MULTI-OBJECTIVE CULTURAL ALGORITHMS

by

CHRISTOPHER BEST

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2009

MAJOR: COMPUTER SCIENCE

Approved by:

_____

Advisor Date

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Robert G. Reynolds, for his efforts and enthusiasm with regards to this project. I would also like to my committee members, Dr. Ming Dong and Dr. Nathan Fisher for their work on my behalf. I would also like to thank my family for their support throughout this process.

DEDICATION

For mom, happy mother’s day. For Asil, Kin, Lliby, Dad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Multi-objective optimization

Multi-objective optimization is the problem of producing solutions which map to good values for several different and conflicting functions. Because they judge solutions according to more than one criterion, multi-objective problems are more expressive and complex than single-objective problems. Consequently, the task of solving them is more difficult. Multi-objective optimization is important because solutions to many real-world problems are viewed from a variety of perspectives, and cannot be expressed using only one objective.

A multi-objective problem is formulated as

Minimize $F(\vec{x}) = (f_1(\vec{x}), \dots, f_M(\vec{x}))$

subject to $g_i(\vec{x}) \leq 0, i = 1, \dots, M$

We compare solutions to multi-objective problems using the notion of dominance. For an m-objective minimization problem, a solution x_1 dominates x_2 if $\forall i = 1, \dots, m,\ f_i(x_1) \leq f_i(x_2)$ and $\exists i \in \{1, \dots, m\} \mid f_i(x_1) < f_i(x_2)$. Therefore, we can only consider solution x better than y if x dominates y. Two solutions which do not dominate each other are said to be incomparable. In this case we cannot deem one to be better than the other unless we are able to prioritize the objective functions. Given a set of solutions $X = \{x_1, \dots, x_n\}$, we say x_i is non-dominated in the set if $\nexists x_j \in X \mid x_j$ dominates x_i. A solution x is considered Pareto Optimal or Pareto Efficient if $\nexists y \mid y$ dominates x for any y in the search space. In order for a Pareto optimal solution to decrease one objective, it must incur an increase in another. We call the set of Pareto optimal solutions is called the

Pareto Set, and the corresponding vectors in objective space are called the Pareto Front [1].

Many techniques have been developed to reduce multi-objective problems to single-objective problems so that single-objective optimization algorithms, such as CA, can be applied. However, we are interested in well formed multi-objective problems, that is, problems which contain no solutions which are optimal for all objective functions. Well formed problems cannot be reduced to an equivalent single-objective problem. Therefore, we must modify CA to natively consider multi-objective problems.

1.2 Cultural Algorithms

In 1979, Reynolds introduced Cultural Algorithms (CA), an extension of Genetic Algorithms [2]. While Genetic Algorithms simulate the process of evolution to optimize a function, Cultural Algorithms recognize interaction between individuals beyond genetic recombination and mutation. Individuals in a population communicate and interact to form a culture, first defined by Edward B. Tylor as “that complex whole which includes knowledge, belief, art, law, morals, custom, and any other capabilities and habits acquired by man as a member of society.” The process by which information is shared between individuals which allows culture to emerge is immensely complex and we do not hope to model it. Instead, we model culture at a high level, examining what type of information is shared, and how individuals are influenced by it. Cultural Algorithms are a functional model of the process by which human fitness is improved through knowledge sharing in a population. The CA has been successful in optimizing a variety of problems. It has also been used to assess how humans have solved Multi-Objective problems relative to site location [3].

Given the success of the CA, we investigate whether a cultural model can be used to solve multi-objective optimization problems. We are encouraged by the abundance of mechanisms within human culture to aid the search for solutions to multi-objective problems faced by individuals. Consider a community of individuals settling a small island, immediately faced with the problem of where to take up residence. They will want to find land with fertile soil to produce crops, a source of fresh water to drink, and an area in which to hunt. Almost certainly, no single place on the island will provide an optimal location for all of these activities. Instead, individuals will begin settling in locations conducive to one or two of the goals. By sharing information about the landscape, the community will avoid locations which are worse for all three goals than some other location. In this sense, culture has helped guide the community onto the Pareto set of locations on the island. It is possible, then, that a functional model of culture can be used to solve multi-objective optimization problems as they can for single-objective optimization problems.

Using CA methods to solve multi-objective problems has been proposed before with some success [3] [4]. However, Coello Coello's CAEP was limited in scope to one type of knowledge source, and CA was used only as a small component of the algorithm. The current approach is to modify CA component-wise to apply all of its mechanisms to multi-objective problems.

In chapter 2 we discuss the design and mechanics of Cultural Algorithms so that we may understand how we can adapt them to solve multi-objective problems. Chapter 3 describes the changes that we made to produce the Multi-Objective Cultural Algorithms (MOCA), and our implementation of the components of the system. We also show a

sample run to exhibit the mechanics of the algorithm. In chapter 4, the test problems and system parameters used to evaluate the performance of MOCA are presented. Chapter 5 gives the results of the experiments and compares MOCA to other multi-objective evolutionary algorithms. The conclusions are given in chapter 6.

CHAPTER 2

SINGLE-OBJECTIVE CULTURAL ALGORITHMS

In this chapter the Cultural Algorithms Toolkit (CAT) is described, and the adjustments that need to be made in order to support MO problems are discussed. To start, the Cultural Algorithm system consists of several components: the Belief Space, the Population, the Acceptance Function, and the Influence Function. In the CAT system, each are implemented in as simple a fashion as possible. This allows us to add complexity back into the system incrementally. Now now each component is briefly discussed.

The five knowledge sources used here are characterized by their ubiquity in the problem solving process. Each can be used in some form in most problems. Also, there is a basis for the presence of each knowledge type in pre-human species as well. Thus, the knowledge sources used do not have to be viewed as unique to the human species.

In the earliest Cultural Algorithms only one knowledge source was used in the Belief Space. This was typically tailored to direct the search for the problem at hand. Situational knowledge was used first, then normative, topographic, domain, and finally history knowledge in succession. Their additions reflect an evolution in the complexity of the problems to which Cultural Algorithms were applied. The knowledge sources are described here in terms of their ability to coordinate the spread of individuals over the landscape for a problem.

2.1 Topographical Knowledge

Topographical Knowledge was originally proposed to reason about region-based functional landscape patterns [5]. It can distribute individuals potentially over the entire

landscape. It was motivated in conjunction with data mining problems where the problem space was so large that a systematic way of partitioning the space during the search process was needed. If we view a state as associated with a region in the functional landscape then the topographic knowledge source is looking for new states. Thus, the state space may vary dynamically as new sub-regions are discovered and added to the mix.

2.2 Normative Knowledge

Normative Knowledge is a set of promising variable ranges that provide standards for individual behaviors, and guidelines within which individual adjustments can be made. Normative knowledge came into play during the learning of rules for expert system applications. Normative Knowledge directs individuals to “jump into the good range” if they are not already there. In other words it produces conduits or regional landing strips that guide the population in moving from one attractive region to the next. It therefore, spreads individuals into subregions of the space.

2.3 Domain Knowledge

Domain Knowledge uses knowledge about the problem domain in order to guide search. Domain Knowledge was first used to find the resource cone(s) of maximum height in a landscape by Saleem [6]. In this case, problem knowledge defines properties of the resource cones that make up the performance surface. For example, in a functional landscape composed of cones, knowledge about cone shape and related parameters will be useful in reasoning about them during the search process. In particular, domain knowledge can be used to generate individuals up and down slope from a location as a function of cone slope and height. Once a relatively productive area is found, the domain

knowledge can guide the exploitation of that region.

2.4 Situational Knowledge

Situational Knowledge provides a set of exemplary cases that are useful for the interpretation of specific individual experiences. Situational Knowledge leads individuals to “move toward the exemplars”. This was the earliest knowledge source used with Cultural Algorithms and was inspired by elitist approaches in Genetic Algorithm. This knowledge source collaborates with domain knowledge to exploit above average regions.

2.5 Historical Knowledge

Historical or Temporal knowledge monitors the search process and records important events in the search. This knowledge source was first used by Saleem [6] and expanded by Peng [7]. Individuals guided by History knowledge can consult those recorded events for guidance in predicting a good move direction. There is no dynamic component in the static examples used, so the variance associated with the knowledge source is a function of the local topography. In more dynamic problems it can distribute individuals over the entire space akin to topographic knowledge in order to monitor regions that were highlighted in the past.

2.6 Population model

In general, the population space can support any evolutionary algorithm to run in tandem with the influence from the belief space. In this paper, we omit any such algorithm running in the population space to study the effectiveness of the Cultural Algorithm. We consider problems for which individuals in the population are vectors of real numbers.

2.7 Acceptance function

The acceptance function determines which individuals and their behaviors can

impact the belief space knowledge. It is often specified as a percentage of the number of current individuals ranging between 1% and 100% of the population size, based upon selected parameters such as performance. For example, we can select the best performers (e.g. top 10%), worst performers (e.g. bottom 10%), or any combination.

2.8 Multi-objective considerations in cultural algorithms

In the CA, the Belief Space holds knowledge contributed by individuals in population, including information about the highest performing individual. Some knowledge sources use this information when generating new solutions. For a multi-objective problem, however, the highest performing individual is not defined, so there is no single best solution to model future solutions after. More generally, we cannot always strictly compare two solutions to a multi-objective problem. MOCA uses the widely used Pareto scheme to compare solutions. Specifically, we use the Goldberg ranking scheme, where all non-dominated solutions in the population are given rank 1. These solutions are removed, and the non-dominated solutions in the remaining population are given rank 2, and so on [8]. The total ordering on solutions is replaced by a partial ordering of individuals into Pareto ranks. Therefore, when we convert CA to handle multi-objective problems, the concept of “the best performing individual” is often replaced by “an individual chosen from the set of non-dominated individuals in the current population.” This choice can be random, or if the problem allows, we can apply a heuristic to suggest an auspicious individual.

A trivial conversion of the CA to handle multi-objective problems would be to maintain copies of the five knowledge sources for each objective. Conceptually, we can think of this as having a belief space specific to each objective. To influence an individual

in the population with a knowledge source, we would choose an objective and a type of knowledge source, then influence the solution with the copy of that knowledge source for that objective. This approach amounts to little more than running a single-objective cultural algorithm separately for each objective and combining the results. This can be viewed as a kind of co-evolutionary process, five different populations are running simultaneously and exchanging solutions. Running Cultural Algorithms for each objective would, in the best case, find dark corners of the true Pareto front. In the expected case the resulting solutions would be optimal for one objective and would not lie on the true Pareto front.

This conversion of CA can outperform a system of separate single-objective CAs, because individuals can be influenced based on different objectives in succession. Theoretically, this can drive solutions toward the Pareto front, but in practice, we find that solutions only reach the true front for simple, low-dimension problems. In order to find the true Pareto front, the objective functions must be considered collectively when knowledge sources influence individuals. In chapter 3 we describe how we modified each knowledge source in order to consider multiple objectives and their roles in guiding the search to the true Pareto front. We also explain how new individuals are selected into the population, and which are accepted to update the belief space.

