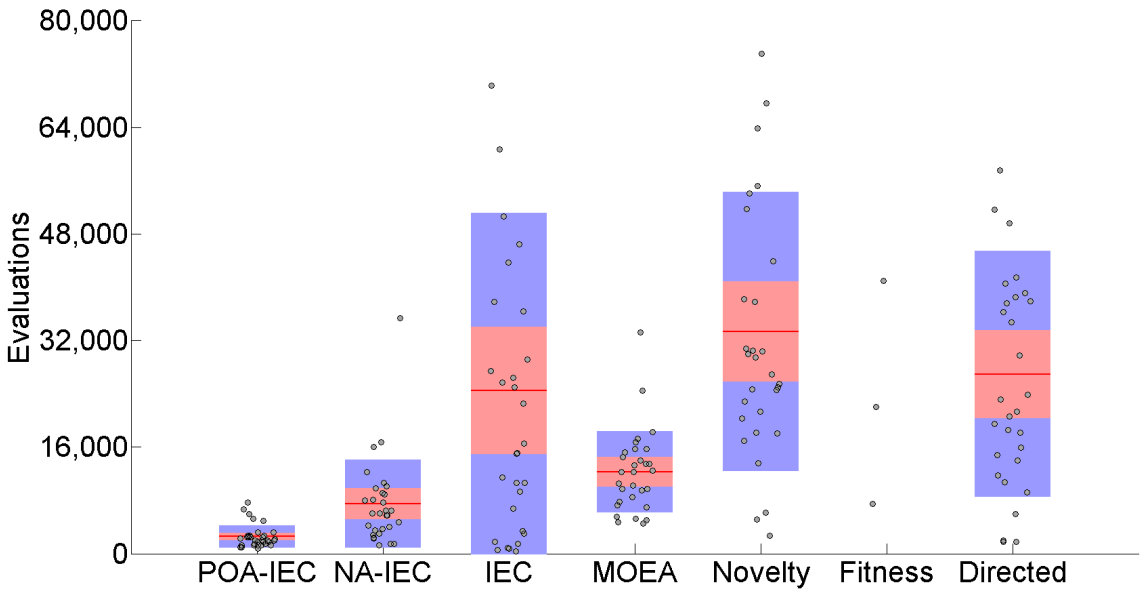


Figure 12. Evaluations Required to Solve Deceptive Maze - The number of evaluations required to solve the maze are shown. POA-IEC, NA-IEC, IEC, automated MOEA, automated novelty, automated fitness, and automated directed search results are marked with a line indicating the average number of evaluations, one and two boxes indicating the standard error from the mean and one standard deviation respectively, and the individual runs marked with dots. The main result is that POA-IEC significantly outperforms all other methods, averaging fewer evaluations per solution than the other methods. Note that automated fitness search only found solutions in 4 out of 30 runs, reaching the max evaluation count (250,000) on all other trials.

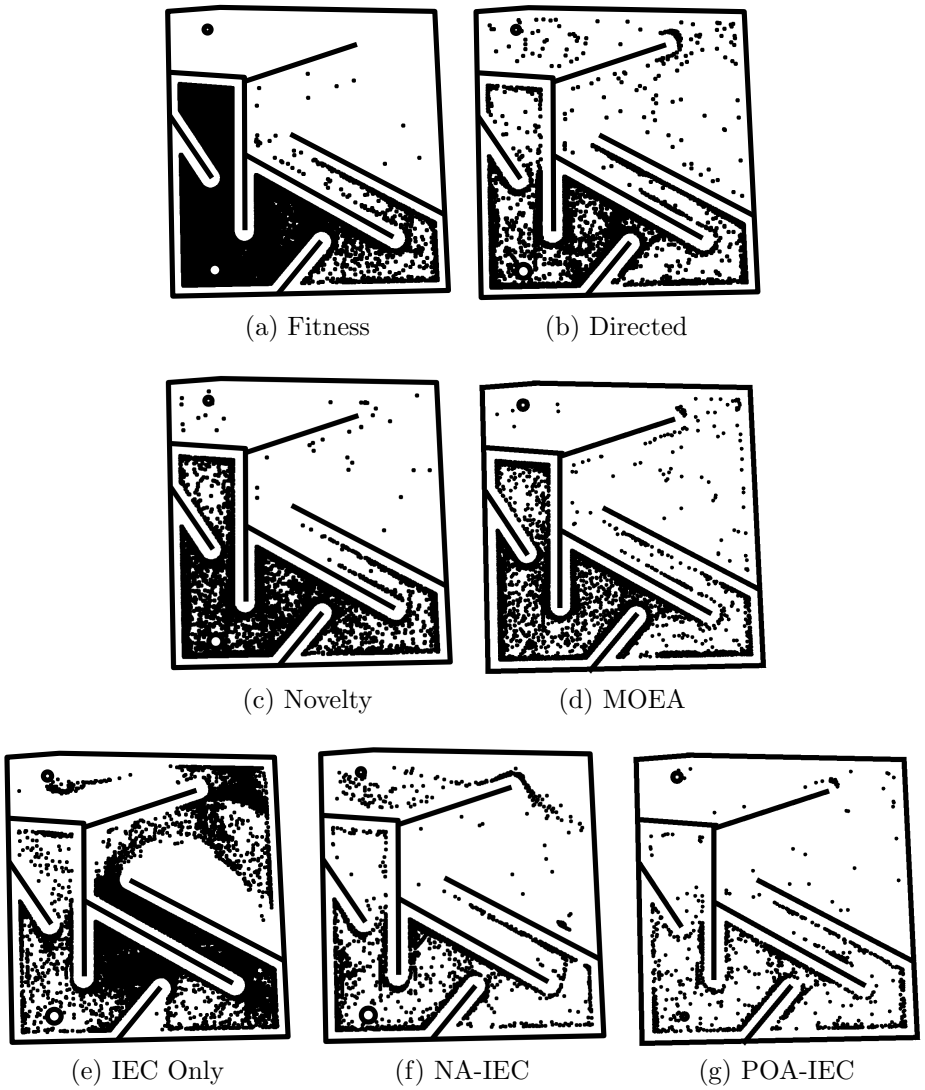
and automated MOEA is the original data from the experiments by Woolley and Stanley [51], which used the same number of users as this experiment. Of note is the fact that the results from Woolley and Stanley [51] were generated from a pool of "expert" users (i.e. familiar with EC and IEC) where the results from this experiment were generated from a pool of users unfamiliar with EC, further illustrating the effectiveness of POA-IEC.

