July, 2014

Feedback Control for Multi-modal Optimization using Genetic Algorithms

Jun Shi, *Carnegie Mellon University*
Ole J Mengshoel, *Carnegie Mellon University*
Dipan K Pal, *Carnegie Mellon University*

Feedback Control for Multi-modal Optimization using Genetic Algorithms

Jun Shi
Carnegie Mellon University
Pittsburgh, PA
jshi1@andrew.cmu.edu

Ole J. Mengshoel
Carnegie Mellon University
Moffett Field, CA
ole.mengshoel@sv.cmu.edu

Dipan K. Pal
Carnegie Mellon University
Pittsburgh, PA
dipanp@andrew.cmu.edu

ABSTRACT

Many optimization problems are multi-modal. In certain cases, we are interested in finding multiple locally optimal solutions rather than just a single optimum as is computed by traditional genetic algorithms (GAs). Several niching techniques have been developed that seek to find multiple such local optima. These techniques, which include sharing and crowding, are clearly powerful and useful. But they do not explicitly let the user control the number of local optima being computed, which we believe to be an important capability.

In this paper, we develop a method that provides, as an input parameter to niching, the desired number of local optima. Our method integrates techniques from feedback control, includes a sensor based on clustering, and utilizes a scaling parameter in Generalized Crowding to control the number of niches being explored. The resulting Feedback Control GA (FCGA) is tested in several experiments and found to perform well compared to previous approaches. Overall, the integration of feedback control and Generalized Crowding is shown to effectively guide the search for multiple local optima in a more controlled fashion. We believe this novel capability has the potential to impact future applications as well as other evolutionary algorithms.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Markov Processes, Probabilistic algorithms including Monte-Carlo; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control methods, Search methods—*Heuristic methods*

Keywords

FCGA; Genetic Algorithms; Generalized Crowding; Feedback Control; Multi-modal Optimization

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2662-9/14/07$15.00.
http://dx.doi.org/10.1145/2576768.2598231.

1. INTRODUCTION

Multi-modal problems have multiple local optima in their solution landscape. In many cases it is desirable to obtain multiple distinct and locally optimal solutions to such problems. One reason for such a requirement is that some of the local optima may be surrounded by a steep fitness landscape, while other local optima may not be. The latter case might be preferred if a more stable solution is needed. Obtaining multiple solutions is also useful in situations where there is considerable uncertainty about the fitness function or important problem aspects are not fully captured in it.

Different niching techniques including sharing and crowding have been developed to converge to multiple solutions [1, 2, 3, 4, 5]. Based on Deterministic [1] and Probabilistic Crowding [6], a new Generalized Crowding method which integrates these two original ideas has been developed [5]. A so-called *scaling factor* is the key parameter in Generalized Crowing which allows us to simulate both Deterministic and Probabilistic Crowding. The existence of a scaling factor of course raises the question of designing schedules for varying it. Recently, several schedules have been investigated including a constant schedule; an exponentially-decaying schedule; a diversity-adaptive schedule based on population entropy; and a self-adaptive schedule [5, 7]. Although these approaches work well, there is no explicit control of the number of local optima, multiple distinct solutions, or niches explored or found.

More broadly, investigating the trade-off between exploration and exploitation is one of the central themes of evolutionary algorithms (EAs). In particular, exploration is emphasized in niching techniques, where the goal is to improve diversity or increase the number of local optima explored. Surprisingly, to the best of our knowledge, the literature has not discussed how to directly control the number of local optima found by niching algorithms. In this paper, we investigate this problem.

Specifically, we use the desired number of distinct niches (number of local optima) to be explored as the set-point in a feedback control loop. This enables the user to directly control, using the set-point, the number of niches explored in a Feedback Computing GA (FCGA) run. A set-point represents external constraints associated with a user or a computational process using an FCGA. In other words, the set-point may not have anything to do with the number of local optima in the fitness function, as long as we make the reasonable assumption that the set-point is less than or equal to the number of local optima.

Before summarizing our approach, we highlight a few situations in which better user control of the number of local optima, through a set-point, promises to be powerful:

- It might be useful in a design process to create diverse physical prototypes of candidate solutions. Clearly, there might exist time and budget constraints for developing such prototypes (e.g. 3D printing). In this case, the FCGA set-point would be the number of distinct physical prototypes to be created.
- When software is evolved, one might require that the candidate solutions be tested extensively. While running computer programs might seem relatively inexpensive, there are always time and budget constraints given the complexity often found in software. In this case, the FCGA set-point would be the number of distinct software prototypes to be evolved.
- A computer screen may visualize candidate solutions being evolved, for example for the purpose of interactive design of shapes [8] or interactive optimization of robot controllers [9]. In this case, the user's FCGA set-point would be the number of diverse shapes or robot controllers that can be meaningfully visualized.

In this paper, we consider applying feedback control on the scaling factor ϕ in Generalized Crowding to better achieve the desired number of locally optimal solutions as defined by the user. We apply simple proportional control, and analyze the impact of control parameters and also compare the resulting performance with existing scaling factor schedules.

The rest of this paper is organized as follows. In Section 2, related research on crowding, self-adaptation, and feedback computing is discussed. Section 3 presents our feedback control approach to EAs, the Feedback Control GA, and defines the components of the control loop in a GA context. In Section 4, we present experimental results for several test functions. Finally, we conclude with Section 5 and outline future research directions.