CHAPTER 3

MULT-OBJECTIVE CULTURAL ALGORITHMS (MOCA)

MOCA, at the top level, is identical to CA. In this sense, we suggest that MOCA *is a* CA, and not an approach *based on* CA. A semanticist, then, might refer to Reynolds' original algorithm as a Single-Objective Cultural Algorithm. We make this distinction to clarify that MOCA and CA are identically motivated. We aim to approximate the mechanism by which human culture evolves solutions to complex problems.

As described in chapter 2 and shown in Figure 1, a Cultural Algorithm has two stores of information: the P*opulation Space*, holding a set of individual solutions, and the B*elief Space*, holding information and statistics collected from the population. The two stores interact through the *accept* and *influence* functions.

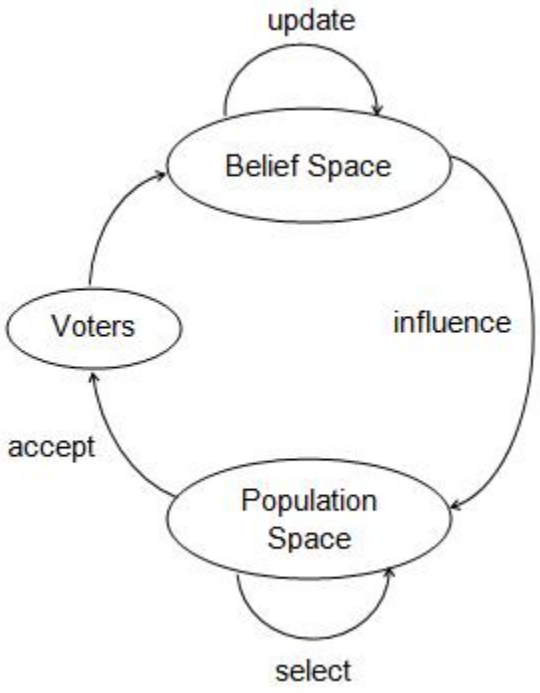

Figure 1: MOCA Structure

The *accept* function determines which individuals from the population are permitted to

contribute information to the belief space. Our implementation of *accept* is describe in section 3.3. We call the set of individuals accepted into the belief space the “voting” solutions. Once they have been identified, the *update* function extracts information needed by the five knowledge sources. In fact, each knowledge source implements its own update function which is called during *update*. The individual knowledge source update functions are described in section 3.1. The *influence* function then creates new solutions by modifying individuals in the current population using information from one of the five knowledge sources. Again, the knowledge sources each implement an influence function and *influence* selects one from a random distribution when it operates on a solution. The knowledge source influence functions are described in section 3.1, and the method by which they are chosen is described in section 3.2. *Select* considers the new individuals generated by the knowledge sources together with the current population and determines which will compose the next generation.

3.1 Knowledge Source Implementation

3.1.1 Situational Knowledge

The goal of the situational knowledge source is to model new individuals after the highest performing individuals currently in the population. To fulfill this role in a multi-objective setting, we recognize that the set of exemplary individuals are readily defined as those in the population which are non-dominated. These individuals are stored when the belief space is updated, and one is chosen to act as the exemplar when the knowledge source influences another individual. The Information about a problem may be exploited to intelligently decide which individual to choose from the Pareto front. For example, we may take the individual with the best value for an objective which the

population is not optimizing well. In the general case, we can select randomly from the front with acceptable results. The influence procedure is very simple. For a given parent p, and an individual e chosen from the non-dominated solutions, the situation knowledge source creates a child $c = p + (e - p) \cdot k$ where k is chosen randomly from $[0,1]$.

Figure 2 and Figure 3 show an example of situational knowledge influence. The example is a two-dimensional, two-objective minimization problem with $f_1(x) = |\overrightarrow{(1,1)} - x|$ and $f_2(x) = x_2$. The true Pareto front lies on $x_1 = 1$. The individuals in the current population are shown in blue and green, and are separated by rank. A solution with Pareto rank 2 is being influenced. The situational knowledge source has chosen a non-dominated solution and generates an individual on a line between the two solutions with $k = 0.5$. The new solution is non-dominated in the population.

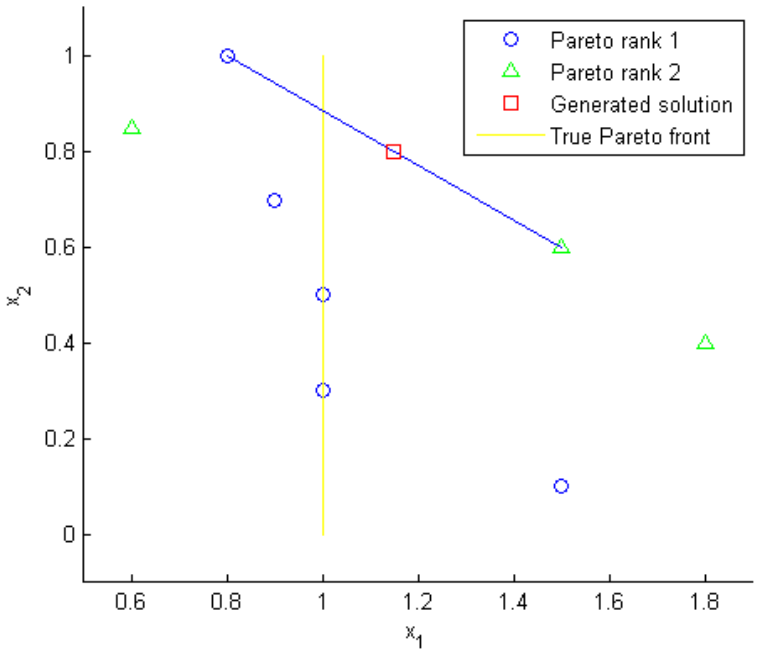

Figure 2: Situational knowledge influence, decision space

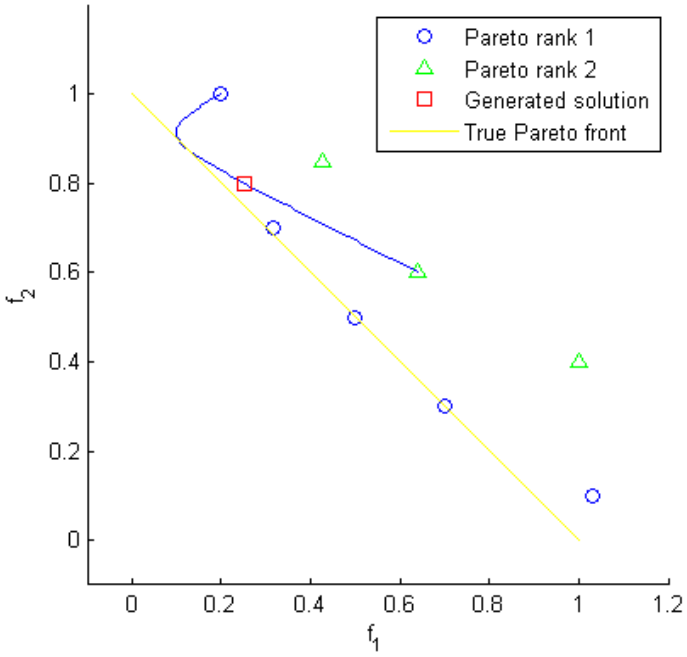

Figure 3: Situational knowledge influence, objective space

By design, the situational knowledge source influences individuals to produce children near known good solutions. We observe that, as a result, the situational knowledge source performs two major functions. The first is to search near the best solutions to improve them, honing the front if it is not optimal. This is also the role of the domain knowledge source. However, situational knowledge does so with a wider search space because it generates solutions between the Pareto front and a parent solution which may not be on the front. The other function of situational knowledge source is to distribute solutions over the true Pareto front once it is found. Other knowledge sources may encounter the true Pareto front by aggressively optimizing one objective. Situational knowledge influencing such a solution will generate a child likely to be on the front, but separated from the parent. Repetition of this type of influence over generations will create and maintain a distributed coverage of the front.

3.1.2 Domain Knowledge

The domain knowledge source is here intended to perform an incremental search of a region of the total search space. For a single objective problem, it is equivalent to a gradient search starting from a given individual. Over several generations of influence, it finds a solution which is at least locally optimal. To implement the domain knowledge source for multi-objective problems, we accept the analog to the single-objective local optimum as our goal: the knowledge source should search toward a solution which is locally Pareto efficient. To find the best direction in which to search, the domain knowledge source in the single-objective cultural algorithm produces 3^n individuals to surround the parent in n-dimensional search space. These 3^n individuals are constructed by increasing, decreasing, and maintaining the value for each dimension independently. The exponential number of individuals is prohibitive to solving high-dimensional problems. In MOCA, we implement a heuristic requiring only $2n+1$ to be examined to approximate a good search direction. Given a parent $\vec{p}$, we generate the $2n$ solutions:

$$(p_0, \dots, p_i \pm \varepsilon, \dots, p_{n-1}) \qquad \forall i \in \{0, \dots, n-1\}$$

From the above solutions, let us call the $m > 0$ which dominate the parent $d_0, \dots, d_{m-1}$. Let $u_i = \frac{d_i - p}{|d_i - p|}$, the unit vector in the direction of $d_i - p$. We sum the direction vectors to get $s = \sum_{i=0}^{m-1} d_i$. Since only 2 of the originally created solutions differ from p in any one dimension, we have $s_i \in \{-1,0,1\}, \forall i \in \{0, \dots, n-1\}$. Finally, we create one more solution $d_m = (p_0 + c_0 \cdot \varepsilon, \dots, p_{n-1} + c_{n-1} \cdot \varepsilon)$. If d_m dominates p, we create a child $c = p + (d_m - p)k$ for some k. If d_m does not dominate p, then we create a child $c = p + (d_j - p)k$ for some k and $j \in \{0, \dots, m-1\}$. Finally, if $m = 0$ and therefore none of the original $2n$ solutions dominated p, then the parent is at least locally Pareto efficient. In this case

we ignore the parent and influence a solution which is known not to be locally Pareto efficient. The domain knowledge source identifies such an individual during the *update* function.

Figure 4 and Figure 5 show an example of domain knowledge influence. This maximization problem has objective functions $f_1(x) = x_1^{0.1}\cos(2\pi x_1)(1 - x_2)$ and $f_2(x) = x_1^{0.1}\cos(2\pi x_1) \cdot x_2$. The true Pareto front lies at $x_1 = 1$ and a local Pareto front is located at $x_1 \approx 0.035$. The domain knowledge source searches in the four cardinal directions around the parent, resulting in the solutions in red, cyan and magent. The new solution positioned directly away from the true Pareto front dominates the other individuals produced because it lies near the local Pareto front. The domain knowledge is successfully searching toward the local Pareto front nearby at the expense of finding the true Pareto front.

