Determining Relative Importance and Effective Settings for Genetic Algorithm Control Parameters

K. L. Mills kmills@nist.gov
Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, Maryland, 20899, USA

J. J. Filliben jfilliben@nist.gov
Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, Maryland, 20899, USA

A. L. Haines ahaines@warrenrogers.com
Warren Rogers Associates, Middletown, Rhode Island 20842, USA

Abstract

Setting the control parameters of a genetic algorithm so as to obtain good results is a long-standing problem. We define an experiment design and analysis method to determine relative importance and effective settings for control parameters of any evolutionary algorithm, and we apply this method to a classic binary-encoded genetic algorithm (GA). Subsequently, as reported elsewhere, we applied the GA, with the control-parameter settings determined here, to steer a population of cloud-computing simulators toward behaviors that reveal degraded performance and system collapse. GA-steered simulators could serve as a design tool, empowering system engineers to identify and mitigate low-probability, costly failure scenarios. In the existing GA literature, we uncovered conflicting opinions and evidence regarding key GA control parameters, and effective settings to adopt. Consequently, we designed and executed an experiment to determine relative importance and effective settings for seven GA control parameters, when applied across a set of numerical optimization problems drawn from the literature. This paper describes our experiment design, analysis, and results. We found that crossover most significantly influenced GA success, followed by mutation rate and population size, and then re-randomization point and elite selection. Selection method and the precision used within the chromosome to represent numerical values had least influence. Our findings are robust over 60 numerical optimization problems.

Keywords

Genetic algorithms, orthogonal fractional factorial experiment design, sensitivity analysis.

1 Introduction

We aim to devise tools that system engineers can use to explore model-based designs for low-probability, costly failure scenarios (Taleb, 2010). As a first approach, we are investigating guided random search techniques that can steer a population of model simulators toward parameter combinations that yield degraded performance or system collapse. We identified genetic algorithms (GAs) as a search technique that might be well suited for our problem. GAs can find good solutions within a large, ill-defined

search space, and can be readily adapted to a wide variety of search problems (Mitchell, 1998).

The classic binary-encoded GA we adopted exhibits a number of control parameters (explained in Section 2), such as: population size, selection method, number of elite individuals, generations at which to re-randomize the population, number of crossover points when swapping chromosomes between pairs of individuals, and mutation rate. As we discuss in Section 3, we consulted the literature and found conflicting advice about relative importance and effective settings for these GA control parameters. Further, we found only a small collection of studies (none definitive) attempting to guide selection of GA control settings. Some authors (DeJong, 2007) advised that one should experiment with a GA in the intended application and use those experiments to determine the most effective control settings. Other researchers (Bartz-Beielstein et al., 2005) provide techniques that can be used to sequentially search the parameter space of evolutionary algorithms to find optimal control settings for individual problems. While sequential parameter optimization (SPO) might prove effective for a wide range of numerical optimization problems, sequentially searching the GA in our intended application is not feasible because the cloud-computing simulators we are exploring require significant computation time. Instead, we designed and conducted an experiment and analysis to search, across a set of 60 numerical optimization problems, for the most important GA control parameters, and effective settings to use. While a classic binary-encoded GA may not exhibit the best performance on numerical optimization problems, we show that a sufficient set of numerical optimization test problems can give significant insight into the effect of various GA control parameters.

This paper makes two main contributions: (1) we define (in Section 4) an experiment design and analysis method that can be adapted to determine relative importance and effective settings for control parameters in any evolutionary algorithm, and (2) for a classic binary-encoded GA we determine (in Section 5) the relative importance and effective settings for seven control parameters. Our findings are robust over 60 numerical optimization problems. The GA control settings discovered using our method proved effective when applied to steer a population of cloud-computing simulators in search of potential failure scenarios (Mills et al., 2013). Our method can also be used to discover effective, initial parameter settings for SPO, when applied to individual evolutionary algorithms used to solve specific problems. Finding effective starting parameter settings could shorten search time when using SPO.

2 Genetic Algorithm Under Test

GAs are a subclass of evolutionary algorithms (EAs). EAs comprise a collection of heuristic methods that use techniques, such as mutation, recombination, and selection (inspired by genetics and biological evolution) to search for optimal solutions to difficult problems. For our application, we adopted a classic GA (Mitchell, 1998) that encodes problem variables as fixed-length bit strings, which enable simple crossover and mutation operations. Some EAs represent problem variables as vectors of floating-point numbers. To achieve good performance, floating-point encoding requires altering the typical crossover and mutation operators used by classic GAs (Janikow and Michalewicz, 1991). Ongoing research (Ali et al., 2005; Rahnamayan et al., 2008; Sahin, 2011) in EAs continues to investigate the best formulation for crossover and mutation operators

when using floating-point representation.

2.1 Fixed-Length Binary Encoding

The classic GA in our experiments uses simple binary encoding, which can represent the full range of variable types (Boolean, integer, and floating-point) required for our cloud-computing simulator. A user first identifies the variables for a problem, and specifies the minimum and maximum values and precision (or quantization) for each variable, and then the GA computes the number of bits per variable, and thus the chromosome (i.e., bit-string) length required to encode all problem variables. The GA includes a control parameter, *precision scaling*, which can increase or decrease the quantization originally specified by a user.

In most of the numerical optimization problems explored in this paper, the GA generates binary encoding of quantized floating-point variables (some variables are integers). The number of bits required to encode quantized variables can allow more values than necessary. In such cases variable values are derived by linear scaling. For example, 10 values require at least 4 bits (3 bits are too few), which can encode as many as 16 values. In this example, each binary value encodes 0.625 (10/16) of a real value.

While simple binary encoding can lead to Hamming walls, where single-bit mutations cannot transform a bit pattern into a neighboring bit pattern, and can thus contribute to premature convergence in numerical optimization problems, our application (searching for failure scenarios in cloud-computing simulations) was not hindered by such issues because we sought general failure outcomes, rather than specific optimal numerical values. For that reason, we were not deterred by the fact that our adopted GA used simple binary encoding instead of Gray encoding, which would eliminate Hamming walls. We discuss this issue further in Section 4, where we identify additional GA control parameters that could be studied, using the same method we describe.

2.2 Control Parameters

The GA begins by generating a random population of individuals, where each individual consists of an appropriate length bit string representing values for every variable of a problem to be solved. The *population size* is a control parameter of the GA. The GA evaluates the fitness of many populations of individuals over time, where each population is called a generation. The population of individuals for generation $n+1$ is created through some transformation of individuals composing generation n.

After completing each generation, the GA considers whether or not the population should be re-randomized, which involves randomly regenerating all or part of the next generation. The GA includes a control parameter, *reboot proportion*, which determines how many generations must be completed before a population is re-randomized.

Whenever the population of individuals is created for an upcoming generation, some number of the most fit individuals (i.e., the elite) from the previous generation can be included unchanged. The GA has a control parameter, *elite selection percentage*, which defines how many individuals from generation n will be placed unaltered into generation $n + 1$. Such elite individuals can be placed into a population whether or not

the remaining individuals will be generated randomly or by transforming individuals from the previous generation.

The GA also includes a control parameter, *selection method*, that defines the algorithm used to select individuals from generation n for inclusion into a candidate pool, where some individuals from generation n may be included multiple times, while others may not be included at all. Given a pair of individuals, chosen randomly from the candidate pool, a GA control parameter, *number of crossover points*, determines how bits will be swapped (or recombined) among the pair. Subsequently, the GA iterates over each bit representing each individual in the population of recombined individuals, while deciding whether or not the bit should be inverted. A GA control parameter, *mutation rate*, specifies the probability that any given bit will be inverted.

3 Related Work

We first consulted the seminal investigation (DeJong, 1975) of behavior in classic GAs, where DeJong studied a small set of numerical optimization problems. In his study, DeJong reported that the best solutions were obtained when GA control parameters were set as follows: a population size of 50, a 0.60 probability that pairs of individuals crossover at a single point, a mutation probability of 0.001, with the most elite individual surviving to the next generation, while the remainder of the population is replaced with transformed individuals from the previous generation. Other researchers (Grefenstette, 1986; Goldberg, 1989; Schaffer et al., 1989; Spears and DeJong, 1992) soon followed with additional studies of the best settings to adopt. Unfortunately, the findings of these studies were not always in agreement. For example, Grefenstette (1986) found that mutation rates above 0.05 were generally harmful, but that failing to use mutation led to poor performance, while using elitism was helpful and population size should be kept in the range of 30 to 100. Goldberg (1989) found that mutation can replace key genetic material that might be lost through crossover, but Tate and Smith (1993) dispute this view, pointing out that high mutation rates can be disruptive when most of the population is replaced with each generation. Schaffer et al. (1989) found an interaction between crossover, mutation rate and population size, specifically suggesting that small populations are quite sensitive to mutation rate and less sensitive to crossover rate. Further, they found that with larger populations, low rates of mutation and crossover proved most effective.