5.2 Endpoint Distributions

Figure 13 shows typical endpoint distributions of the algorithms compared. The images illustrate their exploration patterns. Figures 13a, 13b, 13c, 13e, and 13f, show the results obtained by Woolley and Stanley [51]. Figure 13a shows a typical automated fitness approach, clearly showing the effects of deception on the search.

The fitness search explores the deceptive cul-de-sac extensively, rarely exploring any other area. The directed approach, shown in Figure 13b, show the results of rewarding a a fitness-based search if the path is known a priori. The maze is much more evenly explored, but the solution path is not known beforehand in most domains. Even so, the directed approach uses more evaluations than does NA-IEC or POA-IEC (Figure 12). Novelty search (Figure 13c) explores the maze evenly than does fitness, while Figure 13d shows that the automated MOEA approach provides less dense distributions of endpoints than does novelty search, yet explores the maze in a similar fashion. This seems to once again suggest that fitness diversity provided by MOEAs results in similar behaviors to novelty. The traditional IEC approach (Figure 13e) clearly shows the influence of human led search. The endpoints certainly cluster around the solution path, with very few explorations into the deceptive cul-de-sac. Figures 13f and 13g illustrate the similar influence of human-led selection in the NA-IEC and POA-IEC approaches. Additionally, NA-IEC and POA-IEC seems to spend much less time at critical junctures as their diversity mechanisms quickly find solutions that progress past those points.

The distributions provide several insights into the human selector process that are distinct from automated runs. In all IEC approaches, there are far fewer solutions that end in the deceptive cul-de-sac, indicating that the human aggressively prunes solutions that explore that part of the maze. Additionally, there are persistent clusters of endpoints at "important" junctures in the maze, indicating that the human is providing key insights into how a robot should move through the maze. The exploration patterns also show the most important difference between IEC and automated approaches—IEC approaches find solutions in fewer evaluations than the automated search. Another insight is that POA-IEC and NA-IEC are similar in their exploration patterns (Figures 13f and 13g). While the POA-IEC distribution is less dense than



(a) Fitness (b) Directed

(c) Novelty (d) MOEA

(e) IEC Only (f) NA-IEC (g) POA-IEC

Figure 13. Endpoint Distributions Comparison - The distributions of endpoints for candidate paths show the impact of IEC and MOEA algorithms on the evolution of solutions. The automated novelty search avoids the deceptive cul-de-sac reasonably well. The automated MOEA spends less time exploring around the starting point, as the density of the endpoints around the starting location is less than that of novelty search. Additionally, the MOEA has a more even exploration in the top part of the maze. NA-IEC and POA-IEC have a clear evolutionary driver (i.e. the human) that drives endpoints to fall along a certain path. The endpoint density on the two mazes clearly indicates a human is a favorable addition to the selection algorithm, greatly reducing the endpoint density. Note that, other than the POA-IEC and MOEA distribution charts, all other distributions are from the experiment conducted by Woolley and Stanley [51] and are used for comparison.

NA-IEC, they show evidence of exploring the maze in similar ways, indicating that MOEAs provides similar behavioral diversity that novelty did in NA-IEC.

5.3 Time

With regard to wall-clock time, Woolley and Stanley [51] reported that NA-IEC performed faster in time than the automated novelty search. Here, the automated MOEA search ran in 106 seconds (sd = 85), significantly faster ($p < 10^{-4}$; Student's t-test) than both NA-IEC and POA-IEC. POA-IEC showed improvement over NA-IEC in time however, solving the maze in an average time of just 248 seconds (sd = 269) compared to NA-IEC (402 seconds, sd = 374).

5.4 ANN Complexity

Finally, POA-IEC showed improvement over NA-IEC's reported ANN complexity (0.5 hidden nodes, sd = 1.01), finding solutions that averaged just 0.375 hidden nodes (sd = 0.49). Surprisingly, the automated MOEA evolved even less complex solutions than both IEC methods, finding solutions that averaged 0.23 hidden nodes (sd = 0.43).

5.5 Pareto Front Novelty

To validate the methodology of only presenting the user with the first PF in each run, data was recorded about the *novelty* of every individual at the time they entered the population. If the first PF is more novel (i.e. there is more behavioral diversity) than the remainder of the population, then the addition of the remainder of the population to the human selection population is unnecessary. Furthermore, the addition of these results could dilute the population, increasing fatigue and slowing evolution.