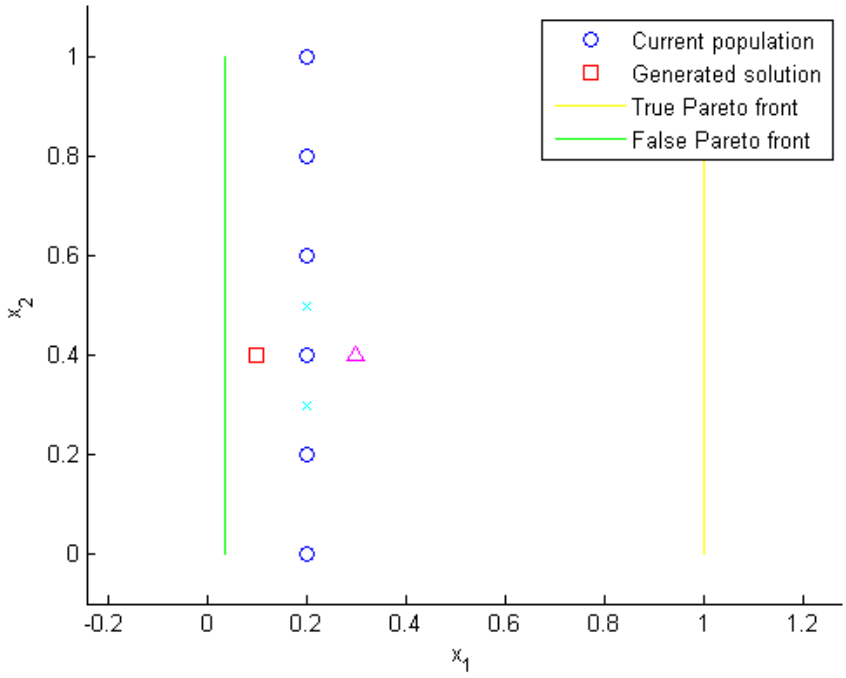

Figure 4: Domain knowledge influence, decision space

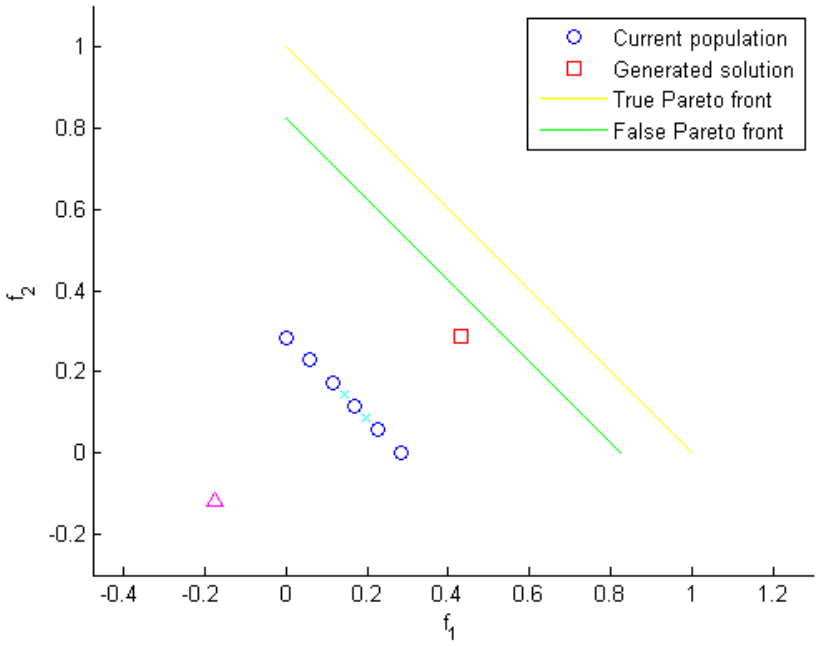

Figure 5: Domain knowledge influence, objective space

3.1.3 Normative Knowledge

There is an immediate conversion of the normative knowledge source to handle multi-objective problems. In a Cultural Algorithm, the normative knowledge source calculates the minimum and maximum values for each dimension over a set of high-performing individuals. For a multi-objective problem, an obvious choice for our set of high-performing individuals is the current non-dominated solutions. Normative knowledge, therefore, constructs a bounding box of the current Pareto front in the decision space. It provides a particularly promising region of the search space on which to focus new solutions.

In order to influence solutions in Cultural Algorithms, the normative knowledge source examines each dimension of the parent solution separately. If the parent's value for the given dimension lies within the normative range, we generate a value for the child nearby. Otherwise, we assign to the child a random value uniformly distributed over the range. In MOCA we adopt this influence algorithm, but augment it with another approach.

For each dimension, i, two copies of the parent are created, and their i^{th} values are set to the minimum and maximum values of the normative range. If exactly one of the two solutions dominate the parent, or one solution dominates both the parent and the other solution, its value for dimension i is assigned to the child. If both dominate the parent but neither dominates the other, we select one randomly and assign it's value to the child. If neither of the solutions dominate the parent, the child is given the parent's value.

In Figure 6 and Figure 7 we see the first type of influence of the normative knowledge source. The individuals in the current population are shown in blue and green, and are separated into Pareto ranks. The objective functions are $f_1(x) = |\overleftarrow{(0.5,0.5)} - x|$ and $f_2 = x_2$. The normative ranges for x_1 and x_2 are indicated by the rectangle. We can see that the bounding rectangle is formed by the minimum and maximum values of the individuals with Pareto rank 1. Since the x_1 value of the parent already lies in the normative range, the generated solution takes that value. A random value is chosen from the normative range for x_2, and we see the new individual in red in the bounding box.

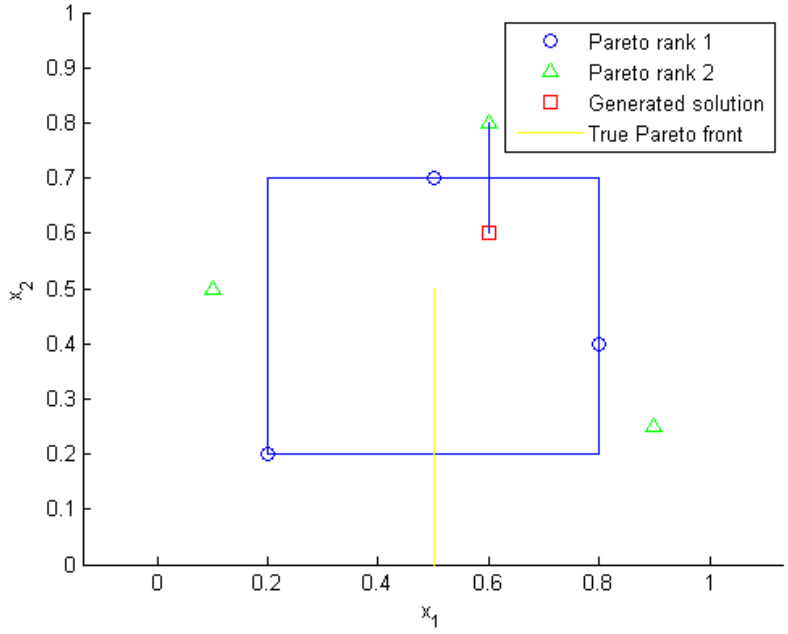

Figure 6: Normative knowledge influence, decision space

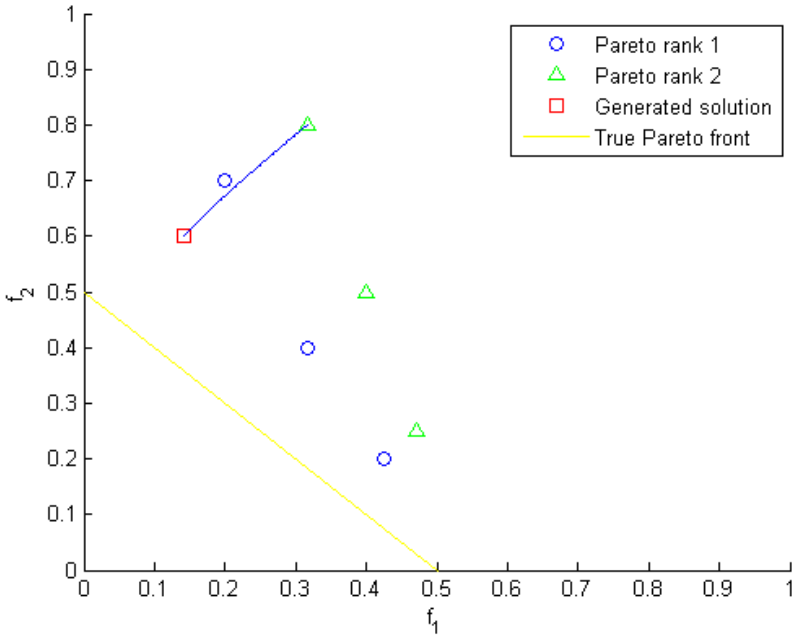

Figure 7: Normative knowledge influence, objective space

Normative knowledge combats the tendency of the domain knowledge source to converge to local optima. Given a well-distributed initial population, normative knowledge serves to keep the search space from excluding the true Pareto front. The normative ranges are initialized to encompass the entire decision space, and gradually contract to

include regions which produce fit individuals. Regions are eliminated only when they fail to provide good solutions, so the true Pareto front will not be bypassed as it might using only greedy information such as domain knowledge.

3.1.4 Historical Knowledge

The historical knowledge source tracks the progress of the elite individual in the population over the course of generations. New solutions, then, can be modeled not only after the best individuals in the current population, but from exemplary individuals from the past. Because the knowledge source depends heavily on the notion of a single best individual, we chose to maintain historical knowledge for each objective separately. For each objective f_i, a historical knowledge source H_i records the individual with the best f_i value for the past k generations. To use historical knowledge to influence an individual, we choose an objective f_i, and use knowledge source H_i to influence the solution as we do in a single-objective cultural algorithm: one of the k solutions stored by H_i is selected at random, and a solution is generated nearby.

Figure 8 shows the historical knowledge source generating a new solution. The problem has one decision variable and at least one objective function. We show H_1 influencing a solution based on the first objective function: $f_1(x) = \cos(4\pi x) + x$. We see the $k = 7$ best individuals from the preceding generations in blue, with solutions from older generations faded to white. The solution generated by H_1 is placed near the exemplar at 0.7 and is shown in red. Although it is approaching only a local optimimum for f_1, the solution may be Pareto optimal given the remaining objectives.