2. RELATED RESEARCH

2.1 Niching Techniques

Niching techniques in Evolutionary Algorithms (EAs) are often motivated by multi-modality in the objective function. Niching techniques include sharing [10] and crowding [11, 1, 6, 5]. Crowding, which we focus on in this paper, was introduced as a technique for preserving population diversity [11]. It is perhaps interesting to consider the mechanism of crowding using a biological perspective. Even when the whole population is derived from the same species, individuals mostly compete with each other within a niche. This results in the preservation of sub-populations in distinct niches, corresponding to local optima and their neighborhoods in the fitness function.

Technically, crowding techniques are applied in the survival stage of GAs. There are two phases in the crowding process: The Grouping Phase, used to pair up individuals according to a similarity metric, and the Replacement Phase, in which competitions are held within each pair to decide the winner [4]. In most crowding approaches the similarity metric used is genotypic distance.

The replacement rule plays an important role in crowding. Depending on how replacement is performed, several rules have been identified, including Deterministic Crowding [1] and Probabilistic Crowding [6]. Deterministic Crowding is an exploitative replacement rule since the winner of a competition is always the one with a higher fitness [1]. One shortcoming of Deterministic Crowding, however, is that it may lead to premature convergence. In Probabilistic Crowding, an individual wins a competition with a probability proportional to its fitness. This approach is more exploratory than Deterministic Crowding, but similar to Deterministic Crowding the selection pressure is not adaptive.

Subsequently, these two techniques were generalized by introduced a scaling factor ϕ, creating *Generalized Crowding* [5]. In Generalized Crowding the exploration-exploitation trade-off is parametrized through ϕ. This is similar, to some extent, to the use of temperature in simulated annealing [12, 13]. Intuitively, high temperature and high scaling factor give exploration, while low temperature and low scaling factor give exploitation. A key difference between simulated annealing and Generalized Crowding is that the latter is a population-oriented technique, while the former is not. Generalized Crowding and other EAs thus lend themselves more naturally to multi-modal optimization.

2.2 Adaptation and Self-adaptation

The idea of self-adaptation, including self-adaptive parameter control, has been extensively performed by EAs. Evolution strategies [14, 15] self-adapts the mutation step sizes throughout the search process. Mutation [16] as well as crossover rates [17] have been self-adapted. Generalized Crowding has been augmented with diversity-adaptive and self-adaptive methods [7]. In diversity-adaptive Generalized Crowding, the scaling factor ϕ is changed according to population diversity. In self-adaptive Generalized Crowding, ϕ is added to the chromosome, and undergoes crossover, mutation, and selection along with it.

2.3 Feedback Computing

Feedback control, both theory and practice, has traditionally been applied to physical systems governed by Newtonian mechanics. More recently, feedback control is starting to be applied to computing hardware and software; this is known as feedback computing [18]. Examples of feedback computing include: control of HTTP servers [19, 20], email servers [21], quality of service assurance [22], and Internet traffic control [23]. While computing systems are clearly different from traditional feedback control applications, many techniques have carried over.

3. FEEDBACK COMPUTING GA (FCGA)

Although the various crowding techniques and their corresponding replacement rules have been proven to work well with multi-modal optimization problems, one drawback is that they provide no direct user control over the number of niches (or local optima) being explored. We seek to attack this problem by adopting techniques from feedback control. Our novel idea of integrating feedback control with evolutionary computation is illustrated in Figure 1. Crucially, the user now has direct control over a set-point $r(t)$. This set-point represents how many niches, with local and diverse optima, a user wants to find using the GA.

We call this feedback computing pattern, where the computational process is a GA operating on a population $\pi(t)$ at time $t \geq 0$, a Feedback Control GA (FCGA). Similar

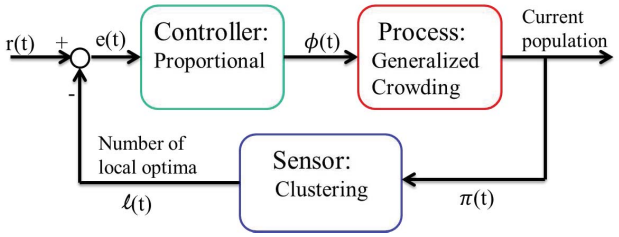


Figure 1: Feedback control loop of the Feedback Computing GA (FCGA) with a Generalized Crowding GA as the *process*. A proportional *controller* is used, while clustering is used as a *sensor*.

to more traditional feedback control loops, there are three important components.

- **Process** $P : \phi(t) \times \pi(t-1) \rightarrow \pi(t)$. The process to be controlled is a GA, currently a Generalized Crowding GA. The (multi-modal) fitness function f and the population $\pi(t)$ are also part of this computational process. The population consists of M individuals $\pi(t) = (\pi_1(t), \pi_2(t), ..., \pi_M(t))$.
- **Sensor** $S : \pi(t) \rightarrow l(t)$. The objective of the sensor is to measure the current process status. In our case, the number of niches (or local optima) in the current population is the statistic $l(t)$ to be sensed. This number is then sent to the controller.
- **Controller** $C : e(t) \rightarrow \phi(t)$. The controller is a mathematical function that maps the error $e(t)$ between the current number of local optima $l(t)$ and the set-point $r(t)$ to the scaling factor $\phi(t)$. The controller sends a signal to manipulate the GA process by adjusting the scaling factor $\phi(t)$. We use $r(t)$ to capture potential variation of the set-point through time t, however, it might be kept constant $r(t) = r$.