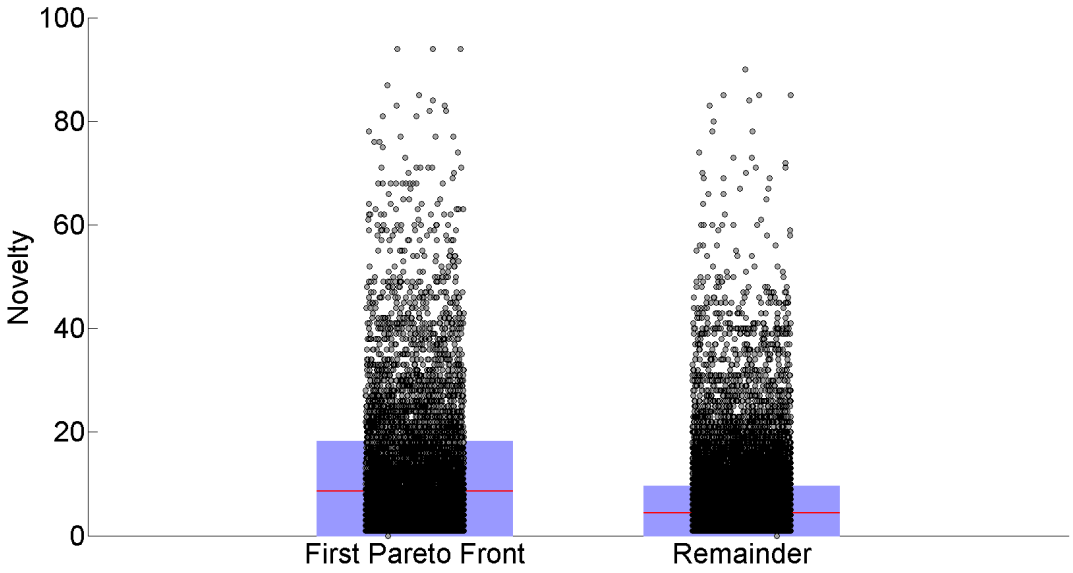
Across all runs, 16,107 individuals were presented to the user (first PF) while 65,072 individuals were hidden. Comparing these two groups, the first PF was found

to be significantly more *novel* ($p = 0$, Student's t-test) with an average novelty of 8.56 (sd = 9.74) compared to all other PFs combined (novelty of 4.43, sd = 5.17). Such results quantify the intuition to apply the first PF as a diversity mechanism. Furthermore, because the non-dominated set is sufficiently novel, the decision to discard the dominated solutions is validated.

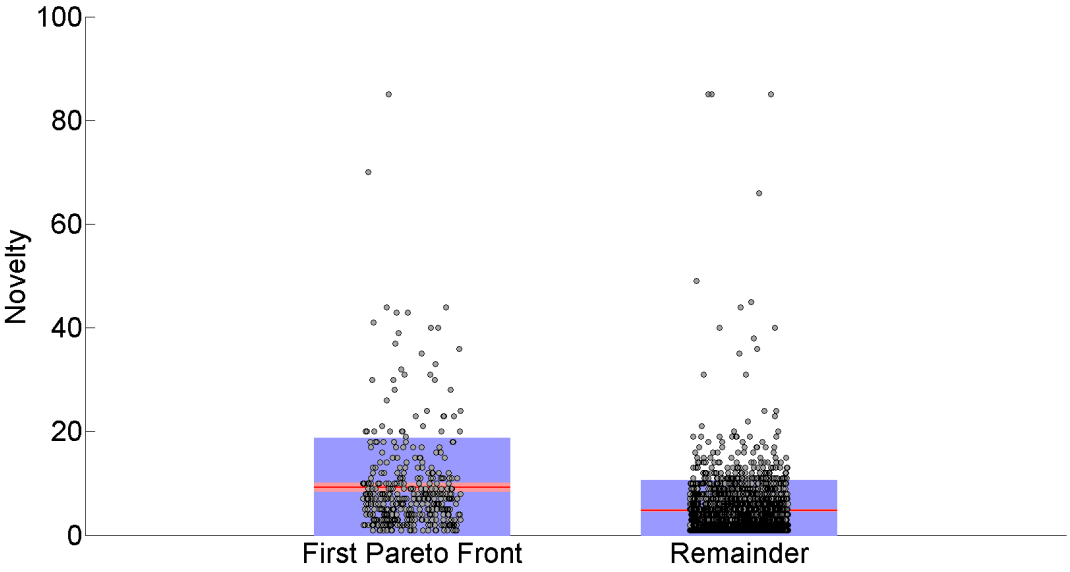
These results are graphically displayed in Figure 14, with a comparison of points across all runs (Figure 14a) and a comparison in a typical run (Figure 14b). The comparison of all points shows that the remainder of the population does, indeed, have some novel individuals. This is certainly the case as there would be individuals that are dominated by a very novel solution in the first PF that are still novel. It is the case, however, that there are *many* more points clustered at low novelty scores than in the first PF. Additionally, the first PF seems to have a higher density of very novel individuals than does the remainder of the population. This is more easily seen in the typical run (Figure 14b), as there are fewer data points. Here, it is apparent that the first PF has a much less dense cluster at low novelty scores than does the remainder of the population. This once again validates the approach of keeping only the first PF and discarding the remainder of the population.