Studies continue (Baeck, 1996; Odetayo, 1997; Digalakis and Margaritis, 2001; Charbonneau, 2002; Rojas et al., 2002; Boyabatli and Sabuncuoglu, 2004; Nunez-Letamendia, 2007; Diaz-Gomez and Hougen, 2009; Arenas et al., 2010; Kapoor et al., 2011) up to the present, and with continued inconsistencies among findings. For example, Rojas et al. (2002) found the most important GA control parameters are selection method, mutation rate, and population size, but that the number of crossover points (1 or 2) did not have much influence, while the probability of crossover did. Arenas et al. (2010) found that crossover and mutation must be combined to have the best outcomes, but that mutation rate must remain within a narrow range because too much mutation prevents convergence to a good solution and too little mutation leads to premature convergence. Charbonneau (2002) argues that mutation rate should adapt, increasing with decreasing population diversity and decreasing with increasing diversity. Digalakis and Margaritis (2001) found that a population size between 200-250 proved optimal, combined with a crossover rate of 70-75%, and that absence of mutation led to poor

outcomes, whereas a small mutation rate improved results. Boyabatli and Sabuncuoglu (2004) report that crossover does not have significant influence on outcomes, while high mutation rate provides better results. Baeck (1996) agrees with Charbonneau that mutation rate should be variable, but argues that crossover must be used in order to find a global optimum. Kapoor et al. (2011) suggest that mutation plays a critical role for simple problems with few parameters, but that crossover is important for complex problems. In general, they conclude that one should combine high crossover rate with low mutation rate and a correctly sized population. Nunez-Letamendia (2007) found that, while various combinations of crossover and mutation probabilities lead to optimum solutions for different problems, the best outcomes usually (but not always) occur when combing high crossover probability with low mutation rate. All in all, the results reported in the literature provide a rather confusing picture.

We found two retrospective articles (DeJong, 1999, 2007) from the researcher who initiated investigation of control parameters in GAs. DeJong (1999) reports a general lack of theory to guide selection of population size, use of selection strategies, and choice of representation. He observed that a few theoretical models exist in the evolutionary strategies research community for finding optimal mutation rates, but that those theories have been unable to explain anomalous results from experiments, some of which find benefits from adaptation in both mutation rate and recombination mechanism. In 2007, DeJong took a broader view, considering what was known with respect to the wider community of evolutionary algorithms (EAs). He observed that while it appears that adapting mutation rate on-line during execution provides advantages, most EAs are deployed with a default set of static parameter values that have been found quite robust in practice. (The method we describe in Section 4 provides a rigorous means to determine effective values to use as default settings for control parameters.) DeJong also observes that larger population size increases parallelism, which aids in solving complex problems, but that there is diminishing return to increasing population size. He reports that choosing a selection method is difficult due to interactions with population size.

DeJong (2007) also notes a promising approach involves periodic restarts of an EA search, using historical information to adapt control parameters in a type of *meta-search* to find the best control parameters that yield optimal outcomes for individual problems. Bartz-Beielstein et al. (2005) define a specific method, sequential parameter optimization (SPO), for conducting such a meta-search. Despite the existence of methods such as SPO, DeJong believes that EAs pre-tuned with default parameter values for particular problem classes will continue providing better performance than EAs that attempt to dynamically adapt too many control parameters for specific problems. So, even after 30+ years of research, the question of best settings for EA and GA control parameters has no widely agreed answer.

A critical review of the previous research into GA control parameters suggests some reasons that might underlie the current state of uncertainty. First, published studies examine performance of GAs only under a small number of numerical optimization problems, ranging from one (Boyabatli and Sabuncuoglu, 2004) to 14 (Digalakis and Margaritis, 2001) problems. These represent an unacceptably small sample, providing little robustness in results. This shortcoming does not appear in studies (Ali et al., 2005; Rahnamayan et al., 2008; Sahin, 2011) that compare the effectiveness and efficiency of

different proposed EAs. EA comparison studies typically use more than 50 problems, and those problems are drawn from the same set (Adorio, 2005) of difficult numerical optimization problems. Second, published GA studies examine different sets of control parameters over different ranges of values. Most studies also examine more values for some control parameters than for others, which can lead to unbalanced results. Third, since 1990, very few GA studies attempt to replicate the results of prior studies. Fourth, only one study (Arenas et al., 2010) uses analysis of variance (ANOVA) to determine the statistical significance of each control parameter studied.

Kleijnen (2010) describes the state of the art in design and analysis of computer experiments, where the main concepts are derived from the statistical design of experiments (Box et al., 2005). Bartz-Beielstein (2006) embodies those concepts in a three-phase approach to find optimal parameter settings for EAs, when applied to solve specific optimization problems. Phase one uses highly-fractionated (resolution III) two-level experiment designs to identify the most important set of control parameters from among a potentially large set. Phase two uses less fractionated (resolution IV and V) two-level experiment designs to assess interactions among important control parameters and to identify regions of the parameter space that might be fertile areas over which to seek optimal settings. Phase three uses SPO, sequential parameter optimization (Bartz-Beielstein et al., 2005), to search for optimal settings for control parameters, when applied to solve some specific optimization problem. SPO amounts to a meta-search for optimal control-parameter settings that yield the best performance on some specific optimization problem. SPO meta-searches must be undertaken for each new optimization problem of interest.

We also adopt fundamental concepts from statistical design of experiments, but we use only the equivalent of the second phase of the Bartz-Beielstein approach. In particular, we use high-resolution designs to simultaneously identify relative importance of control parameters, to assess interactions among control parameters, and to find the most-effective control-parameter settings (from among those considered). We adopt this one-phase approach because: (1) we are performing screening/sensitivity analysis instead of optimization, (2) sequential parameter optimization is not feasible in our intended application, as the cloud-computing simulators we plan to investigate require significant computation time, and (3) the GA we investigate has only a handful ($k = 7$) of control parameters.

We use four levels per control parameter instead of two levels, as typically used in statistical experiment designs. For the GA we investigate, which has only seven control parameters, using four levels enables us to search across 16,384 parameter combinations, rather than the 128 combinations possible with a two-level design. Further, while we cannot directly investigate the GA in the context of our intended cloud-computing application, we do investigate the performance of the GA across a large variety and number of test problems. Exploring substantially more parameter combinations across a wide variety and large number of test problems somewhat offsets our inability to use SPO. On the other hand, the specialized analysis methods typically used with two-level experiment designs cannot always be applied directly to four-level designs. Thus, as described in Section 4 and Section 5, we introduce several analysis innovations that enable us to extract essential information from our experiment results.

In the next section, we outline an experiment design and analysis method that can be used to obtain statistically significant results robust over many numerical optimization problems. Our method can be used to identify effective default parameter settings for EAs, enabling good performance on many search problems. Our proposed method is intended to complement, rather than replace, SPO. Specifically, our method can be used for applications, such as ours, where SPO meta-search is infeasible. When compared with the two (pre-SPO) phases of the Bartz-Beielstein approach, our one-phase method considers substantially more parameter combinations across a wide variety and number of test problems. For this reason, where SPO meta-search is feasible, our method could be used to identify effective starting values for EA control parameters. Next, we discuss our proposed experiment design and analysis method in the context of a classic GA, but the method appears general enough to be applied to study and tune control parameters of other search algorithms.

4 Experiment Design and Analysis Method

Robust and rigorous statistical experiments naively require full factorial designs, which examine each parameter under study at every possible value and then analyze the influence of the parameters on outcomes. For most experiments, a full factorial design proves impractical because even a handful of parameters can encompass an infeasible space of possibilities. For example, suppose (as in our case) a system under study is controlled by seven parameters. Assuming each parameter can be represented within a 32-bit integer, the space of possibilities is of $O(10^{67})$.