Now, we discuss the **Process**, the **Sensor**, and the **Controller** in more detail.

3.1 Process: Generalized Crowding

Various methods have been developed to avoid premature convergence in GAs and to find multiple solutions. Crowding is one such method developed by De Jong [11] and later modified by Mahfoud [1]. Crowding attempts to prevent premature convergence to a single local optimum by trying to distribute individuals across niches. Generalized Crowding introduces a scaling factor ϕ [5], which is the "knob" being controlled in FCGA.

The replacement rule for Generalized Crowding is expressed as follows. Assume that parent $p \in \pi(t)$ and child $c \in \pi(t)$ are paired in the grouping phase. Let $f(c)$ and $f(p)$ be the fitness of the child and the parent, respectively. We let P_c be the probability that child c replaces parent p:

$$P_c = \begin{cases} \frac{f(c)}{f(c)+\phi \times f(p)} & \text{if } f(c) > f(p) \\ 0.5 & \text{if f(c)=f(p)} \\ \frac{\phi \times f(c)}{\phi \times f(c)+f(p)} & \text{if } f(c) < f(p). \end{cases}$$

Putting $\phi = 0$ results in Deterministic Crowding, which mostly exploits the niches already found and does not usually result in the discovery of new niches. On the other hand, putting $\phi = 1$ results in Probabilistic Crowding which is biased more towards exploration and finding new niches. Using $0 < \phi < 1$ allows for a combination of such effects while $\phi > 1$ provides even more exploration than Probabilistic Crowding does.

A larger scaling factor ϕ leads to a greater probability of the less fit individual surviving a competition, on average. This results in more extensive exploration of less fit parts of the search space, including less-fit niches. Control over the scaling factor ϕ enables FCGA to *dynamically vary* the balance between exploration and exploitation. In FCGA, a time index $t \geq 0$ is added to ϕ, and $\phi(t)$ is varied by FCGA's controller.

3.2 Sensor: Clustering

A key component of the FCGA control loop is a reliable sensor. An unreliable sensor would make incorrect estimates of the current process status, which may mislead the controller to make poor adjustments to $\phi(t)$. The statistic that we would like to sense from the population is the number of local optima explored (niches). Hence, a natural way to arrive at such a sensor would be to employ a clustering algorithm.

Clustering solves the problem of finding clusters in a set of data points, given an assumption about the number of clusters. Specifically, the k-means clustering algorithm requires as input the number of clusters $k(t)$. The algorithm therefore does not directly solve our problem, namely estimating the number of sub-populations $l(t)$ in $\pi(t)$. However, there are methods to use a clustering algorithm, such as k-means, to estimate the number of clusters. One such method, which we use in this paper, is to find $l(t)$ through estimating the change in the k-means objective function (see below).

Since FCGA calls the clustering algorithm multiple times during each FCGA generation, we need clustering to be computationally efficient. This justifies our use of k-means clustering, which is linear in the number of data points in each iteration, as the basis for our sensor. The pseudo-code for clustering using k-means in the context of a GA population is presented as follows:

- **Given**: GA population $\pi(t)$ and the number of clusters $k(t)$.
- **Initialize**: Use random initialization, and then compute and store $k(t)$ centriods.
- **Iterate**: While there exists an individual $(\pi_i(t))$ which is not well clustered (an individual is closest to a cluster centroid to which it does not currently belong):
  - Reassign the mis-classified individual to its nearest cluster centroid
  - Recalculate the centroid of each cluster
- **Output**: Population $\pi(t)$ partitioned into $k(t)$ clusters.

We now describe the algorithm used in FCGA to estimate $l(t)$ using k-means. k-means minimizes an objective function (within-cluster sum-of-squares) for a given k. Thus, one can specify different k and measure the objective function. As we increase k, we would see a decrease in the objective function since we allow for greater model complexity. Now, when k approaches the "correct" number of clusters,

the objective function might observe a sharp decrease for well-defined clusters. The decrease is much smaller after this point, since adding more complexity does not help as the data is well-clustered using a smaller k. Using Occam's razor, the smallest k after the drop in the k-means objective function is chosen as the estimate for $l(t)$ for the t^{th} generation.[1]

3.3 Controller: Proportional Control

Inspired by the successful integration of adaptive control and feedback control with Bayesian network computation [26, 27], FCGA uses a simple Proportional-Integral-Derivative (PID) controller. The PID controlled is arguably the most widely used controller in industrial applications due to it being simple and well-suited in most situations. Although it has problems in dealing with non-linear systems and obtaining optimal control strategies, its advantages often dominate. Furthermore, the process to be controlled in our study is hard to model. Thus, another reason to use a proportional controller is that it can work without an explicit model of the process.

In our FCGA method using Generalized Crowding, we explore a simplified application of PID control. The parameter that we control in Generalized Crowding is the scaling factor ϕ, which has a direct impact on the exploration/exploitation trade-off. If we desire to increase the number of niches (local optima) found, we would desire a higher ϕ whereas in order to decrease the number of niches, we would decrease ϕ. This leads us to the following control strategy:

$$\Delta\phi(t) = \begin{cases} \gamma(r(t) - l(t)) & \text{if } l(t) \neq r(t) \\ 0 & \text{if } l(t) = r(t) \end{cases}$$

for some $\gamma > 0$, which we call the *control rate*. Although there exist analytical methods to determine γ, their application to the case in which the process is a Generalized Crowding GA is non-trivial. Thus we have explored a range of values for γ, optimizing it empirically (see Table 2).