5.6 Pareto Optimality Pointing Vector

As one of the primary motivations for using MOEA assistance for IEC, the examination of the POPV should yield insight into the human's intuitions about the problem domain. Figure 15 shows the POPV broken down into its five component objective functions across the entirety of a typical POA-IEC run. At first, the POPV starts as a vector with each element initialized to 0.5 for each objective function. Intuitively, any time a POPV component drop below 0.5, the chosen individual is worse than the average individual seen to that point for that objective function. Therefore,



(a) All Runs



(b) Typical Run

Figure 14. Pareto Front Novelty Comparison - Comparing the novelty of the first Pareto Front with the novelty of the remainder of the population demonstrates the similarites between behavioral diversity and fitness diversity. Figure 14a shows the conglomeration of all experimental runs, with the points indicating an individual in a run. 14b provides the same information for a typical run. In each case, the first PF is significantly more novel ($p < 0.001$; Student's t-test) than the remainder of the population.

the chosen individual is being *deceived* in that objective function as there are individuals that are more fit in that dimension. This chart tells a story of how the user

selected individuals while simultaneously providing insight into the deceptive nature of the domain.

The first individual that the user selects (User Interaction Number 1) creates a new vector for the POPV. In this new vector, the *Distance to Goal*, *Path Length*, and *Robot Speed* heuristics each had their POPV component drop below 0.5, implying that all three of these objective functions were being deceived in the first user selection. In fact, the *Distance to Goal* heuristic is so deceived, that it can be inferred that the user has selected a path that leads far away from the goal. However, the *Path Area* and *Path Smoothness* POPV components each rose very high, implying that these two objective functions were extremely important to the user at this point. From these values, it can be inferred that the user has selected a slightly short, high area, smooth, slower than average solution that leads far away from the goal, compared to all solutions seen to this point as the POPV is a relative vector.

Next, the individual selects an individual that deceives the same three objective functions as the first, while the path smoothness remains important and the area component drops in significance. The *Distance to Goal* heuristic is less deceived relatively than the first individual chosen, implying that the distance to goal became more important than it was with the first selection. Speed and path length did not change in importance, implying that they are not important yet. From this selection, we can infer that the user has selected a slightly short, above average area, smooth, slower than average solution that leads closer to the goal than the first chosen individual.

Third, the POPV implies that only one out of the five objective functions is deceived, path length. This selection shows a major increase in the importance of the distance to the goal and the path area, perhaps implying that the user has navigated the narrow channel in the middle of the maze and reached the large chamber of the

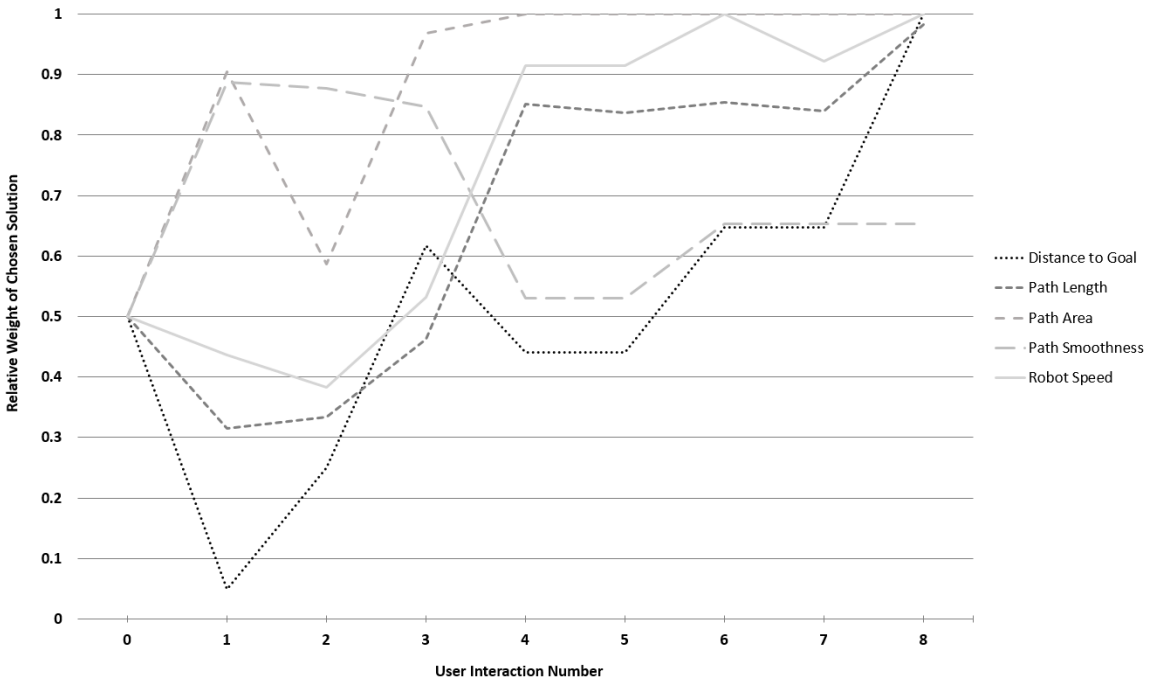


Figure 15. Pareto Optimality Pointing Vector During a Typical Run - The Pareto Optimality Pointing Vector (POPV) is a weight vector that indicates the relative weight of each objective function chosen at each human interaction. This chart shows the POPV changes during a typical run, broken into its component objective functions. In this particular run, 8 individuals were chosen and their POPVs were saved.

deceptive maze. Path smoothness did not change much between selection, while robot speed rose to be above average and the path length component is barely below 0.5.