A natural first step to reduce the search space is to limit the number of values at which to examine each parameter. For example, in our case, we examine each parameter at only four values (i.e., levels), which reduces the search space to 4^7. Restricting parameters to take on a reduced set of values has obvious limitations: only a small number of parameter values are explored, and extrapolating from the results assumes a model behaves monotonically in the range between chosen values. On the other hand, adopting a reduced-level design provides some advantages (Box et al., 2005): (1) requires relatively few runs per parameter, (2) identifies promising directions for future experiments (and may be augmented with thorough local explorations), (3) fits naturally into a sequential strategy (such as SPO), and (4) forms the basis for further reduction in parameter combinations through use of fractional factorial designs.

Even using only a few levels, a full factorial design can still be prohibitively expensive. For example, in our case, a full factorial exploration of 4^7 parameter combinations for 60 numerical optimization problems requires about a million executions of the GA, and we intend to run each GA execution through 500 generations, thus we would need to execute over 60 billion function evaluations (assuming an average population size) to complete a full factorial experiment. To further limit the number of experiment executions, one can adopt orthogonal fractional factorial (OFF) experiment designs, or the geometrically equivalent orthogonal Latin hypercube (OLH) designs. (Box et al., 2005).

OFF and OLH experiment designs sample a full factorial design space in a balanced and orthogonal fashion. Balance ensures good effect estimates by reducing an estimator's bias and variability. Orthogonality ensures good estimates of two-term

interactions. Balance is achieved by ensuring that each level of every factor occurs an equal number of times in the selected sample. Orthogonality is achieved by ensuring that each pair of levels occurs an equal number of times across all experiment parameters in the selected sample. OFF and OLH designs exhibit good space-spanning properties, which aid screening, sensitivity, and comparative analyses. On the other hand, highly fractionated OFF or OLH designs can have poor space-filling properties, which are necessary for optimization analyses. As explained below, we adopt a 4^{7-2} OFF design, which achieves a reasonably space-filling (1/16th) sample of our full 4^7 experiment space. If the number of factors and levels were to increase significantly, then we would need to increase the number of experiment runs, or else adopt some alternate experiment design method. For example, Cioppa and Lucas (2007) describe an algorithm for generating "nearly" orthogonal Latin hypercube designs that exhibit improved space-filling properties, but at the cost of inducing small correlations, which can complicate the analysis.

By ensuring balance and orthogonality, OFF designs achieve two desirable properties, given the limits of the selected sample size. First, main effects estimates are representative of main effects that would be found in a full factorial experiment. This means that a list of experiment parameters, ranked by main effects, tends to be close to the true ordering that would result from a full factorial experiment. Second, main effects estimates exhibit an uncertainty as small as possible, given the sample size. More specifically, running an OFF experiment design provides an estimate of main effects with uncertainty on the order of $\sigma\sqrt{2/(n/v)}$, where n is the number of experiment runs, v is the number of values used per parameter, and σ is the standard deviation among the underlying observations. This formula simply adapts the standard uncertainty estimate for main effects in two-level OFF experiment designs (Box et al., 2005) to account for the fact that our design uses v levels per factor, which means there are n/v observations at each level rather than $n/2$.

Using OFF principles as a basis, we defined an experiment design and analysis method encompassing seven steps: (1) factor identification and level selection, (2) problem-set selection, (3) OFF experiment design, (4) experiment execution and data collection, (5) per-problem factor analysis, (6) per-problem factor-interaction analysis, and (7) results summarization. We explain each step in turn, using exemplary details from our study of the GA that was outlined in Section 2.

The first step entails identification of the experiment *factors* (i.e., control parameters) to investigate, along with selection of *levels* (i.e., values) at which each factor will be examined. In our case, as experiment factors, we used the seven control parameters from a classic GA, as identified in Section 2, and we selected four levels for each factor, as shown in Table 1. We selected population sizes (factor x_1) between 50 and 200 because those values encompassed the range found to provide good performance in previous studies, as recounted in Section 3.

The GA included only two selection methods (factor x_2), stochastic universal sampling (SUS) (Baker, 1987) and paired tournament (also known as 2-tournament) with replacement (T) (Mitchell, 1998), and so we investigated both. The stochastic universal sampling algorithm contained no tunable parameters. In 2-tournament, a pair of evaluated individuals is drawn randomly (uniform distribution) from the previous

Factor	Level 1	Level 2	Level 3	Level 4
x_1 **Population Size**	50	100	150	200
x_2 **Selection Method**	SUS	T(r=.60)	T(r=.75)	T(r=.90)
x_3 **Elite Selection %**	0	2	4	8
x_4 **Reboot Proportion**	0	0.1	0.2	0.4
x_5 **# Crossover Points**	0	1	2	3
x_6 **Mutation Rate**	Adaptive	0.001	0.0055	0.01
x_7 **Precision Scaling**	1/2	1	2	4

Table 1: Seven factors (control parameters) and four levels (parameter values) per factor investigated in the experiment.

generation, and then with some probability r the more fit of those individuals is selected for inclusion in the recombination and mutation processes that will form the next generation. The lesser fit individual in the pair is selected otherwise. As shown in Table 1, we used three values for r. Had the GA included other available selection methods (e.g., q-tournament selection, roulette wheel selection or rank selection), then we could have included two of them (in place of T with multiple values for r), as additional parameter values in our experiment. Had we been able to explore such a larger set of selection methods, then a subsequent experiment iteration could have been designed to explore any tunable parameter values associated with the best performing selection method. (Experiment iteration leverages the ability of OFF designs to support thorough local explorations.) Since specific GA selection methods are likely to have different tunable parameters, exploring them when exploring heterogeneous selection methods would be impractical.

To probe the utility of elitism, we varied elite selection percentage (factor x_3) from 0 to 8% of the population size. Including a value of 0 allows elite selection to be disabled, which permits us to investigate findings, described in Section 3, that elitism is one key to GA success. By exploring a range of other elite selection percentages, we can establish whether too much elite selection impairs the ability of a GA to converge to good solutions. Subsequent experiment iterations could be conducted to search more precisely for effective elite selection percentages, starting from the best value found in our experiment.

To investigate the influence of population re-randomization, we ranged reboot proportion (factor x_4) from 0 to 0.4 of the generations executed. When reboot proportion is set to 0, the population chromosomes are randomized only once, at initiation of the GA. When reboot proportion is set to 0.1, chromosomes for non-elite members of

the population will be randomized 9 more times after the initial randomization, once after incremental completion of each 10% of the total number of generations. Similarly, when reboot proportion is set to 0.2 (or 0.4), non-elite members of the population will be randomized 4 (or 2) times subsequent to the initial randomization, i.e., after incremental completion of each 20% (or 40%) of the total number of generations.

For crossover (factor x_5) we specified different numbers of crossover points, ranging from 0 to 3. When the number of crossover points was 0 no crossover occurred; otherwise, for each pair of recombining individuals, the locations of the specified number (1, 2, or 3) of crossover points were selected randomly and then the indicated bits were swapped. While the GA did allow a probabilistic crossover threshold, to avoid overweighting investigation of crossover techniques, compared with other factors, we did not complicate crossover by adding a probabilistic threshold, nor by considering uniform crossover, where each bit has some probability of being swapped. If desired, these more complicated schemes could be investigated in a subsequent OFF experiment, where other factors are fixed to the best levels discovered here.

For mutation rate (factor x_6) we specified three levels with rates ranging from small (0.001) to high (0.01). We reserved the remaining level for an adaptive algorithm (Charbonneau, 2002) that adjusts the mutation rate between 0.001 and 0.1, lowering the rate when a population exhibits a wide spread in fitness values and raising the rate when the spread is narrow. Adopting these settings enabled us to investigate the various conflicting findings about mutation, as recounted in Section 3.

We also investigated the influence of problem-variable discretization on GA performance by varying precision scaling (factor x_7) from 1/2 to 4, including the value of 1. Precision scaling of 1 means the GA performs a search with variable granularity as determined by the user when specifying the encoding for a problem. A precision scaling value of 1/2 means variable granularity is set to be twice as coarse (requiring fewer chromosome bits) as specified by the user, while values of 2 and 4 mean that variable granularity is set two or four times, respectively, as fine (requiring more chromosome bits) as the user specified. Investigating this factor allowed us to evaluate whether increasing or decreasing the size of a search space would improve the performance of the GA.