FCGA currently uses proportional control to adjust the scaling factor based on the difference between the current number of clusters $l(t)$ and the set-point $r(t)$:

$$\phi(t+1) = \phi(t) + \Delta\phi(t).$$

There are three cases possibly needing control:

- $l(t) < r(t)$: In this case, the current number of niches $l(t)$ is less than desired and therefore we need to explore more niches. Thus we *increase* the scaling factor $\phi(t)$ proportional to the error $e(t) = r(t) - l(t) > 0$, to enable more exploration and less exploitation.
- $l(t) = r(t)$: When the current number of niches equals its set-point, we keep the scaling factor *constant* at its current value.
- $l(t) > r(t)$: In this case, the current number of niches $l(t)$ is greater than desired and therefore we need to lose some niches. Thus we *decrease* the scaling factor $\phi(t)$ proportional to the error $e(t) = r(t) - l(t) < 0$, to enable less exploration and more exploitation.

[1]There are alternative methods for estimating the number of clusters, such as X-means [24] and G-means [25]. A detailed study of these methods versus our method of estimating k through a change in the objective function is left for future research.

Table 1: Parameters used for FCGA experiments unless specified otherwise. Bold indicates defaults.

GA parameter	Value
Population size	$M \in \{25, \mathbf{100}, 400\}$
Mating	Random
Crossover (Uniform)	$P_c = 1$
Mutation (Uniform)	$P_m \in \{0.0125, 0.025, \mathbf{0.3}\}$
Survivor Selection	Generational
Scaling Parameter	Initially $\phi = 1$
Generations per run	$G = 500$
Number of runs	$N = 100$
Control rate	$\gamma \in \{0.1, \mathbf{1}, 10, 100\}$

3.4 Discussion

The idea of integrating feedback control and computing, or feedback computing, is not new [18]. However, we believe that feedback computing for the purpose of controlling an EA process—which is what we do in this paper—is new. We hypothesize that this amalgamation of evolutionary algorithms, feedback computing, and clustering can potentially bring substantial benefit to multi-modal optimization by means of genetic algorithms. While multi-modal optimization using genetic algorithms is well-established, it has so far lacked the capability of controlling the number of local optima being searched for by the algorithm. In hindsight, this appears to be a small but crucial limitation of previous work, since the number of local optima as well as the local optima themselves are of considerable interest to the EA user. Our introduction of a desired number of niches or local optima (a high-level parameter) as a set-point in a feedback control loop is key, and distinguishes the present work from previous efforts.

4. EXPERIMENTAL RESULTS

In this section, we present experimental results using the FCGA with Generalized Crowding. Our aim is to validate the behavior of FCGA when faced with the problem of achieving a user's desired number of niches in the population. Before discussing experimental results, we present the experimental setting, test functions, and metrics that we use for evaluation.

4.1 Experimental Setting

For all experiments, we simulate 100 runs of all GAs. For FCGA we explore different set-points $r(t)$ to illustrate the versatility of the algorithm. For each experiment, we look at the mean number of niches present in the population as generations progress. Further, unless specified otherwise, the initial scaling factor ϕ for FCGA is set to 1. Table 1 showcases the GA parameters that are used in all experiments unless specified otherwise. Generally, these values are close to those used in previous studies [4, 7].

4.2 Fitness Functions

Experiments are concerned with maximizing multi-modal fitness functions. We consider m-dimensional real-valued[2] fitness functions $f : \mathbb{R}^m \to \mathbb{R}$. Normalized versions of the 1D functions we use are shown in Figure 2. Both functions

[2]For the GA implementation used in all experiments in this paper, we use Java doubles to represent reals in $\mathbb{R}$. Doubles are 64-bit double precision floats that follow the IEEE 754 standard. This is the highest precision available in Java.

are representative of multi-modal problems. They also enable comparison as they have been used in several previous studies. Note that we add 1000 to the Schwefel function to make it positive for the range we focus on.

We also experiment with the 2D version of the Schwefel function:

$$f(x,y) = x\sin(\sqrt{|x|}) + y\sin(\sqrt{|y|}) + 2000,$$

where $x, y \in [-500, 500]$.

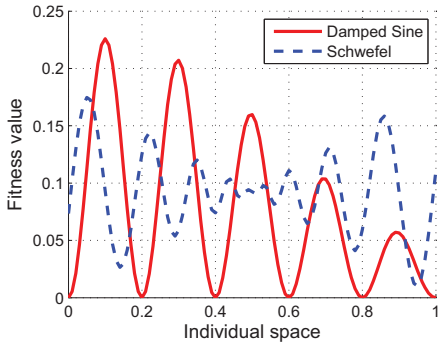


Figure 2: Normalized versions of the 1D Damped Sine function $f(x) = e^{-2(ln2)(\frac{x-0.1}{0.8})^2}\sin^6 5\pi x$ and the Schwefel function $f(x) = x\sin(\sqrt{|x|}) + 1000$.