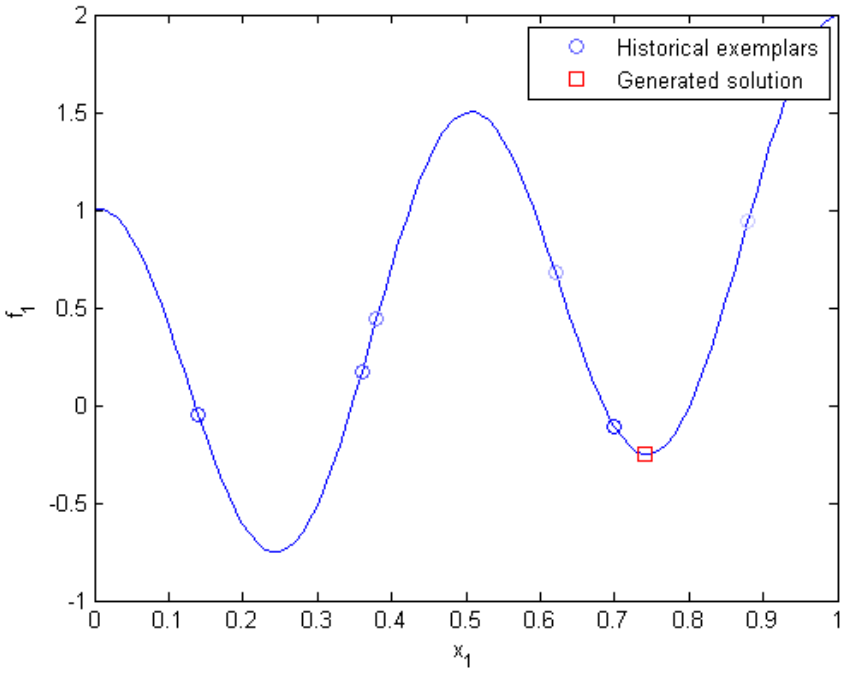

Figure 8: Historical knowledge influence

In MOCA, the historical knowledge source serves mainly to spread the distribution of solutions over the Pareto front once it is found. Since it doesn't actively seek to improve the fitness of solutions, most individuals generated from historical knowledge are soon dominated by improved solutions contributed by other knowledge sources. In fact, the presence of historical knowledge influence in a population can be a useful measure of progress made by the algorithm. Once the true Pareto front has been found, historical knowledge also helps to prevent regions of the front from being deserted by the other knowledge sources. After a sufficient number of generations has passed since the discovery of the true Pareto front, most or all of the historical exemplars will be on the true front. The historical knowledge source will continue to place solutions around them, preventing the search from abandoning the area.

3.1.5 Topographical Knowledge

Topographical knowledge aims to divide the search space and identify regions

which promise good solutions. It is similar to historical knowledge in that it has a strong dependence on the examination of a single objective function. In the case of topographical knowledge, a threshold value in the objective range is selected and used to judge members of the population. Therefore, as with historical knowledge, we implement the topographical knowledge source by maintaining one knowledge source, T_i, for each objective function f_i. Each T_i identifies a threshold value as the median fitness value of the individuals in the population. It then breaks the search space into regions recursively by dividing regions which contain solutions above and below the threshold. To generate a new solution, the topographical knowledge source selects a region containing good solutions, and produces an individual near the best from the region.

In Figure 9, we see an example of a solution being generated by the topographical knowledge source. The problem shown has one decision variable and at least one objective. We plot only the objective function being considered by the topographical knowledge source T_1: $f_1(x) = \cos(3\pi x) + x$. We see the threshold value $f_1 = 0.3$ in red. T_1 has divided the search space $[0,1]$ into the regions $\left[0,\frac{1}{8}\right], \left(\frac{1}{8},\frac{1}{4}\right], \left(\frac{1}{4},\frac{1}{2}\right], \left(\frac{1}{2},\frac{3}{4}\right], \left(\frac{3}{4},\frac{7}{8}\right], \left(\frac{7}{8},1\right]$. These divisions are shown in blue. Looking at the current population shown in blue, we can see that the regions are a result of recursively dividing regions in half until each contains only solutions above the threshold, or only solutions below the threshold.

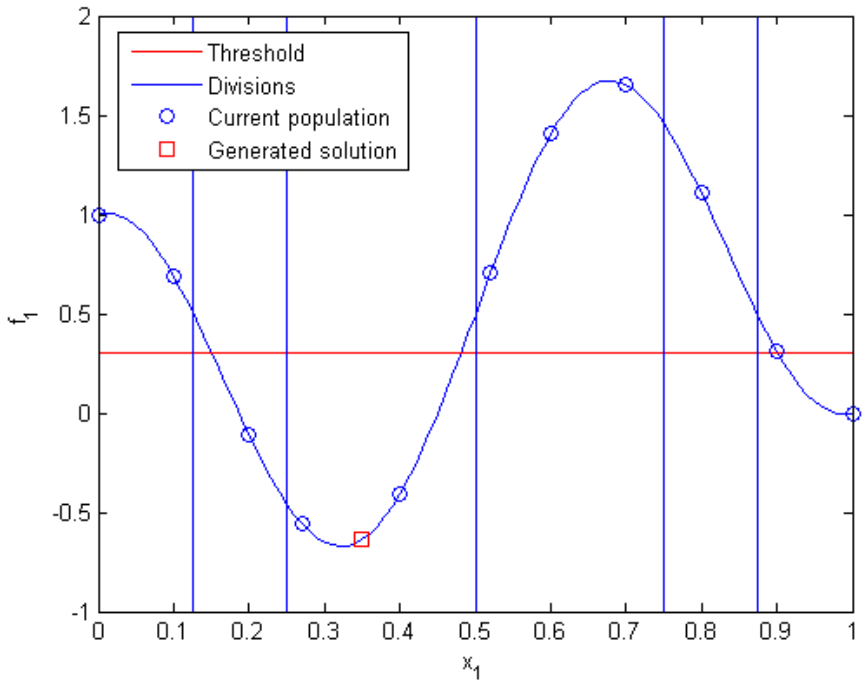

Figure 9: Topographical knowledge influence

Topographcal knowledge can help to avoid stalling on a local Pareto front, but without a good heuristic will search the entire space which will produce additional comparisons. The knowledge source also requires more processing time and allocates more memory than the others, increasing the clock-time required to optimize a problem. Therefore, for simplicity, we exclude the topographical knowledge source here when performing the experiments described in chapter 4.

3.2 Choosing Knowledge Sources

To help guide the optimization process, knowledge sources are selected in order to influence members of the population by sampling a dynamic probability distribution. As the optimization runs, we adjust the distribution to encourage knowledge sources that are producing comparatively fit individuals. We also implement safe-guards against the starvation of any knowledge source, so that all sources have the opportunity to lead the search if they are able. The prevention of starvation is discussed in section 3.4.1

Each knowledge source KS_i has an associated probability P_i of being chosen to influence a given individual. We will define P to form a probability distribution over the knowledge sources. To encourage the exploitation of knowledge sources which are producing high-performing individuals, we derive its probability from the Pareto ranks of the individuals in the current population that they generated. To create the probability distribution, we first assign each knowledge source a score: the average of the inverses of the Pareto ranks of the individuals created by the knowledge source. Let IND_i be the set of individuals in a population created by knowledge source KS_i, and let PR(x) be the Pareto rank of individual x, where non-dominated solutions have Pareto rank 1. Then we define the score of a knowledge source in equation (2).

$$\mathrm{Score}(KS_i) = \frac{\sum_j \frac{1}{PR(IND_{i,j})}}{|IND_i|} \tag{2}$$

For example, we see a population in Figure 10 with individuals from the Δ, d, and W knowledge sources. The individuals form three Pareto ranks. The scores for each knowledge source are as follows:

$$\mathrm{Score}(KS_{\Delta}) = \frac{5/2}{5} = 5/6$$

$$\mathrm{Score}(KS_{\mathrm{O}}) = \frac{5/3}{5} = 5/9$$

$$\mathrm{Score}(KS_{\square}) = \frac{5/6}{5} = 5/12$$

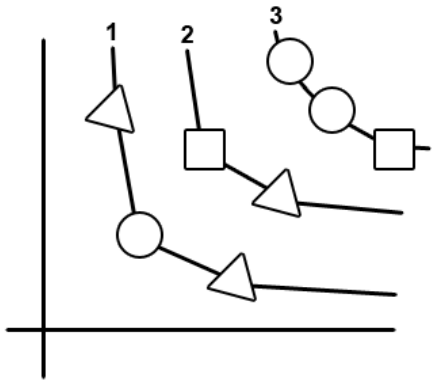

Figure 10: Example Pareto Fronts

To compute the final probabilities for each knowledge source, we divide the score of each knowledge source by the sum of scores for all knowledge sources. We also call this probability of the knowledge source being chosen to influence an individual the weight of the knowledge source.

$$P_i = \frac{\text{Score}(KS_i)}{\sum_j \text{Score}(KS_j)} \tag{3}$$

For our example from Figure 10, we compute the following probabilities:

$$W(KS_{\Delta}) = 6/13 \approx 0.46$$

$$W(KS_{\text{O}}) = 4/13 \approx 0.31$$

$$W(KS_{\square}) = 3/13 \approx 0.23$$

Since Δ is currently outperforming the other knowledge sources, we reward it by increasing the probability that it is chosen to influence individuals.

3.3 Acceptance

The first step in the Cultural Algorithm cycle is to accept a number of individuals to update the belief space. The Belief Space is intended to represent information which can be used to produce high-performing individuals. Therefore, we adopt a very simple acceptance function: 25% of the individuals with the best Pareto ranks are accepted. This

is an elitist acceptance process, and with it, we make no attempt to promote diversity to avoid settling on local optima. Instead of enforcing diversity in the Belief Space, we do so in the population by the design of our selection function. To enforce diversity, we must permit known suboptimal solutions. Since the Belief Space is used to influence the entire population, fitness concessions made to tolerate diverse solutions would feed back into the population, slowing the search for true optimal solutions.

3.4 Selection

After elite members of the population update the belief space, each individual is influenced by one knowledge source and a child is produced. For a population of size N, we have the N individuals from the original population and N newly created individuals from which to select those which will form the next generation. In addition to the standard goal of allowing fit solutions to survive, we also use selection to promote diversity. So as to adhere to the principle of the survival of the fittest, we generate most of our new population from the highest performing individuals. In order to encourage diversity and escape local optima, a small fraction of individuals from each knowledge source regardless of their Pareto rank is selected. Specifically, the N original individuals and N new individuals are first combined into a single collection, and their Goldberg Pareto ranks are computed. Each knowledge soure has $N/20$ of its solutions selected into the new population regardless of their Pareto ranks. This leaves $3N/4$ spots in the new population, which are filled by individuals with the lowest Pareto ranks.

3.4.1 Diversity through selection

In general, the approach to selection described above incurs a brief search of areas away from the current non-dominated solutions which may contain optima. These

dominated solutions can lead to promising regions which would otherwise be lost. If the search nearby is fruitless, the dominated solution and those found nearby will be discarded with high probability after failing to be selected in future generations. The inclusion of meritless individuals also helps alleviate a specific common problem. The problem occurs when the domain knowledge source performs better than the others but only finds a local optimum. After a number of generations of progressing toward the false Pareto front, it is likely that the domain knowledge source will have a high weight and control many individuals. The other more exploratory and less aggressive knowledge sources will be starved for weight and control of individuals. This problem is exacerbated by the social fabric feature which encourages conformity, requiring individuals to accept the influence of a knowledge source if many of its neighbors do. An individual, then, will likely be influenced by the domain knowledge source, even if another is initially chosen for it. By forcing $N/20$ individuals from each knowledge source into the population, we can prevent their starvation, and allocate solutions to the search away from a false Pareto front. Also note that during the influence procedure, we must ensure that at least $N/20$ solutions are chosen for each knowledge source. This ensures that we will have enough individuals from which to select the next population.

3.5 An example run

To illustrate the mechanics of MOCA, we examine the progress of the algorithm in detail as it optimizes a test problem. In chapter 4, we describe the test problems we use to evaluate the performance of MOCA. We use the first of these problems, DTLZ1, for our example run.