Readers familiar with GAs will note that, while we explored a wide range of GA control parameters, there are other such parameters that could have been investigated. In some cases, our investigation was restricted by the range of available parameters in the GA. For example, as discussed above, the GA offered only two of the possible selection methods sometimes available in GAs. Similarly, GAs often offer Gray encoding to forestall Hamming cliffs, as we discussed in Section 2.1, or other encoding methods, such as permutation encoding or real value encoding. The GA used only simple binary encoding. In other cases, we chose to limit the specific control parameters investigated in order to maintain a balanced experiment that did not place too much emphasis on any particular control parameter. For example, as discussed above, the GA offered the possibility of setting a threshold for comparing against a uniformly sampled random variate to determine whether or not crossover would take place for specific pairs of individuals. We elected to design our experiments so that crossover always took place, unless crossover was disabled. Had additional experiment factors (control parameters)

been of interest, OFF experiment design techniques provide suitable means to include them. One means to expand the number of control parameters in an experiment would simply be to include them within the experiment factors. For example, had the GA offered them, additional selection methods could have been added as parameter values (levels), replacing the multiple r values we used for 2-tournament selection.

In cases where particular control parameters identify algorithms with tunable variables, OFF experiment design can be applied sequentially. For example, various selection methods have unique tunable variables. To establish appropriate settings for those variables, an experimenter can first run an OFF experiment design, fixing selection method, to determine which other control parameter values yield best performance on a set of test problems. Subsequently, an experimenter can design OFF experiments to explore a range of tunable values for each selection method, while fixing other control parameters to values determined in a preceding experiment. A similar approach can be used to investigate potential unbalanced control parameters. For example, an experiment can first set the crossover threshold so that crossover (if enabled) will always occur, and then run a subsequent experiment that varies the crossover threshold, while fixing other parameters to the values that performed best in a preceding experiment. Similarly, an experimenter can follow up any OFF experiment with a related experiment that increases the range of parameter settings investigated for each control parameter. The second experiment will establish whether or not the results from the first experiment are robust (the same) over the increased range of value settings.

Step two in our method selects a set of numerical optimization problems to evaluate when the GA is configured with various combinations of level settings for the seven factors identified in Table 1. A full factorial (4^7) exploration of all combinations would require executing the GA 16,348 times for each selected problem. Perhaps this explains why previous studies were limited to 14 or fewer problems? To increase the robustness of our results, we aimed for around the same number (55-58) of problems used in recent studies comparing various EA algorithms, as outlined in Section 3. We selected 53 test problems from that set, where problems ranged from relatively simple (two parameters) to complex (100 parameters). We chose seven additional problems, two from the search literature (Ingber, 1993; Brent, 2002) and five from the statistics literature (Box and Bisgaard, 1996; Box et al., 2005; Saltelli et al., 2004), bringing the total number of test problems to 60 (see Appendix A for a specific list with sources). Using 60 test problems extends the scope of our results substantially, compared to previous studies of GA control parameters that we found in the literature.

Step three applies OFF design to reduce significantly the number of required experiment runs. We chose a 4^{7-2} OFF design, reducing the number of parameter combinations per GA run to 1024, which, assuming an average population size, requires about 64 million function evaluations for each test problem and just over 3.8 billion for all 60 problems. Using a 4^{7-2} OFF design yields good estimates of main effects, with an uncertainty of 0.088 σ, while expending only 6.25% of the computational resources required for a full factorial experiment. (See Appendix B for a comparison of results from a 4^{7-2} OFF design against results from a full factorial design for three of our test problems.)

Step four executes the OFF design against each test problem, and collects the

resulting data. For each problem, the GA is run once under each of the 1024 parameter combinations identified by the OFF design. For a given parameter combination, the GA is executed for 500 generations and the output (y) is recorded as the maximum value discovered by the GA. Let n (= 1024) be the total number of parameter combinations, k (= 7) be the number of factors and n_ℓ (= 4) be the number of levels within each factor. The output for each problem is a table containing n rows, where each row includes $k + 1$ columns: a level setting $\ell_j|_{j=1,2,\ldots\ell_n}$ for each of the factors $x_i|_{i=1,2,\ldots k}$ and the resulting y value.

Step five analyzes the results for each test problem. This requires computing for each factor: the absolute (E_i) and relative (RE_i) effects, and statistical significance (F_{cdf_i}). A multidimensional factor-analysis plot (e.g., Figure 1) reveals the relative factor importance, and the most effective setting for each factor. Computing (E_i) requires determining $\overline{y}_{ij}$, the average y values for factor x_i at level setting ℓ_j. Let S be the set of all OFF parameter combinations and $S_{ij} = \{x \in S | x_i = \ell_j\}$,

$$\overline{y}_{ij} = \frac{\sum_{d \in S_{ij}} y_d}{|S_{ij}|} \tag{1}$$

computes the average value of y when $x_i = \ell_j$, and then $E_i = \mathbf{max}\{\overline{y}_{ij}\}$ - $\mathbf{min}\{\overline{y}_{ij}\}$. $RE_i = 100 \cdot E_i/\overline{\overline{y}}$, where

$$\overline{\overline{y}} = \frac{\sum_{d=1}^{n} y_d}{n}. \tag{2}$$

To measure statistical significance we used analysis of variance (ANOVA) by computing $F_{cdf_i} = P(F_{stat_i} \leq F_{\nu_1,\nu_2})$, where F_{ν_1,ν_2} is the reference F distribution with $\nu_1 = n_\ell - 1$ and $\nu_2 = n - n_\ell$ degrees of freedom and

$$F_{stat_i} = \frac{\sum_{r=1}^{n/n_\ell} \sum_{j=1}^{n_\ell} (\overline{y}_{rj} - \overline{\overline{y}})^2/(n_\ell - 1)}{\sum_{r=1}^{n/n_\ell} \sum_{j=1}^{n_\ell} (\overline{y}_{rj} - \overline{y}_{ij})^2/(n - n_\ell)}. \tag{3}$$

After computing E_i, RE_i, and F_{cdf_i} for each factor, we plot the results, as shown for example in Figure 1 for numerical optimization problem #2. The figure plots 28 points: mean $\overline{y}_{ij}$ for each level (ℓ_j) of each factor (x_i). The dashed horizontal line reports $\overline{\overline{y}}$. The bottom of the plot (just above the x-axis) reports the E_i, RE_i, and F_{cdf_i} for each factor.

The factor exhibiting largest E_i has most influence on GA success for the problem. We highlight this factor, mutation rate (x_6), with a dashed rectangle in Figure 1. Ordering E_i values from high to low reveals the relative importance of each factor, as we indicate by labeling the rank-ordered factors from 1 (most important) to 7 (least important) in Figure 1. Where E_i values are tied, we order the ranking based on visual inspection of the plot.

Large effects are not necessarily statistically significant, but the F_{cdf_i} values adjudicate that question. We scaled F_{cdf_i} to a percentage, which means that the probability of type I error is $p < 0.05$ when $F_{cdf_i} > 95$ and $p < 0.01$ when $F_{cdf_i} > 99$. On our plots, we mark $p < 0.05$ with a * symbol and $p < 0.01$ with a ** symbol beneath F_{cdf_i} values associated with statistically significant factors. In Figure 1, statistically