4.3 Metrics

It is desirable to not only find the required number of niches but also to have the most fit solutions as evaluated by the fitness function f. In order to measure the overall quality of solutions, we report a statistic which is computed by dividing the mean of the sum total fitness of the best fit individuals in each desired niche (say, the top $r(t)$ niches) over all runs by the ideal value. The ideal value for some $r(t)$ is the total optimal fitness value attainable within the best $r(t)$ number of niches. This statistic, which we term the *Solution Quality Coefficient* (SQC), lies between 0 and 1, and would ideally be 1. If the complete search space is denoted by Ω, then $\Omega = \bigcup_{i=1}^{Z}\omega_i, \forall i = 1, ..., Z$, where ω_i is the i^{th} partition of Ω (i^{th} niche) such that each ω_i contains just one local optimum of fitness f_i. If $\{\omega_j | j = 1, ..., \lambda\}$ is the set of partitions of Ω containing the top $r(t)$ local optima as measured by the fitness function, then $\{f_j | j = 1, ..., \lambda\}$ is the set of the corresponding fitness values. Further, $\left\{\widehat{f}_j | j = 1, ..., \lambda\right\}$ is the set of the fitness values for the solutions found in each of $\{\omega_j | j = 1, ..., \lambda\}$. Then for set-point $r(t)$, SQC=mean$(\sum_{j=1}^{\lambda}\widehat{f}_j / \sum_{j=1}^{\lambda} f_j)$, with the mean taken over the individual SQC values of all runs. Note that this does not measure any statistic of non-desired niches and that this SQC metric changes as we change the set-point.

We use another metric to effectively measure the total deviation of the number of niches from the set-point. We term this metric ρ, and define it as $\rho = \sum_{i=1}^{N}(W(i) - r(t))^2$, where $W(i)$ is the number of niches present at the end of the i^{th} run out of the N runs. A low ρ signifies that the distribution of niches present in the population over multiple runs is not only sharp but is actually close to the desired ideal distribution as expressed by the set-point.

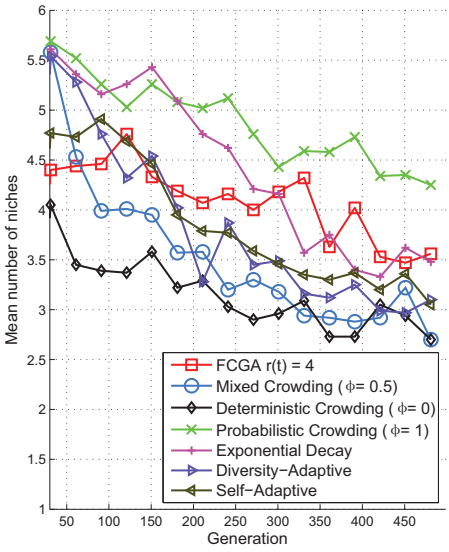


Figure 4: Mean number of niches for different algorithms on the 2-dimensional Schwefel function ($P_m = 0.3$ and $\gamma = 1$).

4.4 Feedback Control for Crowding

In this experiment, we develop a broader understanding of FCGA. We explore $r(t) = 3$ and $r(t) = 5$ for the Damped Sine function, and for the Schwefel function we set $r(t) = 2$ and $r(t) = 4$. We choose these set-points so that other algorithms do not converge to this many niches on average. This illustrates the control that we introduce in FCGA.

Varying Control Rate: We first vary γ to study its effect on the behavior of FCGA for the Damped Sine function. The left part of Table 2 shows the SQC values for feedback control on every generation. While SQC values vary along with generations, we always report the SQC value at the end of the run (for the last generation). We also report the standard deviation of the number of niches across the end of the 100 runs. A lower deviation reflects a better control through a sharper distribution of the number of niches. However, this statistic does not measure how close the distribution is to the desired $r(t)$ which is measured by ρ. Figure 3 showcases a number of scaling factor schedules affecting the number of niches as generations progress. We see that reasonable control of the actual number of niches is only possible with FCGA, wherein the mean number of niches present in the population closely matches the set-point $r(t)$.

Here we briefly examine the effect of varying γ-values on the convergence towards the set-point for feedback control. We choose to focus on the Damped Sine function for this study and vary the control rate γ through 0.1, 1, 10 and 100 for the set-points $r(t) = 3$ and $r(t) = 5$. Figure 5 shows the results for this study. Generally, the converge properties are not very different for the different γ-values, while $\gamma = 0.1$

	SQC: Control every generation								SQC: Control every 5th generation							
	$\gamma = 0.1$		$\gamma = 1$		$\gamma = 10$		$\gamma = 100$		$\gamma = 0.1$		$\gamma = 1$		$\gamma = 10$		$\gamma = 100$	
r	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
3	0.94	0.53	0.95	0.83	0.79	1.25	0.59	1.56	0.95	0.69	0.94	0.85	0.70	1.18	0.61	1.39
5	0.92	1.57	0.85	2.13	0.77	2.44	0.74	2.34	0.95	0.82	0.89	1.72	0.86	2.23	0.80	2.15

Table 2: Mean μ and standard deviation σ of measured SQC values for FCGA with control on every generation (left in table) or every five generations (right in table) for the Damped Sine function. This is for varying setpoints r and control rates γ with $P_m = 0.3$.