Next, the user chose an individual that deceives only one objective function, this time distance to goal. Additionally, from now until the end of the run, the path area component is maximized at 1. This means that the user chooses individuals that have the highest area explored until the end of the run. From these two heuristics, we can infer that the user has chosen a solution that reaches near the top right corner of the maze, which would deceive the distance to goal heuristic but maximize area explored. Additionally, the path length and robot speed heuristics both show dramatic increases in importance, implying that the path length is one of the longest seen yet, further supporting the idea that this selection reached the top right corner of the maze, and that it does not stop, which would impact the robot speed score.

The next user selection does not change any of the POPV components, implying a similar (or even the same) selection as the previous step. Perhaps no significant progress was made in the child generation, which would compel the user to select a similar behavior as the next evolutionary step.

The next selection, however, is the first selection that does not deceive any of the objective functions, which are all above 0.5. As the distance to goal component has risen above 0.5, it can be inferred that the user has selected an individual which reaches into the chamber of the deceptive maze where the goal resides, but it does not touch the goal. The next iteration shows similar behaviors implying that the user selected the same or a similar individual because no progress was made.

The final step shows four out of the five objective function components converging to be near or at 1. This shows that reaching the goal maximizes fitness on each objective function such that no other solution has achieved a score as high. The path smoothness objective is the only one that does not converge to one. This intuitively happens because a perfectly smooth path is one with no curves, which are necessitated by the structure of the maze. It is not, however, deceived by the solution.

Of note is the fact that at least one objective function was deceived for 5 out of the 8 user selections. This shows the importance of the non-dominated set in avoiding deception as the user's selected individual was maintained in the population because it was fit in a different aspect. In this way, the PF preserves diversity.

Figure 16 shows an estimate of what the user selections likely were, based on what was inferred from the POPV graph in Figure 15. The actual, final path of the solution is shown in red overlaid on the actual distribution of the run that generated the POPV graph. Possible selection points are shown by a blue diamond. The amount of information that can be gleaned from the POPV is significant, allowing the user's intuitions to be quantified.

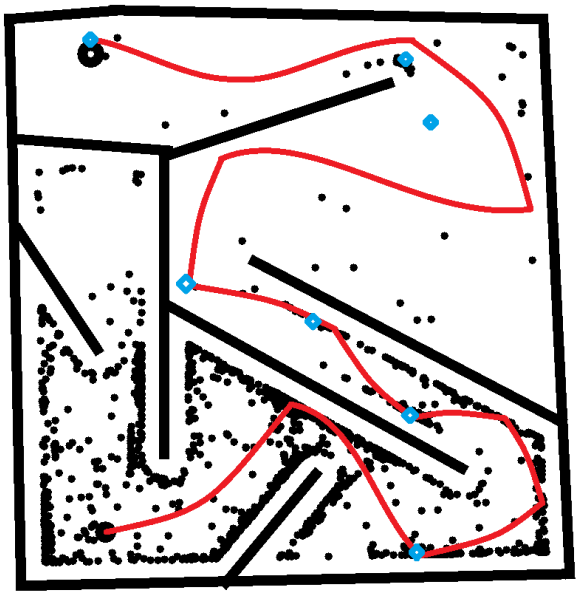


Figure 16. Pareto Optimality Pointing Vector Inferred Selections - This diagram shows the actual distribution from which the POPV is Figure 15 derived. The actual, final selection of the user is shown in red overlaid on top of the actual distribution of the same run. Based on the POPV changes, the inferred selections that the user made are indicated by blue diamonds.

5.7 User Interactions

Finally, it is important to analyze the human behaviors in the IEC context. Takagi [48] stressed the importance that fatigue is a high priority in an IEC system, a problem that NA-IEC and POA-IEC both attempt to reduce by providing short-term search assistance. As shown by Woolley and Stanley [51], NA-IEC reduced the number of user operations (i.e. mouse clicks selecting various evolutionary options) from an average of over 1,000 operations for the deceptive maze to an average of 32.0 operations (sd = 23.5) using NA-IEC. POA-IEC further reduces this number ($p < 10^{-5}$; Student's t-test), with users finding solutions in an average of just 10.26 operations (sd = 5.81). Of these operations, 94.97% were *Pareto* operations and 5.03% were *Optimize* functions. Similar to the results with NA-IEC [51], *Optimize* functions were applied most often near the end of an evolutionary run when a user was improving an

established solution to reach the end. Such results show that the MOEA assistance in POA-IEC speeds evolution and reduces the required number of human interactions.

VI. Conclusion

As demonstrated by Woolley and Stanley [51], IEC paired with short-term automated search has a synergistic effect that reduces evaluation and operation count in the deceptive maze. The MOEA assistance introduced here as a surrogate for novelty maintained behavioral diversity while simultaneously providing a measurable, learnable heuristic to improve the quality of solutions presented to the user. As is common for MOEAs, deception can be mitigated in a problem domain through the development of several simple objective functions. While they are not a complete picture and will not capture all of the subtleties of a domain, the combination allows evolution to side-step major deceptive pitfalls by preserving behavioral diversity. Additionally, while it can be difficult for a human to express what they like about a certain solution, a variety of objective functions define a behavior vector that can be associated with a given solution. This behavior vector preserves information that can articulate what those preference might be.

An interesting discovery from the experimental results was that the automated MOEA and POA-IEC outperformed pure novelty and NA-IEC respectively. Such findings suggest that the maze is no longer "hard enough". That is not to say that POA-IEC is a panacea, because there will always exist a case where a fitness-based search (even a multi-objective one) can be deceived. It is plausible to imagine that a deceptive maze could be created that cause MOEA and POA-IEC to perform at a lower level due to the nature of that domain.