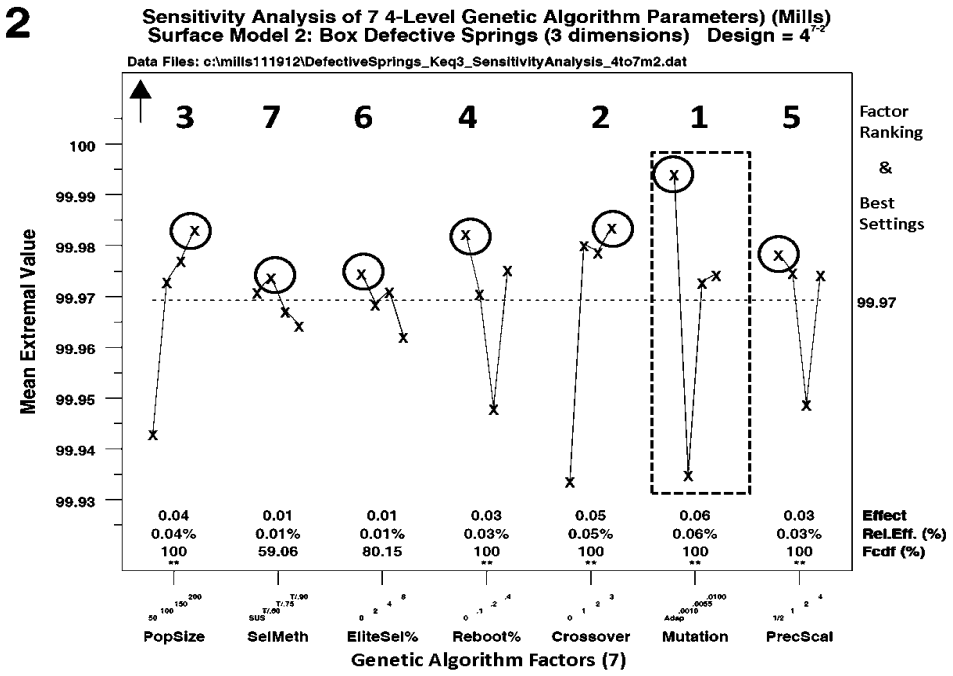

Figure 1: Factor-analysis plot for numerical optimization problem #2: maximizing the percentage of non-defective springs.

significant factors include: population size (x_1), reboot proportion (x_4), number of crossover points (x_5), mutation rate (x_6), and precision scaling (x_7).

The plot also identifies the best setting for each factor, as represented by the maximum $\overline{y}_{ij}$. In Figure 1, we circle the best $\overline{y}_{ij}$ for each factor: population size (200), selection method (2-tournament, r=0.60), elite selection % (0), reboot proportion (0), number of crossover points (3), mutation rate (adaptive), and precision scaling (1/2 as fine).

Step six of our method computes two-term interactions for all pairs of experiment factors for each test problem. We computed (6 x 7 =) 42 two-term interactions for each of the 60 test problems. Figure 2 shows four cells extracted from the upper left corner of a 7x7 matrix containing a per-factor interaction analysis for numerical optimization problem #1: maximizing chemical yield. Each row of the matrix corresponds to one factor and contains seven columns. One column in each row (the cell appearing on the matrix diagonal) identifies the factor for which two-term interactions are analyzed, while the six remaining columns show the interaction of that factor with each of the other six factors. The extracted submatrix shows interaction analyses for population size (x_1) and selection method (x_2).

The cell in the upper right of Figure 2 shows five curves. The first curve is the main effects plot for population size. Each data point is the average of 256 experiment data points, where population size is set to each of its four levels. Each data point

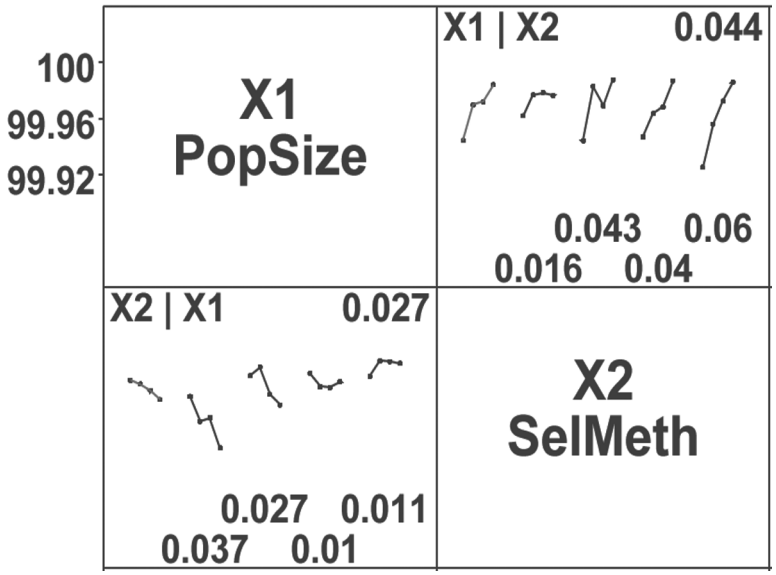

Figure 2: Four cells extracted from the upper left corner of a 7x7 matrix containing a per-factor interaction analysis for numerical optimization problem #1: maximizing chemical yield. Two of the cells contain two-term interaction analysis plots for population size (x_1) and selection method (x_2). The upper-left cell establishes the y-axis scale for the interaction analysis plots. The upper-right cell plots five curves. The first curve is the main effects plot for x_1. Each additional curve reports the effect of varying x_1 while holding x_2 constant, first at level 1, then level 2 and so on to level 4. The number under each curve reports the related effect. The number in the upper right hand corner of the cell is the overall interaction effect, computed by subtracting the smallest of the four interactions from the largest. The lower-left cell depicts the interaction analysis plots when varying x_2 while holding x_1 constant at each of its four levels.

on the second curve gives the average of only 64 data points, where population size varies across its four levels, while holding selection method to its first level. Curves three through five vary population size, while holding selection method to its second, then third, and finally fourth level. The numbers below each curve report the size of each interaction effect. The number in the upper-right corner of a cell represents the overall interaction effect (IE_{ij}), which is computed by subtracting the smallest of the four interactions from the largest. Note that interaction effects are not necessarily symmetric for pairs of factors. For example, Figure 2 reports an interaction effect of 0.044 for $x_1|x_2$ and of 0.027 for $x_2|x_1$.

The seventh, and final, step in our method summarizes and analyzes results across all test problems. We describe and demonstrate this step in the next section.

5 Results and Discussion

Given E_i, RE_i, F_{cdf_i}, IE_{ij}, and related factor-analysis and factor-interaction plots representing GA performance on each of the 60 test problems, we summarized the analysis

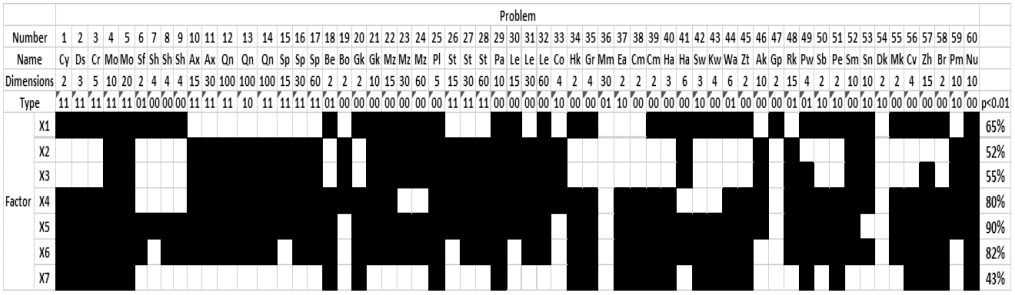

Figure 3: Factor-significance matrix with 420 cells, one row for each of 7 factors and one column for each of 60 test problems, where cells in black indicate the identified factor had significant effect ($p < 0.01$) for the identified problem. The problems are identified by number, name, type, and dimension, all of which key to more complete problem descriptions given in Appendix A. A 61st column gives the percentage of test problems on which each factor was statistically significant ($p < 0.01$).

results using five main techniques: (1) factor-significance matrix, (2) main-effects-rank histograms, (3) results-summary table, (4) factor-rank/most-effective-level table, and (5) factor-interaction summary. We describe each of these in turn, illustrating them with data collected from our experiment, and we discuss the findings revealed by these techniques.

Figure 3 shows a factor-significance matrix for the 60 test problems solved by the GA. A visual scan of the matrix reveals that factors x_1, x_4, x_5, and x_6 had most statistically significant influence on GA success. Factors x_2, x_3, and x_7 had less influence. The last column in Figure 3 provides a quantitative verification of these visual impressions by reporting the percentage of test problems on which each factor was statistically significant ($p < 0.01$).