3.5.1 DLTZ1

Minimize

$f_1(x) = \frac{1}{2} x_1 x_2 \cdots x_{M-1}(1 + g(x_M))$,

$f_2(x) = \frac{1}{2} x_1 x_2 \cdots (1 - x_{M-1})(1 + g(x_M))$,

$\vdots$

$f_{M-1}(x) = \frac{1}{2} x_1 (1 - x_2)(1 + g(x_M))$,

$f_M(x) = \frac{1}{2}(1 - x_1)(1 + g(x_M))$,

subject to $0 \leq x_i \leq 1$, for $i = 1,2, \ldots, n$.

where $g(x_M) = 100(|x_M| + \sum_{x_i \in x_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)))$

The example run uses a population size of 100 and runs for 200 generations. We take $M = 3$, and $k = |x_M| = 5$. The total number of variables is 7.

The Pareto set lies on $x_i = 0.5$ for $x_i \in x_M$ and the Pareto front lies on $\sum_{i=1}^{M} f_i = 0.5$. The has $g(x_M)$ function produces $11^k - 1$ false Pareto fronts where $x_i \in \{0.1, 0.2, \ldots, 1\}\ \forall x_i \in x_M$.

Figure 11 plots the minimum and median values of $\sum_{m=1}^{M} f_m(x)$ for $x \in \text{population}$. We show the same plot in Figure 12 at a smaller scale so the behavior at the end of the optimization can be seen clearly. The knowledge source which produced the minimum solution is also indicated. Indicators are omitted when a knowledge source provides the minimum solution over consecutive generations. Figure 14 displays the number of individuals in the population generated by each knowledge source. Figure 13 shows the weights associated with each knowledge as described in section 3.2

In Figure 11 we see that the domain knowledge source immediately makes drastic improvement on the poorly performing initial population.

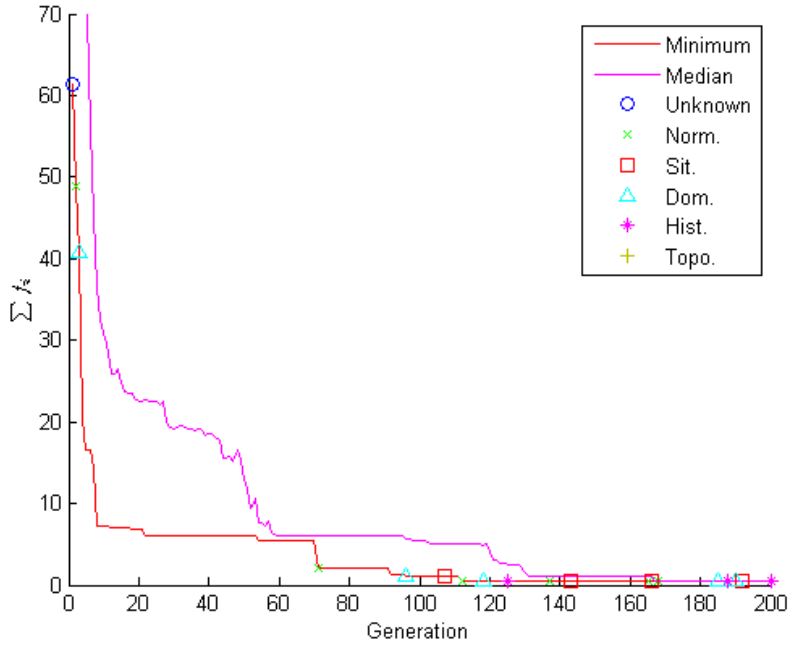

Figure 11: The minimum and median values for $\sum_{m=1}^{M} f_m$

At generation 8, the search begins approaching the locally Pareto-optimal front at $x_M = (0.6, 0.6, 0.3, 0.7, 0.6)$ with $\sum f_i = 6$. The minimum solution at generation 8 has $x_M = (0.6007, 0.6005, 0.3033, 0.7005, 0.5993)$ and $\sum f_i = 7.204$. By generation 24, the domain knowledge source has found a solution very near the local optimum, and makes little progress thereafter. A better local optima isn't found until generation 51. During the generations between, the situational knowledge source begins to place individuals on the local front. In Figure 13, we see the resultant increase in weight for situational knowledge which peaks in generation 46.

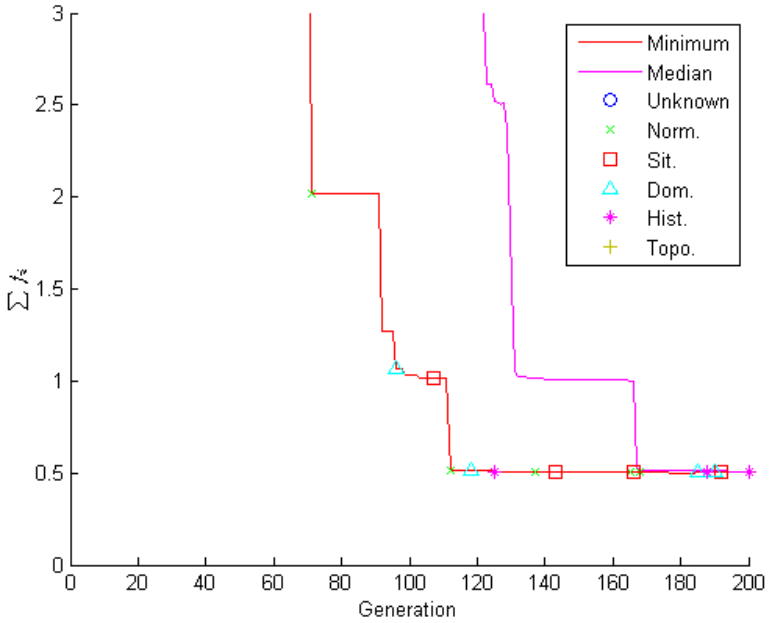

Figure 12: The minimum and median values for $\sum_{m=1}^{M} f_m$

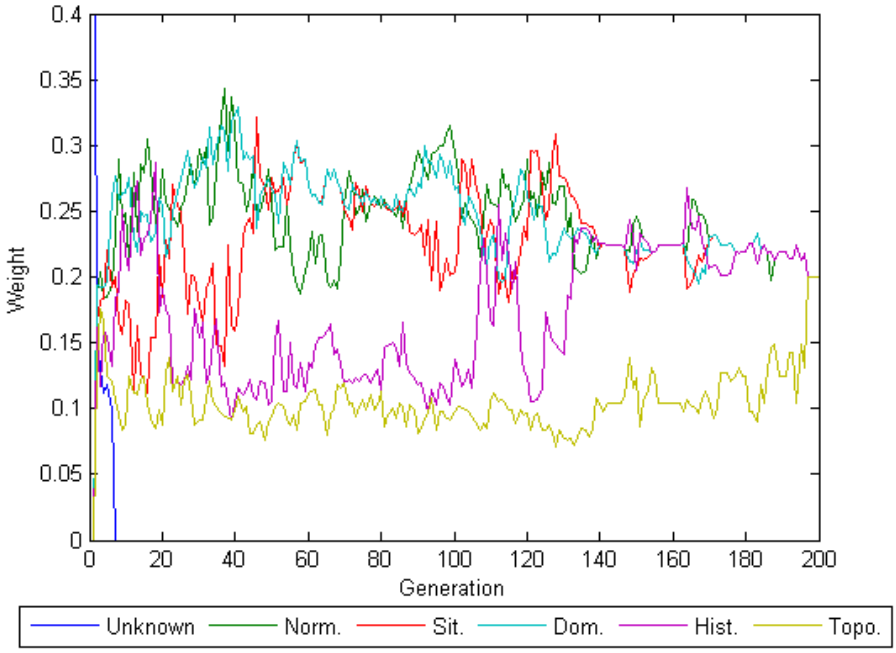

Figure 13: Knowledge source weights

This high weight allows situational knowledge to increase its presence in the population until it matches that of the domain in generation 69, as shown in Figure 14.

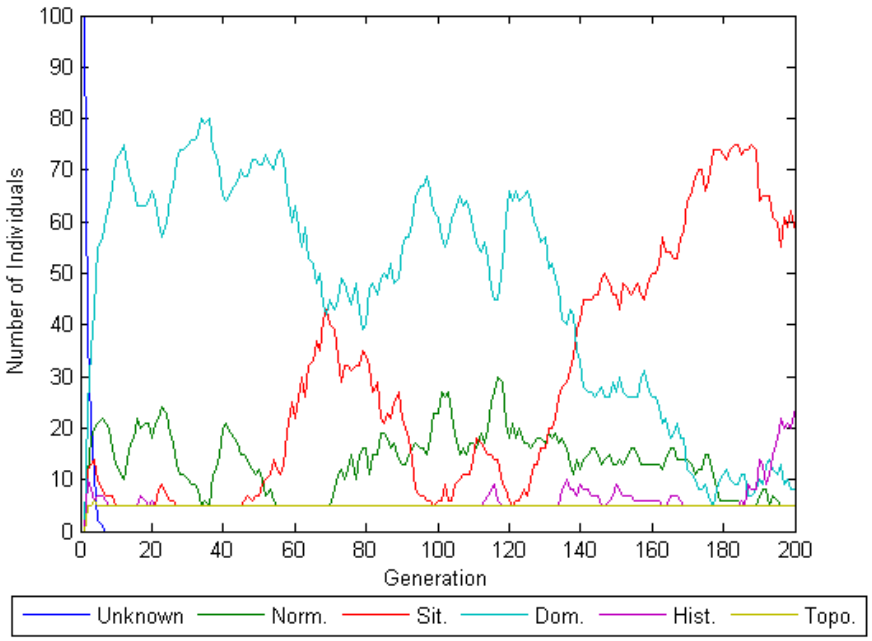

Figure 14: Knowledge source presence

In generation 71 the normative knowledge source finds the local Pareto front at $x_M = (0.6, 0.5, 0.4, 0.4, 0.5)$. The individuals generated by situational knowledge then begin to be dominated and expelled from the population. The normative knowledge finds one more false Pareto front in generation 92, which is improved by domain knowledge in generation 96, before approaching the true Pareto front in generation 112. After domain knowledge refines the solutions, situational knowledge begins distributing solutions across the front once again.

The final population in objective space is shown in Figure 15. We can see that the distribution of solutions is not uniform, and a significant portion of the true Pareto front is not represented in the final population. No solutions from the topographical knowledge source are visible because they do not lie on the Pareto front. In fact, throughout the optimization, topographical knowledge produced no solutions with Pareto rank 1, and never had more than the five guaranteed individuals accepted into the population.