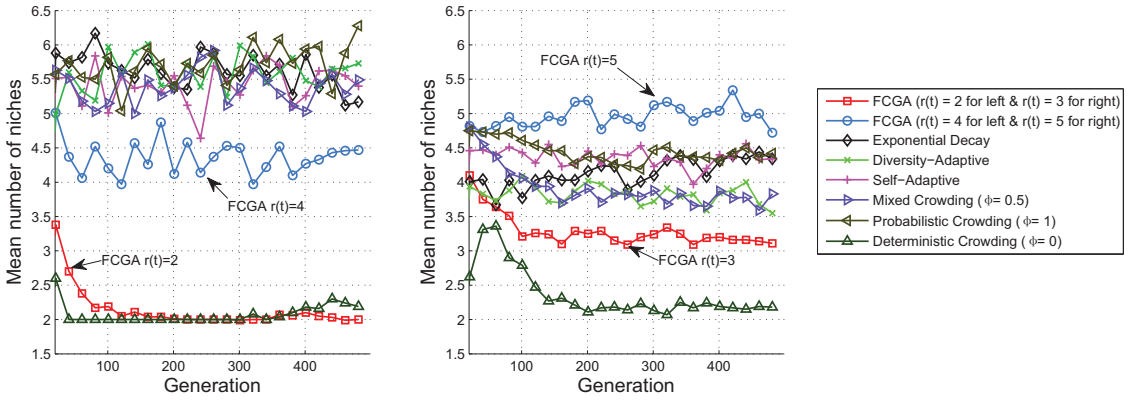
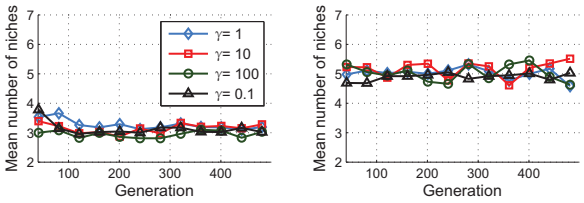


Figure 3: Mean number of niches for different algorithms evaluated on the 1D Schwefel function (left) and the 1D Damped Sine function (right). FCGA, using varying $r(t)$, allows us to converge to a desired number of niches, making the algorithm more versatile than previous niching methods ($P_m = 0.3$ and $\gamma = 1$).



(a) $r(t) = 3$ (b) $r(t) = 5$

Figure 5: Effect of varying γ on FCGA's performance using different set-points $r(t)$ for the Damped Sine function ($P_m = 0.3$ and $\gamma = 1$).

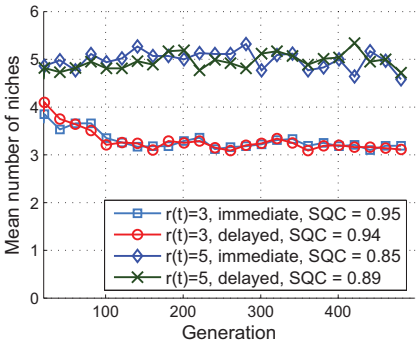


Figure 6: Effect of control attempts at every generation (immediate control) versus every 5^{th} generation (delayed control) for FCGA, using varying $r(t)$, on the Damped Sine function. SQC is reported for the last generation ($P_m = 0.3$ and $\gamma = 1$).

and $\gamma = 1$ perform the best overall as measured by the SQC in the left part of Table 2.

Immediate versus Delayed Control: Since GAs are inherently stochastic, once we change the scaling factor ϕ, the effects in the population might not be immediate. In fact, it might take a few generations to reflect change, during which time an estimate $l(t)$ would not be very accurate since the population would have "not settled" in distinct clusters after the last change. In order to compensate, we explore a minimal gap of $t_{\min}$ generations between each control attempt. A control attempt is estimation of $l(t)$, using k-means clustering, followed by a potential change in $\phi(t)$.

We now briefly explore the effect of immediate control versus delayed control. In immediate control, control attempts are made on every generation and in delayed control, they are made more sparingly. In this study, we examine $t_{\min} =$ 0 (immediate control) and $t_{\min} = 5$ (delayed control). We set γ=1 for this experiment. Figure 6 shows the mean number of niches for the two cases. Table 2 shows results where also γ is varied. Overall, varying γ has a bigger impact than varying $t_{\min}$, and using $\gamma = 0.1$ or $\gamma = 1$ is reasonable for both immediate and delayed control.

4.5 Varying GA Parameters

We now conduct a study of the effect of varying the population size M and the mutation rate P_m when FCGA eval-

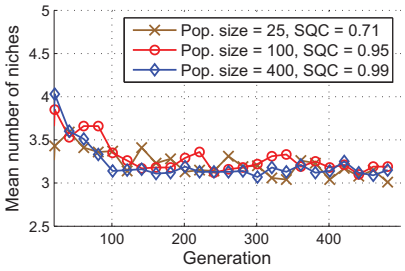


Figure 7: Effect of varying the size of the GA population on FCGA for the 1D Damped Sine function. SQC value is reported for the last generation ($P_m = 0.3$ and $\gamma = 1$).

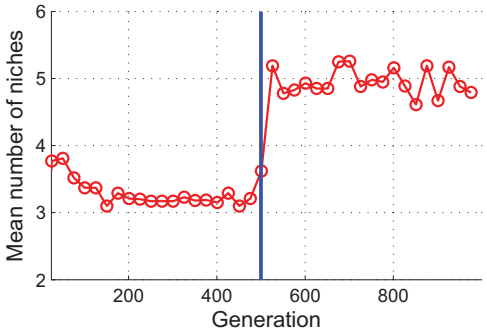


Figure 8: Effect of changing the set-point, at generation $t = 500$, from $r(t) = 3$ to $r(t) = 5$ on FCGA for the 1D Damped Sine function. The vertical line signifies the point of the set-point change ($P_m = 0.3$ and $\gamma = 1$).

uates the Damped Sine function. We examine population sizes $M \in \{25, 100, 400\}$ and mutation probabilities $P_m \in \{0.0125, 0.025, 0.3\}$ while keeping $r(t) = 3$ and $\gamma = 1$. Figure 7 and Table 3 show the results of these studies. While using a larger population has a higher computational cost, it seems to provide better solutions in terms of SQC, as expected. Regarding varying the mutation probability P_m, we observe in Table 3 that FCGA performs competitively over a broad range of mutation probabilities.