We note that GAs attempt to balance two competing processes: exploitation and exploration. Exploitation leverages good problem solutions in a given generation by selecting them as the basis for solutions forming the next generation, and by adding (optionally) the very best (elite) solutions unchanged to the next generation. Exploration expands the scope of a search by transforming the best solutions in a given generation into recombined and mutated variants, and by occasionally re-randomizing an entire population, except for any elite solutions from the previous generation. Each generation can explore a population of solutions in parallel, thus larger population sizes appear likely to increase the ability of a GA to exploit good solutions, as well as to explore mutated and recombined variants. The factor-significance matrix shown in Figure 3 suggests that exploration (i.e., crossover, mutation, and population re-randomization) has more influence on GA success than does exploitation (i.e., selection and elitism). Further the matrix also indicates that population size somewhat influences GA success.

The 60 test problems can be categorized using the scheme from Jamil and Yang (2013), which classifies optimization functions based on continuity, differentiability, scalability, separability, and modality. Over ninety-three percent (56) of the test functions are continuous and differentiable; only the three Step functions and the

Corana function are discontinuous and non-differential. Just over half (35) of the test functions are scalable, though we did not make use of this property in our experiments. Sixty percent (36) of the problems are non-seperable and sixty-five percent (39) are multimodal. For each problem the type field in Figure 3 reports separability and modality, using two bit positions. The first bit position denotes separability (separable = 1) and the second bit position denotes modality (uni-modal = 1). Examining Figure 3 closely reveals no specific pattern with respect to problem type or number of dimensions. This means that the significance of factor influence applies across the 60 test problems, regardless of function type or dimension.

The factor-significance matrix in Figure 3 also shows that no factors (or only one) significantly influenced GA performance for three problems: #36 Multi-mod (Mm: 30 variables), #47 Goldstein-Price (Gp: 2 variables) and #54 Deckkars and Aarts (Dk: 2 variables). To gain more insight, we consulted the factor-analysis plots for each of these problems.

For the Multi-mod problem, three levels of each factor successfully found the theoretical best solutions, while one level for each factor did not. Because three levels in each factor led to successful solutions, the factors could not be distinguished as statistically significant; however, no crossover performed worse than crossover (at any level) and the lowest level of fixed mutation rate performed worse than higher mutation rates (including adaptive mutation). For the Goldstein-Price problem, only population size (x_1) exceeded the higher significance threshold ($p < 0.01$), while precision scaling (x_7) exceeded the lower threshold ($p < 0.05$). The smallest population size (50) led to poor solutions, as did the finest (four times) precision scaling. The exploration factors (re-randomization point, crossover, and mutation rate) each exhibited $F_{cdf_i} > 80\%$, suggesting influence, but not establishing statistical significance. For the Deckkars and Aarts problem, elite selection percentage (x_3) exceeded $p < 0.05$, with no elite section performing worst. Population size, selection method, and crossover exhibited influence, but not to the level of statistical significance. This analysis reveals that altering factor levels did influence GA success on these problems, but not sufficiently to pass our chosen level ($p < 0.01$) for statistical significance. We concluded that there was nothing particular about these three problems that prevented the GA control parameters from influencing the success of the GA. Examining these problems in detail reinforced our decision to establish $p < 0.01$ as the threshold for statistical significance.

Figure 4 shows main-effects-rank histograms for each of the seven control parameters across the 60 test problems. A scan of the histogram reveals that crossover (x_5) had low relative influence on main effects for very few problems; in contrast, precision scaling (x_7) had high relative influence for very few problems. The relative influence of mutation rate (x_6) and population size (x_1) varied more evenly across the problems, with mutation rate exhibiting moderate relative influence (rank of 3) among a cluster of problems. Similarly, the relative influence of selection method (x_2) varied fairly evenly across the problems, while showing increased influence (rank of 2) among a cluster of problems. The relative influence for reboot proportion tended to clump in the middle (ranks 3 and 4) for most problems, exhibiting high relative influence on very few problems. For one problem (#36 Multi-mod), all factors exhibited the same ranking. The reason for this was discussed above.

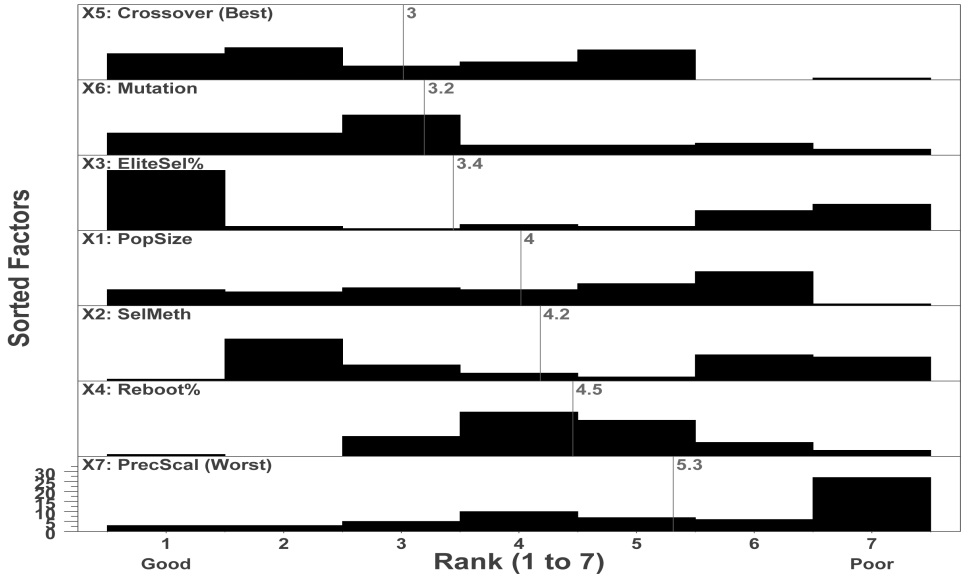

Figure 4: Seven main-effects-rank histograms (one for each of the 7 factors), where the 7 cells in each histogram represent the frequency with which the factor exhibited most (1) to least (7) influence on main effects across the 60 test problems. The factor histograms are presented ordered, top to bottom, by average rank from highest (3 for factor x_5, crossover) to lowest (5.3 for factor x_7, precision scaling).

The test problems can be divided roughly into three categories, based on problem dimensionality: (1) 31 low ($\leq$ 10) dimensional problems, (2) 22 moderate ($\leq$ 30) dimensional problems and (3) 7 high ($>$ 30) dimensional problems. The relative influence of crossover, mutation rate and population size tended to moderate on the highest (100) dimensional problems, while the influence of elite selection percentage (and to some degree selection method) increased for those problems. Analysis of the main-effects-rank histograms, while providing only relative information, suggests that (if possible) a future iteration of a set of test problems would benefit from extension to include about 10 additional problems with high dimensionality. Readers should bear this information in mind when considering our findings. In general, analysis of the main-effects-rank histograms supports our argument in Section 3 that conflicting conclusions from previous studies of GA control parameters likely arose (in part) because those studies considered an insufficient number of problems (at most 14 and usually many fewer). Further, our analysis also suggests the need to increase the number of high dimensional problems included in the set of numerical optimization problems often used to evaluate evolutionary search algorithms.

The average ranks of the seven factors, shown by vertical lines in Figure 4, indicate their influence on main effects, i.e., the degree to which the factor influences the ability of the GA to find good numerical solutions. First, factors x_5 (crossover) and x_6 (mutation rate) most influenced the main effects produced across the test problems. These two factors also proved among the most statistically significant factors, as identified from Figure 3. Second, factor x_3 (elite selection percentage) showed next

most influence on main effects. Examining x_3 on the factor-analysis plots for each problem revealed that the main effects of this factor arose when comparing a positive elite selection percentage against no elite selection. From this, we infer that including some amount of elite selection has a large influence on the ability of the GA to find good solutions to numerical optimization problems. On the other hand, as discussed previously when considering Figure 3, the influence of elite selection percentage is statistically significant in just over half of the problems. Third, factors x_1, x_2, and x_4 showed comparable (i.e., average rank of about 4) levels of influence on main effects. While x_1 (population size) and x_4 (reboot proportion) were also among the must statistically significant influences on GA success (see Figure 3), x_2 (selection method) was not. Examination of x_2 on the factor-analysis plots for each problem showed that the main effects arose when comparing 2-tournament selection with r=.60 and r=.75 against SUS and 2-tournament selection with r=.90. From this, we inferred that 2-tournament selection with insufficient selection pressure (i.e., r probabilities that are too low) hampered the ability of the GA to find good solutions to numerical optimization problems. Third, factor x_7 (precision scaling) appeared to have least influence on the ability of the GA to find good solutions.