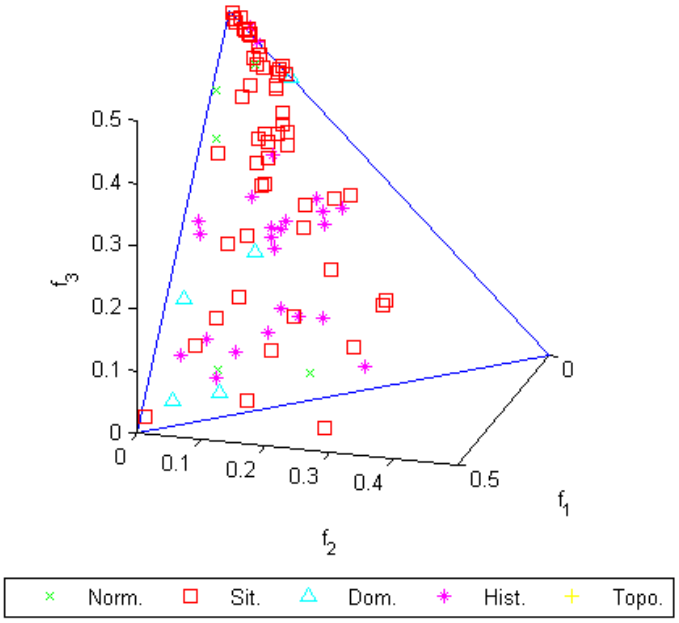

Figure 15: The final population in objective space

Figure 16 plots x_1 against x_2 for the final population. We omit the remaining five decision variables because, for solutions on the true Pareto front, they are all approximately 0.5. Again, the population is not evenly distributed, and leaves two corners of the search space bare.

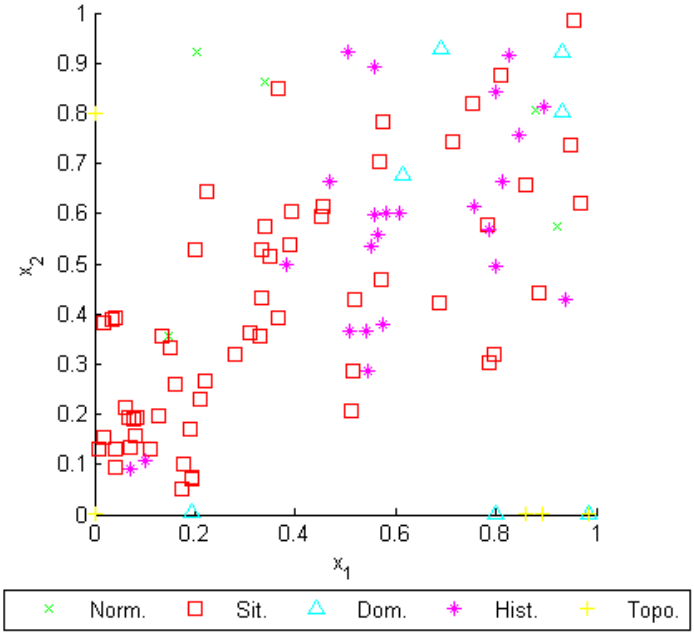

Figure 16: The final population in decision space

CHAPTER 4

EXPERIMENTAL FRAMEWORK

We test the performance of MOCA on the DTLZ problems originally presented in *Scalable Multi-Objective Optimization Test Problems* in 2002 by Deb, Thiele, Laumanns, and Zitzler. The problems were designed to scale to any number of objective functions or decision variables [9]. We use the definitions of the problems given in *Evolutionary algorithms for solving multi-objective problems* [10] and compare the results of MOCA to NSGA-II. We also follow the recommendations for the number of decision variables and objectives for each problem. For all of the problems, MOCA uses a population size of 100 and runs for 300 generations. The social fabric component introduced by used in CA was also used in MOCA. Here we used LBEST. The topographical knowledge source is not included in MOCA for any of the experiments. Statistics for NSGA-II were produced using the implementation provided by jMetal [11]. NSGA-II is run with a population size of 100 and is limited to 100,000 fitness evaluations. It uses a crossover probability of 0.9 and a mutation probability $1/n$ for n decision variables. We also use the jMetal implementation to compute the performance measures for MOCA. Below we describe the individual problems used and the parameters of the algorithm.

4.1 DTLZ1

Minimize

$$f_1(x) = \frac{1}{2}x_1x_2\cdots x_{M-1}(1+g(x_M)),$$
$$f_2(x) = \frac{1}{2}x_1x_2\cdots(1-x_{M-1})(1+g(x_M)),$$
$$\vdots$$
$$f_{M-1}(x) = \frac{1}{2}x_1(1-x_2)(1+g(x_M)),$$
$$f_M(x) = \frac{1}{2}(1-x_1)(1+g(x_M)),$$

subject to $0 \le x_i \le 1,$ for $i = 1,2,\ldots,n.$

where $g(x_M) = 100(|x_M| + \sum_{x_i \in x_M}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)))$

The Pareto set lies on $x_i = 0.5$ for $x_i \in x_M$ and the Pareto front lies on the hyperplane $\sum_{i=1}^{M} f_i = 0.5$. The sinusoidal $g(x_M)$ function produces $11^k - 1$ false Pareto fronts where $x_i \in \{0.1, 0.2, \dots, 1\}\ \forall x_i \in x_M$. For our tests we use $M = 3$, and $k = |x_M| = 5$. The total number of variables is 7.

4.2 DTLZ2

Minimize
$$f_1(x) = (1 + g(x_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2),$$
$$f_2(x) = (1 + g(x_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2),$$
$$f_3(x) = (1 + g(x_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\sin(x_{M-2}\pi/2),$$
$$\vdots$$
$$f_{M-1}(x) = (1 + g(x_M))\cos(x_1\pi/2)\sin(x_2\pi/2),$$
$$f_M(x) = (1 + g(x_M))\sin(x_1\pi/2),$$
subject to $0 \le x_i \le 1,$ for $i = 1,2, \dots, n.$
where $g(x_M) = \sum_{x_i \in x_M} (x_i - 0.5)^2$

The Pareto set lies on $x_i = 0.5$ for $x_i \in x_M$ and the Pareto front lies on the unit hypersphere $\sum_{i=1}^{M} f_i^2 = 1$. DTLZ2 uses a simple parabolic $g(x_M)$ function which does not incur any false Pareto fronts. For our tests we use $M = 3$ and $k = |x_M| = 10$.

4.3 DTLZ3

Minimize
$$f_1(x) = (1 + g(x_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2),$$
$$f_2(x) = (1 + g(x_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2),$$
$$f_3(x) = (1 + g(x_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\sin(x_{M-2}\pi/2),$$
$$\vdots$$
$$f_{M-1}(x) = (1 + g(x_M))\cos(x_1\pi/2)\sin(x_2\pi/2),$$
$$f_M(x) = (1 + g(x_M))\sin(x_1\pi/2),$$
subject to $0 \le x_i \le 1,$ for $i = 1,2, \dots, n.$
where $g(x_M) = 100(|x_M| + \sum_{x_i \in x_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)))$

DTLZ3 is identical to DTLZ2 but uses the $g(x_M)$ function from DTLZ1, producing the same $11^k - 1$ false Pareto fronts as DTLZ1. The Pareto set lies on $x_i = 0.5$ for $x_i \in x_M$ and the Pareto front lies on $\sum_{i=1}^{M} f_i^2 = 1$. For our tests we use $M = 3$ and

$k = |x_M| = 10.$

4.4 DTLZ4

Minimize
$f_1(x) = (1 + g(x_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2) \cdots \cos(x_{M-2}^\alpha \pi/2)\cos(x_{M-1}^\alpha \pi/2),$
$f_2(x) = (1 + g(x_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2) \cdots \cos(x_{M-2}^\alpha \pi/2)\sin(x_{M-1}^\alpha \pi/2),$
$f_3(x) = (1 + g(x_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2) \cdots \sin(x_{M-2}^\alpha \pi/2),$
$\vdots$
$f_{M-1}(x) = (1 + g(x_M))\cos(x_1^\alpha \pi/2)\sin(x_2^\alpha \pi/2),$
$f_M(x) = (1 + g(x_M))\sin(x_1^\alpha \pi/2),$
subject to $0 \le x_i \le 1,$ for $i = 1,2, \dots, n.$
where $g(x_M) = \sum_{x_i \in x_M} (x_i - 0.5)^2$

DTLZ4 modifies DTLZ2 by replacing x_i with x_i^α. This replacement concentrates solutions near the $f_1 - f_M$ plane. The Pareto set lies on $x_i = 0.5$ for $x_i \in x_M$ and the Pareto front lies on $\sum_{i=1}^{M} f_i^2 = 1$. For our tests we use $M = 3,$ $k = |x_M|$, and $\alpha = 100.$

4.5 DTLZ5

Minimize
$f_1(x) = (1 + g(x_M))\cos(\theta_1 \pi/2)\cos(\theta_2 \pi/2) \cdots \cos(\theta_{M-2} \pi/2)\cos(\theta_{M-1} \pi/2),$
$f_2(x) = (1 + g(x_M))\cos(\theta_1 \pi/2)\cos(\theta_2 \pi/2) \cdots \cos(\theta_{M-2} \pi/2)\sin(\theta_{M-1} \pi/2),$
$f_3(x) = (1 + g(x_M))\cos(\theta_1 \pi/2)\cos(\theta_2 \pi/2) \cdots \sin(\theta_{M-2} \pi/2),$
$\vdots$
$f_{M-1}(x) = (1 + g(x_M))\cos(\theta_1 \pi/2)\sin(\theta_2 \pi/2),$
$f_M(x) = (1 + g(x_M))\sin(\theta_1 \pi/2),$
subject to $0 \le x_i \le 1,$ for $i = 1,2, \dots, n.$
where

$$\theta_i = \frac{\pi}{4\big(1 + g(x_M)\big)}(1 + 2g(x_M)x_i), \qquad \text{for } i = 1,2, \dots, (M-1)$$

$$g(x_M) = \sum_{x_i \in x_M} (x_i - 0.5)^2$$

The Pareto set lies on $x_i = 0.5$ for $x_i \in x_M$ and the Pareto front lies on a curve on $\sum_{i=1}^{M} f_i^2 = 1$. For our tests we use $M = 3$ and $k = |x_M| = 10.$

4.6 DTLZ6

Minimize
$f_1(x) = (1 + g(x_M))\cos(\theta_1 \pi/2)\cos(\theta_2 \pi/2) \cdots \cos(\theta_{M-2} \pi/2)\cos(\theta_{M-1} \pi/2),$

$$f_2(x) = (1 + g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2),$$
$$f_3(x) = (1 + g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\sin(\theta_{M-2}\pi/2),$$
$$\vdots$$
$$f_{M-1}(x) = (1 + g(x_M))\cos(\theta_1\pi/2)\sin(\theta_2\pi/2),$$
$$f_M(x) = (1 + g(x_M))\sin(\theta_1\pi/2),$$

subject to $0 \le x_i \le 1$, for $i = 1,2,\ldots,n$.

where

$$\theta_i = \frac{\pi}{4\left(1 + g(x_M)\right)}(1 + 2g(x_M)x_i), \qquad \text{for } i = 1,2,\ldots,(M-1)$$

$$g(x_M) = \sum_{x_i \in x_M} (x_i)^{0.1}$$

The Pareto set lies on $x_i = 0$ for $x_i \in x_M$ and the Pareto front is identical to that of DTLZ5. For our tests we use $M = 3$ and $k = |x_M| = 10$.