Another insight from the experimental results is that the POPV provides a picture of the human user's thought process during evolution. The ability to track user selections provided such insight that the user's selections could be roughly reconstructed. Additionally, the POPV provided specific information about which objective functions were being deceived at various points in the maze. This gives precise details

about the domain, such that a "maze profile" could be generated as a metaheuristic for automated search. In other words, the user's selections could allow a profile to be built that would allow an automated process to "click" for the user, providing the same harsh selection criteria that speeds IEC.

Another interesting realization was that MOEA outperformed POA-IEC in time. In previous experiments, the addition of a human evaluator helped evolution to such an extent that the time spent visually processing information overcame the raw speed of an automated search [51]. In this case, the MOEA outperformed the human-led search evaluator in wall-clock time, but still expended more evaluations than both POA-IEC and NA-IEC. In a more difficult maze (e.g. larger with a longer solution path), it is possible that the search space would be expanded enough that a human evaluator is once again be needed to contribute their insights, causing POA-IEC to outperform a fully automated MOEA search in wall-clock time. Perhaps the true power of POA-IEC may only be properly revealed in a more difficult domain.

Finally, it can be concluded that, in this implementation, fitness diversity and behavioral diversity are synonymous. This is supported by the observation that the novelty scores of the first PF are significantly higher than those of the remainder of the population, implying that the PF is a diverse population. Subjectively speaking, it is clear from Figures 10 and 11 that the PF presents a diversity of behaviors to the user, even at the midpoint of the run. In traditional fitness approaches, the behaviors quickly converge to a single phenotype, eliminating all behavioral diversity. Without using speciation, the MOEA assistance provides the necessary phenotypic diversity to maintain a useful population for the user.

In addition to this, the endpoint distributions of the various runs show similar behaviors (Figure 13). In these distributions, the automated MOEA explores the maze in a very similar manner to that of the automated novelty search. Similarly,

POA-IEC and NA-IEC share similar exploration behaviors. This implies that MOEA (and thus MOEA assistance) provide many of the same stepping stones that novelty search does, creating a similar search pattern in the maze and providing the assistance to speed IEC. What is surprising, however, is the extent to which MOEAs improve on the evaluation count of novelty search. This is perhaps because the multiple objective functions provide the evolutionary process with enough information about the domain that it is a more capable algorithm for evolution. A more difficult maze would once again help test this idea.

6.1 Future Work

There are multiple avenues of further research into this area. Of particular interest would be to implement one of the many-objective algorithms to test its impact on POA-IEC. That is, since there are currently five objectives, this experiment technically fell into the many-objective (not multi-objective) category. Therefore, a more capable MOEA to emulate would be NSGA-III [10, 21, 53] rather than NSGA-II [11]. However, NSGA-II was chosen to implement because of its simple algorithm. It is surprising then, even with an algorithm known to struggle as the number of objective functions exceeds three, that POA-IEC should perform at such a level. Intuitively, an NSGA-III-like assistance mechanism could only help reduce evaluations and runtime further.

Additionally, research into "more difficult" mazes could answer the question as to whether the multi-objective aspects of POA-IEC are providing a greater benefit than novelty in the general case. It is possible that, as the maze grows larger and more deceptive, the search space expands and MOEAs would spend more time searching unproductive areas of the maze. In particular, this could influence the fact that MOEAs were faster in wall-clock time than were POA-IEC and NA-IEC, contrary

to the results reported by Woolley and Stanley [51] where NA-IEC was significantly faster in wall-clock time than novelty search. In a larger, more complex domain, the benefit of human guidance could overcome the raw speed of an automated MOEA.

Furthermore, an interesting extra function to build into the interface would be a *POPV* search button. This button would perform an aggressive search based on the POPV, essentially selecting an individual and clicking the button for the user based on the POPV. The user would be able to provide a limit on the number of "automated clicks", allowing the user to skip a number of user interactions. To account for the fact that the user might choose from the top n solutions, some work would need to be done to identify how often the user selects which individual. That is, determine the probability that the user selects the top POPV-sorted candidate versus the second POPV-sorted candidate and so on. Intuitively, this would follow a Gaussian distribution with the user selecting the top POPV-sorted candidate the largest percentage of the time. Should this be the case, the POPV-based search should click on the top n individuals following the same empirically determined probability scheme. In other words, the automated search should select the top POPV-sorted candidate with the same probability the user does and so on. This functionality can be essentially described as "automating the human," using the human's previous selections as a guide. This would theoretically reduce the number of human interactions further, though a larger maze might once again be needed to determine the significance of this feature.

Lastly, exploration into non-trivial domains to test the efficacy of this framework in generating useful, autonomous behaviors could prove fruitful in furthering the Air Force Research Labs goal of autonomy. Perhaps applying this framework to the problem of Unmanned Aerial Vehicle swarming or flocking behaviors would be an effective domain where human insights are particularly fruitful. Any autonomous task that can be represented visually for the human has potential for harnessing his

or her intuitions, allowing evolutionary computing to create effective autonomous behaviors.