Comparing Figure 3 (statistical significance) to Figure 4 (size of effect) gives a different ordering of factors influencing GA success. For this reason, we decided to rank factors based on a combination of statistical significance and relative effect. Next, we present and discuss quantitative summaries encompassing both these characteristics, leading to a relative ranking of factor importance.

Table 2 provides a quantitative summary of relative effect, statistical significance, and most effective setting, across all 60 test problems, for each factor. The quantitative measures of significance ($p < 0.01$) provide a precise ranking of factors from most to least influential: number of crossover points (x_5), mutation rate (x_6), reboot proportion (x_4), population size (x_1), elite selection percentage (x_3), selection method (x_2), and then, least influential, precision scaling (x_7).

Table 2 also reports the average, standardized, relative effect, $\overline{\mathrm{S}(RE)}$, for each factor across all test problems. Recall (Section 4) that relative effect measures how much difference a factor makes in the best outcome found for individual numerical optimization problems. This measure, which is independent of statistical significance, gauges the influence of a factor on effective outcomes.

To compute $\overline{\mathrm{S}(RE)}$ for a given factor i, we first standardized RE_i. Let N (= 60) be the number of test problems and K (= 7) be the number of GA control parameters. For each numerical optimization problem f (= 1..N) we can define the set, E_f, of relative effects for each GA control parameter k, where $E_f = \{RE_k \in E_f | k = 1..K\}$. We then standardized RE_i

$$\mathrm{S}(RE_i)_f = \frac{RE_i - \mathbf{min}\{RE_k \in E_f\}}{\mathbf{max}\{RE_k \in E_f\} - \mathbf{min}\{RE_k \in E_f\}}. \quad (4)$$

Subsequently, we averaged $\mathrm{S}(RE_i)_f$ over all N test problems

$$\overline{\mathrm{S}(RE_i)} = \frac{\sum_{f=1}^{N} \mathrm{S}(RE_i)_f}{N}. \quad (5)$$

Factor	$\overline{S(RE)}$	Significant $p < 0.01$	Most Effective Level on Individual Functions Level 1	Level 2	Level 3	Level 4
Population Size (x1)	0.71	39 (65%)	0 (0%)	2 (3%)	12 (20%)	52 (87%)
Selection Method (x2)	0.22	31 (52%)	35 (58%)	8 (13%)	11 (18%)	14 (23%)
Elite Selection % (x3)	0.43	33 (55%)	14 (23%)	12 (20%)	15 (25%)	26 (43%)
Reboot Proportion (x4)	0.35	48 (80%)	24 (40%)	4 (7%)	2 (3%)	46 (77%)
Crossover (x5)	0.87	54 (90%)	0 (0%)	14 (23%)	8 (13%)	46 (77%)
Mutation (x6)	0.65	49 (82%)	37 (62%)	4 (7%)	10 (20%)	12 (20%)
Precision Scaling (x7)	0.24	25 (43%)	25 (42%)	16 (27%)	5 (8%)	18 (30%)

Table 2: Results-summary table reporting for each factor the average, standardized, relative effect, $\overline{S(RE)}$, and the number (and percentage) of 60 test problems on which the factor significantly ($p < 0.01$) influenced GA success. Also, for each level of each factor, the number (and percentage) of test problems on which the level led the GA to the best answer. Note that in some cases more than one level led the GA to the best answer ($|\overline{y}_{ix_i=l_j} - \overline{y}_{ix_i \neq l_j}| \leq 0.0001$), thus the related percentages can sum to more than 100.

When computed for each factor, this yields the values shown in the $\overline{S(RE)}$ column of Table 2.

The $\overline{S(RE)}$ values in Table 2 quantify the influence of each factor on the ability of the GA to achieve the best outcomes over all the test problems. As Table 2 shows, crossover (x_5), population size (x_1), and mutation rate (x_6) have most influence on relative effect, followed by elite selection percentage (x_3) and reboot proportion (x_4). Precision scaling (x_7) and selection method (x_2) have relatively little influence. Comparing the statistically significant influence of factors (column labeled $p < 0.01$) against their influence on relative effect (column labeled $\overline{S(RE)}$) finds general agreement, though population size ranks second on relative effect but only fourth on statistical significance, and reboot proportion ranks third on statistical significance but only fifth on relative effect.

Finally, Table 2 identifies the most effective level settings found for each factor: population size = 200, selection method = SUS, elite selection percentage = 8%, reboot proportion = 0.4, number of crossover points = 3, mutation rate = adaptive and precision scaling = 1/2 as fine as specified by the user. Differences in the percentages for each level setting for a given factor provide a measure of how much changing the level influenced GA performance.

Summary data from Table 2 can be extracted to produce a factor-rank/most-effective-level table (Table 3), which answers the two main questions addressed by our experiment design and analysis: (1) What is the relative importance of the GA control parameters evaluated? and (2) What is the most effective level setting (among those examined) to use for each control parameter? Table 3 reports factor rank and most effective level setting for each GA control parameter. Because rankings differed somewhat when based solely on either statistical significance or relative effect, we chose to rank the factors on D, the Euclidean distance of each factor from an ideal outcome,

Rank	Factors	$p<0.01$	$\overline{S(RE)}$	D	Most Effective Level (Setting)
1	# Crossover Points (x_5)	0.90	0.87	0.16	4 (3 points)
2	Mutation Rate (x_6)	0.82	0.65	0.39	1 (Adaptive)
3	Population Size (x_1)	0.65	0.71	0.45	4 (0.4)
4	Reboot Proportion (x_4)	0.80	0.35	0.68	4 (200)
5	Elite Selection % (x_3)	0.55	0.43	0.73	4 (8%)
6	Selection Method (x_2)	0.52	0.22	0.92	1 (SUS)
	Precision Scaling (x_7)	0.43	0.24	0.95	1 (1/2 as fine)

Table 3: Factor-rank/most-effective-level table ordering GA control parameters by increasing distance (D) from ideal (see Figure 5) and then partitioning them into four groups based on relative differences in D. The table also reports for each factor: the fraction of 60 test problems on which the factor had statistically significant ($p < 0.01$) influence on GA success, the average relative effect ($\overline{S(RE)}$) and the most effective level (setting).

which is the point (1,1) on a Cartesian plot of X := $p < 0.01/100$ against Y := $\overline{S(RE_i)}$. Using the data from Table 3, we illustrate such a plot as Figure 5.

Figure 5 shows a grouping of three factors–crossover (x_5), mutation (x_6) and population size (x_1)–closest to the ideal point (1,1). The next closest factor appears to be reboot proportion (x_4), which is followed by another grouping of three factors–elite selection percentage (x_3), precision scaling (x_7) and selection method (x_2)–that appear farthest from the ideal point. Computing the Euclidean distance between each factor and the ideal point yields the values reported under the column labeled D in Table 3, which we used to rank the influence of each factor, across all 60 test problems, in order of increasing distance from (1,1).

The D values in Table 3 identify a clear ranking: (1) number of crossover points showed most influence, followed by (2) mutation rate and (3) population size, and then by (4) reboot proportion and (5) elite selection percentage. Selection method and precision scaling proved least influential. Evidently, two of the exploration control parameters (crossover and mutation rate) had large statistical significance and also substantial influence on relative effect. The other exploration parameter (reboot proportion) had a statistically significant influence on 80% of the test problems, but the relative effect of the parameter was not large. Population size, which affects both exploration and exploitation, influenced GA outcomes substantially, and was statistically significant on about two-thirds of the problems. The exploitation control

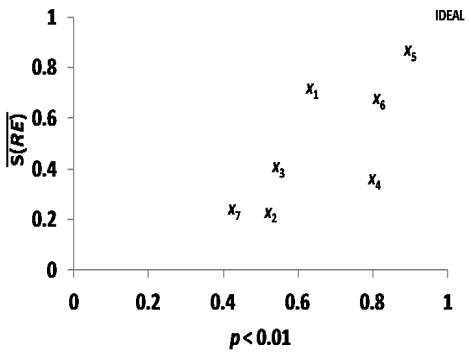

Figure 5: Cartisian plot of $p < 0.01/100$ (x axis) against $\overline{\mathrm{S}(RE_i)}$ (y axis) for each of the seven GA control parameters, and for the ideal point (1,1).

parameters (elite selection percentage and selection method) exhibited only modest influence on relative effect, and were statistically significant on only about half of the problems.