4.6 Changing the Set-point Mid-run

We now investigate the impact of a mid-run change of the FCGA set-point. In particular, we change $r(t) = 3$ to $r(t) = 5$ at $t = 500$ generations while keeping $\gamma = 1$. Figure 8 shows how changing the set-point $r(t)$ mid-run has a drastic and desired effect on the number of niches in the population. This experiment validates that the PID controller works, at least in this case.

In addition, this experiment illustrates an FCGA user's meaningful control over the exploitation/exploration tradeoff through the set-point $r(t)$. Such control is not offered by previous niching methods when applied to multi-modal optimization. A use case for such a mid-run change is when an FCGA user wants to see a larger (and more diverse) set of candidate solutions starting at $t = 500$.

4.7 Different Scaling Factor Schedules

We now experiment with different scaling factor schedules. The schedules that we compare against FCGA are the following. **Fixed Generalized Crowding** [5] uses a constant scaling factor ϕ. For $\phi = 0$ this becomes Deterministic Crowding and for $\phi = 1$ it becomes Probabilistic Crowding. We also experiment with $\phi = 0.5$ (Mixed Crowding). **Exponentially-Decaying Scaling** decreases the scaling factor, following an exponentially decaying function [7]. **Diversity-Adaptive Scaling** is a scheme where the population entropy is computed and the scaling factor is varied accordingly [7]. **Self-Adaptive Scaling** embeds the scaling factor in each individual's chromosome and the GA changes it adaptively in each generation [7].

We run all schedules with 100 individuals for 500 generations. Only for FCGA do we expect to see a change in the mean number of niches in response to a change in the set-point. Figure 3 shows the mean number of niches for multiple scaling factor schedules for the 1D Damped Sine and the Schwefel functions. Figure 4 showcases the means for the same scaling factor schedules for the 2D Schwefel function. As we see, all schedules other than FCGA seem to converge to an arbitrary mean number of niches. On the other hand, FCGA is able to converge quite reliably to the user defined set-point $r(t)$. Thus, FCGA allows us to have reasonable control over the number of niches, which opens up a new dimension in versatility for GAs.

Table 4 shows the run statistics for the 1D Damped Sine and Schwefel functions. For FCGA, we used the control loop at every 5^{th} generation with $\gamma = 0.1$ and ϕ was initialized at 1 for the comparison. For the Damped Sine function in Table 4, we find that feedback control performs the best for $r(t) = 5$. For $r(t) = 3$, it performs very competitively while keeping the number of niches very close to the set-point. For the Schwefel function in Table 4, we see that FCGA performs the best overall for $r(t) = 2$, however, the Schwefel function presents a more challenging scenario for $r(t) = 4$. Here we see that though FCGA returns a lower SQC, it provides a better distribution of niches close to that defined by $r(t)$ as seen by ρ being relatively smaller. Even though Deterministic Crowding seems to obtain a more ideal distribution (a lower ρ), it fails at achieving a high SQC.

5. CONCLUSION AND FUTURE WORK

This paper formulates and develops a Feedback Control GA (FCGA) with Generalized Crowding as the GA. The FCGA algorithm is used to better control the number of niches that a user wants the GA to explore. Thus, the algorithm is able to allocate more resources to exploring desired niches, resulting in a higher-quality and more directed search. The FCGA sensor is designed using k-means clustering while the controller uses proportional control. In experiments, FCGA reliably controls the number of niches and finds high-quality solutions as evaluated by the fitness function.

This work brings to the table many questions which we leave for further investigation. One of the key steps in FCGA is using k-means to determine the number of niches in the population. However, determining the "optimal" number of clusters in a population is still an active area of research

	Control every generation						Control every 5th generation					
	$P_m = 0.0125$		$P_m = 0.025$		$P_m = 0.3$		$P_m = 0.0125$		$P_m = 0.025$		$P_m = 0.3$	
r	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
3	0.98	1.18	0.98	0.98	0.95	0.83	0.96	1.09	0.99	1.08	0.94	0.85
5	0.90	1.75	0.95	1.60	0.85	2.13	0.92	1.32	0.95	1.28	0.89	1.72

Table 3: Mean μ and standard deviation σ of measured SQC values for FCGA with control on every generation (left in table) or every five generations (right in table) for the 1D Damped Sine function for varying values of P_m and r; here $\gamma = 1$.