CHAPTER 5

RESULTS

5.1 Performance Measures

Table 1 shows the comparison of NSGA-II and MOCA for two performance metrics: hypervolume (HV) [12] and generational distance (GD) [13]. The HV value provides a measure of how much of the true Pareto front is covered. Since we use three objective functions for each problem, the hypervolume is the volume of the union of cuboids with corners $(0,0,0)$ and $(f_1(x_i), f_2(x_i), f_3(x_i))\ \forall x_i$ in the population. Before computing the volume, the decision variables are scaled so that, for each dimension, the minimum and maximum values of a reference Pareto front are 0 and 1 respectively. Finally, each decision variable of each solution is inverted, that is, subtracted from 1, to obtain the set of solutions for which the hypervolume will be calculated. The HV of the reference Pareto front for DTLZ1 is 0.8257. The reference Pareto front for DTLZ2 through DTLZ4 has HV 0.4678. For DTLZ5 and DTLZ6, the reference Pareto front has HV 0.0956.

The generational distance measures how close a population is to the true Pareto front. It is implemented as $\sqrt{\sum_{x_i \in Pop} D(x_i, PF_{TRUE})^2}\ /n$ where $D(x, PF)$ is the distance from x to the nearest point in a reference Pareto front PF, and n is the number of solutions in the population. Values for GD are in $[0, \infty)$.

5.2 Algorithm Comparison

For each of the problems, MOCA provides a competitive mean HV but is outperformed by NSGA-II. The standard deviation of HV for MOCA is an average of a factor of ten higher than that of NSGA-II. MOCA performs best on DTLZ1 and DTLZ5, giving an average HV within a factor of 0.9 of NSGA-II. For DTLZ4, MOCA is unable to

distribute solutions over the face of the spherical Pareto front, placing solutions only on the f_1-f_2 or f_1-f_3 planes. The mechanisms in the knowledge sources designed to provide a good distribution are functioning correctly but need improvement to match the leading MOEAs.

MOCA gives very competitive GD metric values for most of the problems tested. It performs worst on DTLZ2, with a mean GD an order of magnitude greater than NSGA-II. This is because the parabolic fitness functions for DTLZ2 are much less steep near the Pareto front than other problems. In MOCA, the domain knowledge source is the primary means of placing solutions at precisely optimal values, and works better with steep functions.

Table 1: DTLZ Performance Measures

MOP	MOEA	HV Mean and SD	HV Median and IQR	GD Mean and SD	GD Median and IQR
DTLZ1	NSGA-II	$7.61e-01_{7.9e-03}$	$7.62e-01_{7.1e-03}$	$3.13e-02_{1.8e-01}$	$6.68e-04_{2.3e-04}$
	MOCA	$7.15e-01_{2.4e-02}$	$7.18e-01_{2.6e-02}$	$1.25e-03_{3.3e-04}$	$1.21e-03_{4.0e-04}$
DTLZ2	NSGA-II	$3.75e-01_{5.3e-03}$	$3.75e-01_{7.1e-03}$	$1.30e-03_{1.6e-04}$	$1.31e-03_{2.2e-04}$
	MOCA	$2.60e-01_{6.4e-02}$	$2.33e-01_{1.2e-01}$	$1.27e-02_{1.4e-02}$	$1.66e-03_{2.4e-02}$
DTLZ3	NSGA-II	$3.74e-01_{8.6e-03}$	$3.75e-01_{1.1e-02}$	$1.30e-03_{2.9e-04}$	$1.22e-03_{2.2e-04}$
	MOCA	$2.53e-01_{1.0e-01}$	$2.81e-01_{1.4e-01}$	$4.37e-03_{1.1e-02}$	$1.93e-03_{5.6e-04}$
DTLZ4	NSGA-II	$3.76e-01_{4.6e-03}$	$3.77e-01_{7.1e-03}$	$5.17e-03_{2.6e-04}$	$5.19e-03_{3.4e-04}$
	MOCA	$1.35e-01_{9.3e-02}$	$1.97e-01_{2.0e-01}$	$3.40e-03_{1.1e-02}$	$2.12e-03_{2.4e-03}$
DTLZ5	NSGA-II	$9.29e-02_{2.0e-04}$	$9.29e-02_{2.6e-04}$	$3.76e-04_{7.3e-05}$	$3.59e-04_{8.0e-05}$
	MOCA	$9.05e-02_{1.5e-03}$	$9.10e-02_{1.6e-03}$	$5.02e-04_{9.5e-04}$	$2.64e-04_{5.8e-05}$
DTLZ6	NSGA-II	$8.35e-02_{1.2e-02}$	$9.36e-02_{1.8e-02}$	$2.30e-03_{2.8e-03}$	$6.69e-04_{2.4e-03}$
	MOCA	$3.40e-02_{4.0e-02}$	$1.48e-11_{7.9e-02}$	$6.70e-04_{1.7e-03}$	$8.02e-05_{5.3e-04}$