6.2 Final Remarks

This thesis introduced the *pareto optimality-assisted interactive evolutionary computation* (POA-IEC) approach. The framework simultaneously generates phenotypic (i.e. behavioral) diversity while harnessing human intuition to create a synergistic effect in which the algorithm learns human preferences during evolution and presents the user with candidate behaviors most similar to those he or she just chose. In this manner, the approach improves upon previous fitness-based [3, 19] and novelty-based [51] methodologies by incorporating learnable heuristics into the approach. This opens new avenues for further exploration into additional automation possibilities by developing a metaheuristic that models human selections in a complex domain. The framework allows for the definition of objective functions that simplify a domain, thus reducing the requirements on the algorithm at work by harnessing the power of human intuition to combine the objective functions in meaningful ways. The objective functions provide insight to the domain via measurement of human selections, providing impetus for further fatigue reduction techniques based on automation of human selections. The results show that the approach reduces fatigue over previous approaches, accelerating the process of finding high-quality solutions without requiring step by step evaluation of every candidate.

Bibliography

1. T Back, Ulrich Hammel, and HP Schwefel. Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

2. I A Basheer and M Hajmeer. Artificial Neural Networks: Fundamentals, Computing, Design, and Application. *Journal of Microbiological Methods*, 43: 3–31, 2000.

3. Josh C. Bongard and Gregory S. Hornby. Combining Fitness-based Search and User Modeling in Evolutionary Robotics. In *Proceeding of the 15th Annual conference on Genetic and Evolutionary Computation Conference - GECCO '13*, pages 159–166, New York, New York, USA, 2013. ACM Press.

4. Chrislb. Artificial Neuron Model, 2005. URL `http://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png`.

5. Chrislb. Multilayer Perceptron, 2010. URL `http://commons.wikimedia.org/wiki/Neural_network#mediaviewer/File:MultiLayerNeuralNetwork.png`.

6. Carlos A Coello Coello, Gary B Lamont, and David A Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer, 2007.

7. Richard Dawkins. *The Blind Watchmaker*. Longman, 1986.

8. Kalyanmoy Deb. Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(1995):205–230, 1999.

9. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Ltd, 2001.

10. Kalyanmoy Deb and Himanshu Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2013.

11. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

12. J Doucette. *Novelty-Based Fitness Measures in Genetic Programming*. PhD thesis, Dalhousie University, 2010.

13. Scott Draves. Electric Sheep, 1999. URL `http://www.electricsheep.org/`.

14. Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: From architectures to learning, January 2008.

15. David Goldberg and John Holland. Genetic Algorithms and Machine Learning. *Machine Learning*, 3:95–99, 1988.

16. Jorge Gomes, Paulo Urbano, and Anders Lyhne Christensen. Evolution of Swarm Robotics Systems with Novelty Search. *Swarm Intelligence*, 7(2-3): 115–144, April 2013.

17. Amy K. Hoover, Michael P. Rosario, and Kenneth O. Stanley. Scaffolding for Interactively Evolving Novel Drum Tracks for Existing Songs. In *Proceedings of the Sixth European Workshop on Evolutionary and Biologically Inspired Music, Sound, Art and Design*, pages 412–422, 2008.

18. Amy K Hoover, Paul a Szerlip, and Kenneth O Stanley. Interactively Evolving Harmonies through Functional Scaffolding. *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO 2011*, pages 387–394, 2011.

19. Gregory S. Hornby and Josh Bongard. Accelerating Human-Computer Collaborative Search through Learning Comparative and Predictive User Models. In *Proceedings of the 14th annual conference on Genetic and Evolutionary Computation - GECCO 2012*, pages 225–232, New York, New York, USA, 2012. ACM Press.

20. AK Jain, JC Mao, and KM Mohiuddin. Artificial Neural Networks: A Tutorial. *Computer - Special issue: neural computing: companion issue to Spring 1996 IEEE Computational Science & Engineering*, 29(3):31–44, 1996. ISSN 0018-9162. doi: 10.1109/2.485891.

21. Himanshu Jain and Kalyanmoy Deb. An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, 2014.

22. Derek James. ANJI, 2005. URL `http://anji.sourceforge.net/`.

23. Y. Jin. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Computing*, 9(1):3–12, October 2005.

24. ED De Jong. The Incremental Pareto-Coevolution Archive. *Proceedings of the 6th annual conference on Genetic and Evolutionary Computation - GECCO 2004*, pages 525–536, 2004.

25. Steijn Kistemaker and Shimon Whiteson. Critical Factors in the Performance of Novelty Search. *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation - GECCO 2011*, pages 965–972, 2011.

26. JD Knowles, RA Watson, and DW Corne. Reducing Local Optima in Single-Objective Problems by Multi-objectivization. *Evolutionary Multi-Criterion Optimization*, pages 269–283, 2001.

27. Abdullah Konak, David W. Coit, and Alice E. Smith. Multi-Objective Optimization Using Genetic Algorithms: A Tutorial. *Reliability Engineering and System Safety*, 91(9):992–1007, September 2006.

28. Joel Lehman and K. O. Stanley. Revising the Evolutionary Computation Abstraction: Minimal Criteria Novelty Search. *Proceedings of the 12th annual conference on Genetic and Evolutionary Computation - GECCO 2010*, pages 103–110, 2010.

29. Joel Lehman and K. O. Stanley. Novelty Search and the Problem with Objectives. *Genetic Programming Theory and Practice IX*, pages 37–56, 2011.

30. Joel Lehman and Kenneth O Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Artificial Life XI*, pages 329–336, 2008.

31. Joel Lehman and Kenneth O. Stanley. Efficiently Evolving Programs Through the Search for Novelty. *Proceedings of the 12th annual conference on Genetic and Evolutionary Computation - GECCO 2010*, pages 837–844, 2010.