	Damped Sine Function						Schwefel Function					
	Setpoint $r(t) = 3$			Setpoint $r(t) = 5$			Setpoint $r(t) = 2$			Setpoint $r(t) = 4$		
Schedule	SQC	σ	ρ	SQC	σ	ρ	SQC	σ	ρ	SQC	σ	ρ
FCGA ($r(t)$ varies)	0.95	0.69	6.92	0.95	0.82	8.24	1.00	0.40	4.00	0.75	2.38	23.91
Exponential Decay	0.87	1.33	18.08	0.85	1.33	15.32	0.98	2.66	44.77	0.82	2.66	31.00
Self-Adaptive	0.90	1.22	17.83	0.88	1.22	14.07	0.99	2.76	42.52	0.82	2.76	30.19
Diversity Adaptive	0.83	1.39	17.94	0.81	1.39	16.03	0.96	2.64	43.10	0.80	2.64	32.26
Deterministic Crowding	0.38	0.92	11.61	0.30	0.92	28.61	1.00	0.81	8.18	0.54	0.81	20.37
$\phi = 0.5$	0.99	0.89	12.00	0.75	0.89	14.96	0.98	2.85	43.78	0.81	2.85	31.38
Probabilistic Crowding	0.98	1.16	18.16	0.92	1.16	13.03	0.93	2.64	44.66	0.83	2.64	30.83

Table 4: Experimental results for different scaling factor schedules, including FCGA with varying setpoints, for the Damped Sine and Schwefel functions ($P_m = 0.3$ and $\gamma = 1$).

in machine learning. Also, although we show that the simple approach of proportional control works reasonably well, more sophisticated control schemes might bring substantial improvements. Further improvements can only help make FCGA more effective as a multi-modal optimization scheme. Finally, interesting results are expected from having setpoints for parameters other than the number of niches and also exploring the control perspective for evolutionary algorithms other than crowding.

6. REFERENCES

[1] S. W. Mahfoud. Crowding and preselection revisited. In *Parallel Problem Solving from Nature 2 (PPSN-II)*, pages 27–36. Elsevier, 1992.

[2] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In *Proc. of the 6th International Conference on Genetic Algorithms*, ICGA'95, pages 24–31, 1995.

[3] P. J. Ballester and J. N. Carter. Real-parameter genetic algorithms for finding multiple optimal solutions in multi-modal optimization. In *Proc. of the Genetic and Evolutionary Computation Conference*, GECCO'03, pages 706–717, 2003.

[4] O. J. Mengshoel and D. E. Goldberg. The crowding approach to niching in genetic algorithms. *Evolutionary Computation*, 16(3):315–354, 2008.

[5] S. F. Galán and O. J. Mengshoel. Generalized crowding for genetic algorithms. In *Proc. of the Genetic and Evolutionary Computation Conference 2010 (GECCO-10)*, pages 775–782, 2010.

[6] O. J. Mengshoel and D. E. Goldberg. Probabilistic crowding: Deterministic crowding with probabilistic replacement. In *Proc. of the Genetic and Evolutionary Computation Conference*, GECCO'99, pages 409–416, Orlando, FL, July 1999.

[7] O. J. Mengshoel, S. F. Galán, and A. De Dios. Adaptive generalized crowding for genetic algorithms. *Information Sciences*, 258:140–159, 2014.

[8] G. S. Hornby and J. C. Bongard. Accelerating human-computer collaborative search through learning comparative and predictive user models. In *Proc. of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO'12, pages 225–232, 2012.

[9] J. C. Bongard and G. S. Hornby. Combining fitness-based search and user modeling in evolutionary robotics. In *Proc. of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO'13, pages 159–166, 2013.

[10] D. E. Goldberg and L. Wang. Adaptive niching via coevolutionary sharing. *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*, pages 21–38, 1997.

[11] K. A. De Jong. *Analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[13] E. Aarts, J. Korst, and W. Michiels. Simulated annealing. In *Search Methodologies*, pages 187–210. Springer, 2005.

[14] I. Rechenberg. Evolution strategy: Natures way of optimization. In *Optimization: Methods and applications, possibilities and limitations*, pages 106–126. Springer, 1989.

[15] H.-P. Schwefel. *Numerical optimization of computer models*. John Wiley, 1981.

[16] T. Back. Self-adaptation in genetic algorithms. In *Proc. of the First European Conference on Artificial Life*, pages 263–271, 1992.

[17] S. Meyer-Nieberg and H.-G. Beyer. Self-adaptation in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, pages 47–75. Springer, 2007.

[18] J.L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. Wiley, 2004.

[19] T. F. Abdelzaher and N. Bhatti. Adaptive content delivery for Web server QoS. *Compututer Networking*, 31:1563–577, 1999.

[20] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury. Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. In *Proc. of Network Operations and Management Symposium (NOMS-02)*, pages 219–234, Florence, Italy, 2002.

[21] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus. Using control theory to achieve service level objectives in performance management. *Real-Time Systems*, 23(1/2):127–141, July 2002.

[22] C.-Z. Xu, B. Liu, and J. Wei. Model predictive feedback control for QoS assurance in webservers. *Computer*, 41:66–72, March 2008.

[23] C. Hollot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proc. of IEEE INFOCOM*, pages 1726–1734, 2000.

[24] D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proc. of the Seventeenth International Conference on Machine Learning*, ICML'00, pages 727–734, 2000.

[25] G. Hamerly and C. Elkan. Learning the k in k-means. In *Neural Information Processing Systems*, 2003.

[26] E. Reed, A. Ishihara, and O. J. Mengshoel. Adaptive control of Bayesian network computation. In *Proc. of 2012 - 5th International Symposium on Resilient Control Systems*, Salt Lake City, UT, August 2012.

[27] O. J. Mengshoel, A. Ishihara, and E. Reed. Reactive Bayesian network computation using feedback control: An empirical study. In *Proc. of BMAW-12*, Catalina Island, CA, August 2012.