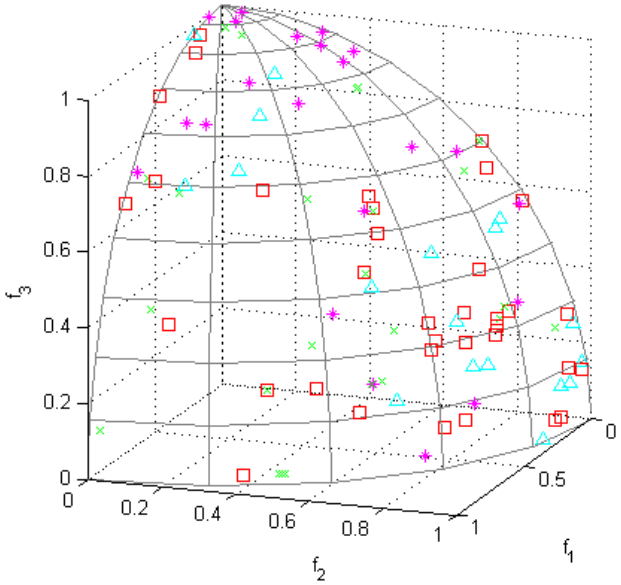

Figure 17: MOCA final population in objective space for DTLZ2

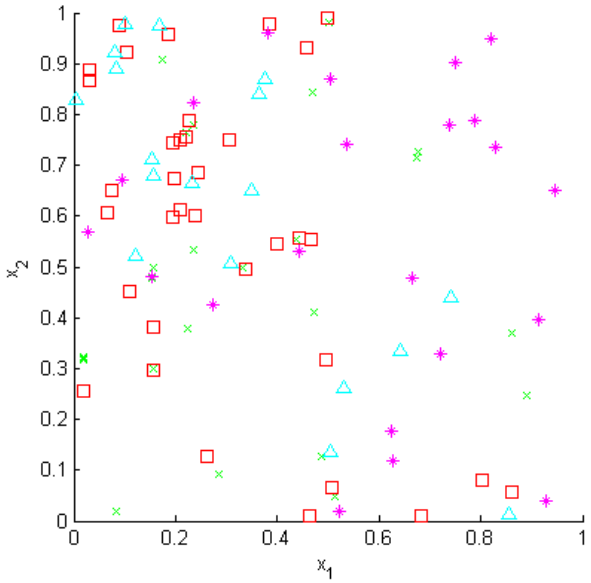

Figure 18: MOCA final population in decision space for DTLZ2

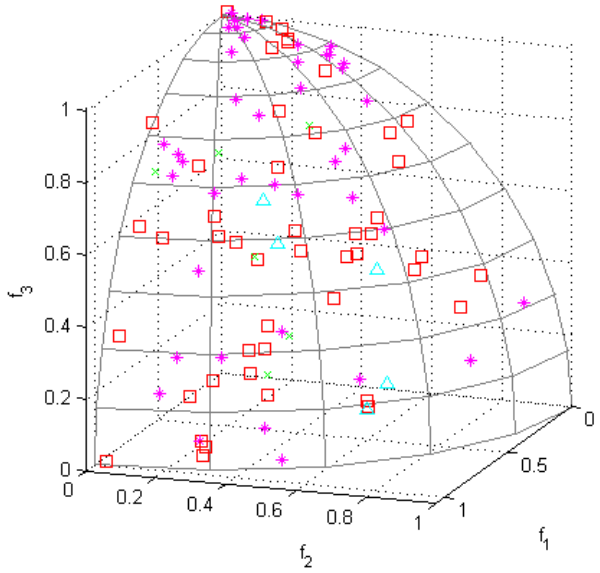

Figure 19: MOCA final population in objective space for DTLZ3

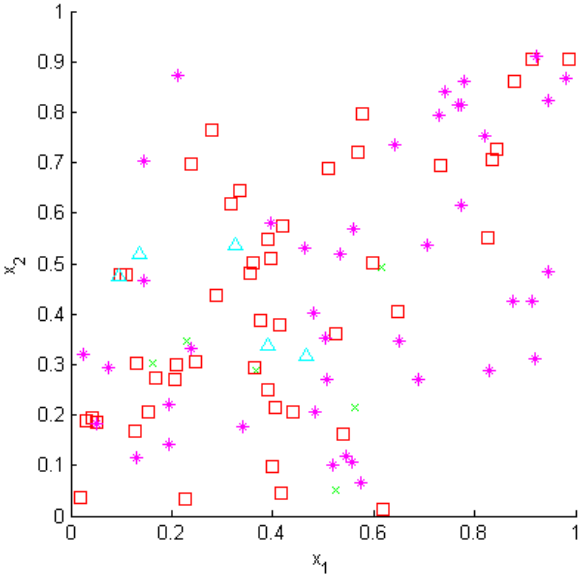

Figure 20: MOCA final population in decision space for DTLZ3

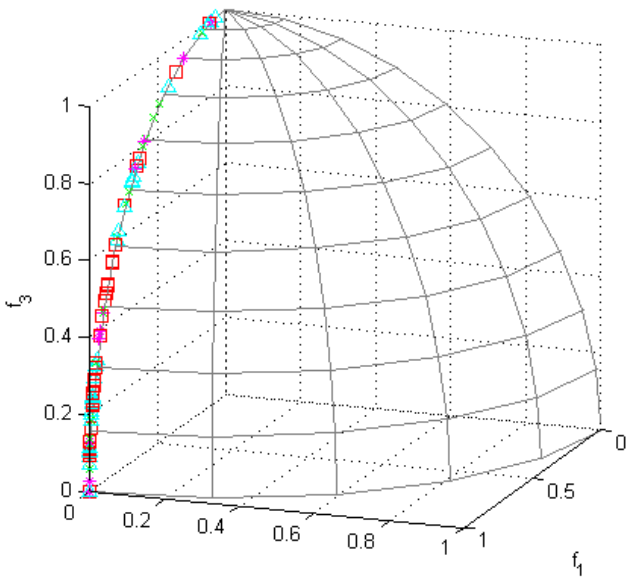

Figure 21: MOCA final population in objective space for DTLZ4

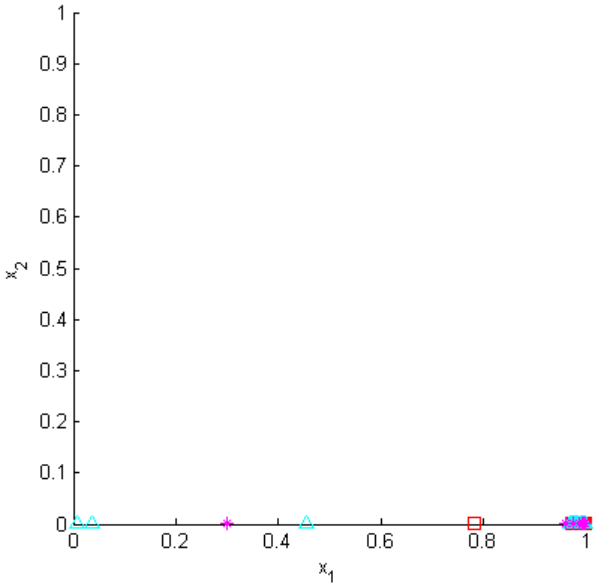

Figure 22: MOCA final population in decision space for DTLZ4

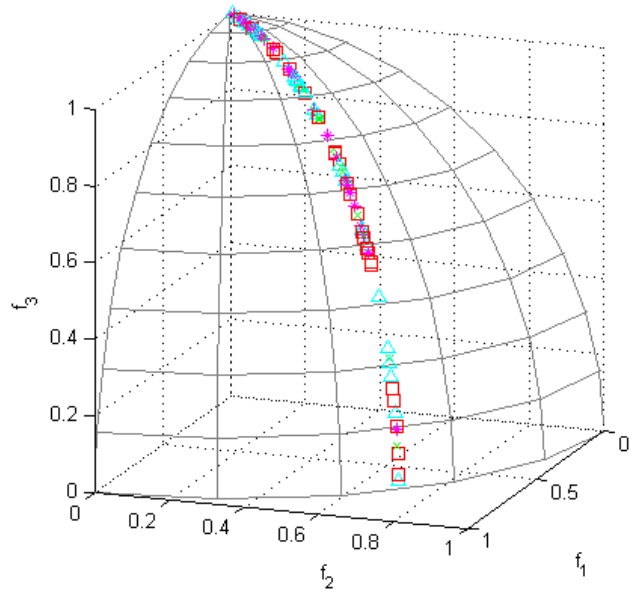

Figure 23: MOCA final population in objective space for DTLZ5

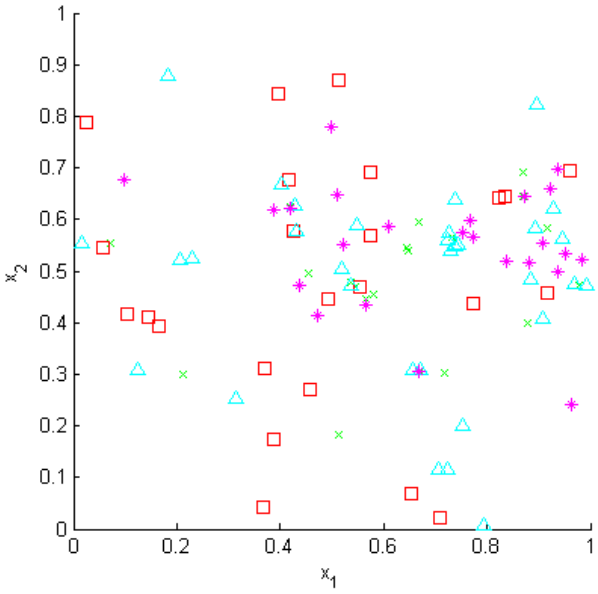

Figure 24: MOCA final population in decision space for DTLZ5

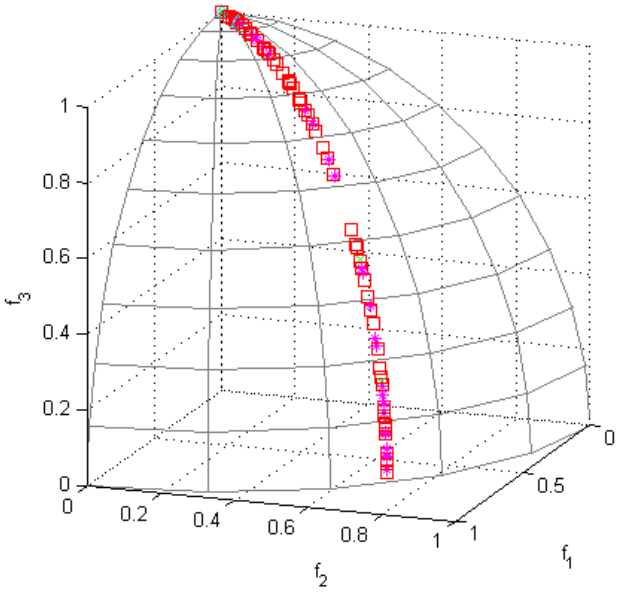

Figure 25: MOCA final population in objective space for DTLZ6

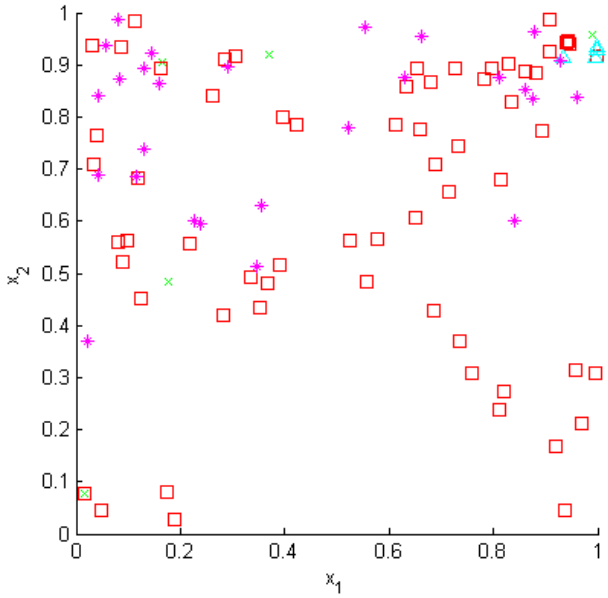

Figure 26: MOCA final population in decision space for DTLZ6

CHAPTER 6

CONCLUSION

MOCA is a functional model of the multi-objective problem-solving mechanisms which occur naturally in human culture. We found that complex cultural processes can be approximated using simple rules and used to solve multi-objective optimization problems.

CA is a proven approach for single-objective optimization and provides the basis for our work. MOCA shows great potential by successfully optimizing problems as a standalone system. However, like CA, it can also be used to augment existing MOEAs. Any generational optimization algorithm can be run in the Population Space and accept influence from the knowledge sources. Furthermore, the mechanisms used in MOCA are simple enough to be integrated individually into other methods. Any one knowledge source can be implemented and used to guide the search of an MOEA. MOCA is accessible due to its conformance to the structure of the widely known CA. It uses the same communication protocol between the Population Space and Belief Space, and implements the same five knowledge sources. This makes the system transparent to those who have studied CA.

The topographical knowledge source is unproductive when considering only one of the objective functions in the problem at a time. We suggest modifying the topographical influence function to divide the search space based on the Pareto rank of the individuals in the population. This can be thought of as maintaining the single-objective approach of topographical knowledge, and using the Pareto rank of a solution as the objective function. We suggest investigating the use of a Pareto rank other than the Goldberg method for this purpose. We also recommend implementing a real

gradient approximation method for the domain knowledge source. We were able to avoid an exponential number of fitness evaluations by using a simple and effective heuristic. However, our heuristic has no mathematical basis and can certainly be improved. Finally, we recommend that the mechanisms for spreading the solutions evenly over the Pareto front be improved. The situational and historical knowledge sources can achieve an acceptable spread on simple problems but not on problems with very uneven density. We expect that a better implementation of the topographical knowledge source can be used to fill this role.

BIBLIOGRAPHY

1 Van Veldhuizen, David A. and Lamont, Gary B. *Multiobjective Evolutionary Algorithm Research: A History and Analysis*. Air Force Institute of Technology, 1998.

2 Reynolds, Robert G. *An adaptive computer model of the evolution of agriculture for hunter-gatherers in the valley of Oaxaca, Mexico*. Univ. of Michigan, Ann Arbor, 1979.

3 Reynolds, Robert G. and Nazzal, Ayman. Using Cultural Algorithms with Evolutionary Computing to Extract Site Location Decisions from Spatio-Temporal Databases. In *Proceedings of the 6th International Conference on Evolutionary Programming VI* (London 1997), Springer-Verlag, 443-456.

4 Coello Coello, Carlos A. and Becerra, Ricardo Landa. Evolutionary Multiobjective Optimization using a Cultural Algorithm. *IEEE Swarm Intelligence Symposium* (2003), 6-13.

5 Jin, X. and Reynolds, R.G. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach. In *Proceeding of the 1999 Congress on Evolutionary Computation* (1999), 1672-1678.

6 Reynolds, R.G. and Saleem, S.M. The impact of environmental dynamics on cultural emergence. In *Perspectives on Adaptions in Natural and Artificial Systems*. Oxford University Press, 2001.

7 Reynolds, R.G. and Peng, B. Knowledge learning and social swarms in cultural algorithms. *Journal of Mathematical Sociology*, 29 (2005), 1-18.

8 Goldberg, David E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, 1989.

9 Deb, Kalyanmoy, Thiele, Lothar, Laumanns, Marco, and Zitzler, Eckart. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation* (2002).

10 Coello Coello, Carlos A., Lamont, Gary B., and Van Veldhuizen, David A. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer-Verlag Inc., 2006.

11 Durillo, Juan J, Nebro, Antonio J, Luna, Francisco, Dorronsoro, Bernabe, and Alba, Enrique. *jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics*. University of Malaga, 2006.

12 Zitzler, E. and Thiele, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. In *IEEE Transactions on Evolutionary Computation* (1999), 257-271.

13 Valenzuela-Rendon, Manuel and Uresti-Charre, Eduardo. A Non-Generational Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the Seventh International Conference on Genetic Algorithms* (San Fransico 1997), Morgan Kaufmann, 657-665.

ABSTRACT

MULTI-OBJECTIVE CULTURAL ALGORITHMS

by

CHRISTOPHER BEST

August 2009

Advisor: Dr. Robert Reynolds

Major: Computer Science

Degree: Master of Science

Multi-objective optimization is a widely applicable problem in Engineering and Computer Science. In the past, Cultural Algorithms have been used to solve complex optimization and design problems. In this thesis we extend the Cultural Algorithm Framework to support multi-objective problems. The resultant system, Multi-Objective Cultural Algorithms (MOCA), can be used independently or as a supplement to existing MO optimization methods. We compare the performance of our algorithm with NSGA-II using problems from the DTLZ test suite, a popular MOEA test suite. We found that Cultural Algorithms are a promising technique for solving multi-objective problems.

AUTOBIOGRAPHICAL STATEMENT

Chris Best graduated from the University of Michigan with a B.S.E. in Computer Science in 2007 where he worked for the UM COE IOE C4E as a programmer for several years. He has been studying in Dr. Robert Reynolds AI lab for two years. Chris enjoys puzzles, guitar, mathematics, piano, soccer, and recently joined an ice hockey team.