32. Joel Lehman and Kenneth O Stanley. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2): 189–223, 2011.

33. Klaus Meffert. JGAP, 2012. URL `http://jgap.sourceforge.net/`.

34. Jean Baptiste Mouret. Novelty-Based Multiobjectivization. In *Studies in Computational Intelligence*, volume 341, pages 139–154, 2011.

35. Enrique Naredo and Leonardo Trujillo. Searching for Novel Clustering Programs. *Proceedings of the 15th annual conference on Genetic and Evolutionary Computation - GECCO 2013*, pages 1093–1100, 2013.

36. S. Risi, C. E. Hughes, and K. O. Stanley. Evolving Plastic Neural Networks with Novelty Search. *Adaptive Behavior*, 18(6):470–491, 2010.

37. JJ Romero and P Machado. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Springer, 2007.

38. Michael D. Schmidt and Hod Lipson. Actively Probing and Modeling Users in Interactive Coevolution. *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO 2006*, pages 385–386, 2006.

39. Jimmy Secretan, Nicholas Beato, David B D'Ambrosio, Adelein Rodriguez, Adam Campbell, and Kenneth O. Stanley. Picbreeder: Evolving Pictures Collaboratively Online. In *Proceedings of the 26th annual CHI conference on human factors in computing systems - CHI 2008*, pages 1759–1768, 2008.

40. Jimmy Secretan, Nicholas Beato, David B D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T Folsom-Kovarik, and Kenneth O Stanley. Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evolutionary Computation*, 19(3):373–403, 2011.

41. Paul Sherman. Biological Neuron Diagram, 2013. URL `http://www.wpclipart.com/medical/anatomy/cells/neuron/neuron_label_parts.png.html`.

42. Karl Sims. Artificial Evolution for Computer Graphics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques - SIGGRAPH 1991*, pages 319–328, 1991.

43. Karl Sims. Genetic Images, 1993. URL `http://www.karlsims.com/genetic-images.html`.

44. Karl Sims. Evolving Virtual Creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH 1994*, pages 15–22, 1994.

45. Karl Sims. Galápagos, 1997. URL `http://www.karlsims.com/galapagos/index.html`.

46. Kenneth O. Stanley and Risto Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary computation*, 10(2):99–127, 2002.

47. Kenneth O. Stanley and Risto Miikkulainen. Competitive Coevolution through Evolutionary Complexification. *Journal of Artificial Intelligence Research*, 21 (1):63–100, 2004.

48. Hideyuki Takagi. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

49. Nao Tokui and H Iba. Music Composition with Interactive Evolutionary Computation. In *Proceedings of the Third International Conference on Generative Art*, pages 215–226, 2000.

50. Brian G. Woolley and Kenneth O. Stanley. On the Deleterious Effects of A Priori Objectives on Evolution and Representation. *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation - GECCO 2011*, pages 957–964, 2011.

51. Brian G Woolley and Kenneth O Stanley. A Novel Human-Computer Collaboration: Combining Novelty Search with Interactive Evolution. In *Proceedings of the 16th annual conference on Genetic and Evolutionary Computation - GECCO 2014*, pages 233–240, 2014.

52. Hao Yu and Bogdan Wilamowski. LevenbergMarquardt Training, 2010. URL `http://www.eng.auburn.edu/~wilambm/pap/2011/K10149_C012.pdf`.

53. Yuan Yuan, Hua Xu, and Bo Wang. An Improved NSGA-III Procedure for Evolutionary Many-objective Optimization. In *Proceedings of the 16th annual conference on Genetic and Evolutionary Computation - GECCO 2014*, pages 661–668, 2014.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
03–26–2015	Master's Thesis	Sept 2013 — Mar 2015

4. TITLE AND SUBTITLE

Leveraging Human Insights by Combining
Multi-Objective Optimization with Interactive Evolution

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Christman, Joshua, R, 2d Lt, USAF

5d. PROJECT NUMBER

15-200A

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-15-M-060

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Lincoln Laboratory
Massachusetts Institute of Technology
244 Wood Street
Lexington, MA 02420-9108
Dr. Marc Viera, mviera@ll.mit.edu
781-981-1077

10. SPONSOR/MONITOR'S ACRONYM(S)

MIT-LL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Deceptive fitness landscapes are a growing concern for evolutionary computation. Recent work has shown that combining human insights with short-term evolution has a synergistic effect that accelerates the discovery of solutions. While humans provide rich insights, they fatigue easily. Previous work reduced the number of human evaluations by evolving a diverse set of candidates via intermittent searches for novelty. While successful at evolving solutions for a deceptive maze domain, this approach lacks the ability to measure what the human evaluator identifies as important. The key insight here is that multi-objective evolutionary algorithms foster diversity, serving as a surrogate for novelty, while measuring user preferences. This approach, called Pareto Optimality-Assisted Interactive Evolutionary Computation (POA-IEC), allows users to identify candidates that they feel are promising. Experimental results reveal that POA-IEC finds solutions in fewer evaluations than previous approaches, and that the non-dominated set is significantly more novel than the dominated set. In this way, POA-IEC simultaneously leverages human insights while quantifying their preferences.

15. SUBJECT TERMS

Evolutionary computation, interactive evolutionary computation, human-led search, fitness, pareto, pareto front, multi-objective, multi-objective evolutionary algorithm, non-dominated set, deception, non-objective search, novelty search, stepping stones, serendipitous discovery

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj. Brian G. Woolley, AFIT/ENG
U	U	U	UU	77	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4618; brian.woolley@afit.edu