Figure 6 shows a factor-interaction summary, which plots interaction effects, standardized and averaged across all test problems, for each of the 42 two-term interactions. For each two-term interaction (i, j), we standardized the interaction effect (using the same method shown above in equation (4), but substituting IE_{ij} for RE_i) to compute $\mathrm{S}(IE_{ij})$ for each of the test problems. We then averaged those 60 values – using the method shown in equation (5) – to give $\overline{\mathrm{S}(IE_{ij})}$, representing the average interaction effect for (i, j) across the entire set of test problems.

We plot these values on the y axis of Figure 6 against the factor-pair identifier (i, j) on the x axis, which is sorted from largest to smallest $\overline{\mathrm{S}(IE_{ij})}$. The plot identifies two groups: largest interactions and smallest interactions, highlighted within labeled rectangles in Figure 6. The group of largest interactions can be divided into three subgroups, highlighted in Figure 6 within ovals. We extract the numeric values for $\overline{\mathrm{S}(IE_{ij})}$ in the group of largest interactions and report them in a sparse matrix in the upper right of Figure 6. Since those 14 values correspond to seven pairs of pairs – $\{(i, j), (j, i)\}$ – we average the $\overline{\mathrm{S}(IE_{ij})}$ and $\overline{\mathrm{S}(IE_{ji})}$ for each of these pairs of pairs, and then report those seven averages, in the lower left of Figure 6, as a sorted table.

In comparing the seven values representing $(\overline{\mathrm{S}(IE_{ij})} + \overline{\mathrm{S}(IE_{ji})})/2$ in Figure 6 to the $\overline{\mathrm{S}(RE_i)}$ values in Table 3, we see that the $\overline{\mathrm{S}(RE_i)}$ values for the three largest main effects (x_5, x_1, and x_6) exceed the values for the largest standardized interaction effects. This implies that those main effects are more influential on GA success than two-term interactions. On the other hand, the seven two-term interactions exceed the main effects values for the other factors. For that reason, we decided to examine those seven two-term interactions in detail by reviewing the per-factor interaction analyses for those interactions across the test problems.

The largest two-term interaction occurs between selection method (x_2) and elite

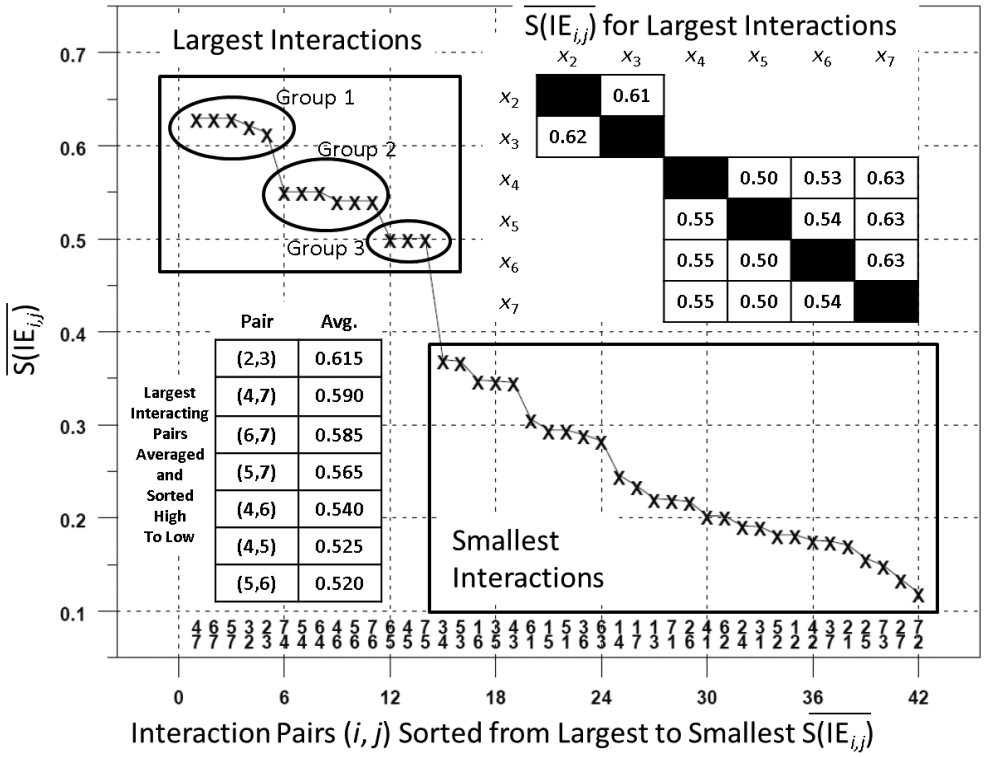

Figure 6: $\overline{\mathrm{S}(IE_{ij})}$, standardized relative interaction effect for each of 42 two-term pairs of factors, plotted on the y axis against pair identifiers (i, j) on the x axis. The x axis pair identifiers are sorted from largest to smallest $\overline{\mathrm{S}(IE_{ij})}$. Two rectangles highlight the smallest and largest interaction effects. Ovals identify three subgroups among the largest interactions. A sparse matrix reports numeric values for the 14 largest $\overline{\mathrm{S}(IE_{ij})}$. Pairs of two-term interactions are averaged – $(\overline{\mathrm{S}(IE_{ij})} + \overline{\mathrm{S}(IE_{ji})})/2$ – and reported in a table of seven values, sorted from largest to smallest.

selection percentage (x_3). In about two-thirds of the test problems, combining no elite selection with 2-tournament (T) selection when r=0.6 leads to poor outcomes. This implies that elite selection can compensate somewhat for the low selection pressure of T (r=0.6), while lack of elite selection allows the low selection pressure to cause the GA to perform poorly. The importance of this interaction is low because SUS performed better than 2-tournament selection with any value for r.

The next three largest two-term interactions involve precision scaling (x_7) combined with reboot proportion (x_4), mutation rate (x_6), and number of crossover points (x_5). In general, each of these interactions arise on about half the test problems. The precise nature of these interactions is somewhat muddled, as they vary with problem (but not dimension) and also among specific combinations of levels. For example, re-randomizing frequently (10% of the time) combined with precision scaling of 1 and re-randomizing somewhat frequently (20% of the time) combined with precision scaling of 2 (twice as fine) both seem to be combinations that can yield bad GA outcomes,

depending on problem type. On the other hand, re-randomizing never or infrequently (40% of the time) and coarsening (1/2 as fine) or highly increasing (four times as fine) precision scaling seem to mute this interaction. Similarly, low or moderate mutation rates combined with increases in precision scaling yielded poor results on specific problems. On the other hand, coarsening the precision scaling combined with having less than three crossover points led to bad outcomes on specific problems. Using three crossover points appeared to compensate somewhat for coarsening precision scaling. We concluded that these interactions were not particularly important because: (1) the nature of the interactions varied, (2) the interactions appeared on only about half the test problems, and (3) the interactions varied on specific problems rather than by problem dimension.

On about half the problems, two other two-term interactions arose for some combinations of reboot proportion (x_4) with mutation rate (x_6) or number of crossover points (x_5). A combination of frequent re-randomization (after every 50 generations) and moderate mutation rate (0.0055) yielded bad outcomes on specific problems. Similarly, on selected problems, a combination of no re-randomization and one or two crossover points led to poor GA outcomes. For some problems, three crossover points compensated for lack of re-randomization. We concluded that these interactions were not particularly important, as the specific interactions varied, they appeared on only about half the test problems, and they were limited to specific problems rather than to particular problem dimensions.

The final large two-term interaction concerned number of crossover points (x_5) and mutation rate (x_6). We found that the combination of one crossover point with adaptive mutation performed poorly on 21 of the test problems. The importance of this interaction is low because three crossover points performed best on 77% of the test problems.

In summary, there were only seven parameter interactions that showed somewhat large effects, when averaged across the test problems. For the reasons outlined above, we concluded that none of the interactions was particularly important, and that the success of the GA was driven primarily by number of crossover points, mutation rate, and population size. Earlier work by Schaffer et al. (1989) found an interaction among crossover, mutation rate, and population size. While our experiments found an interaction between crossover and mutation rate, we found no significant interactions between population size and any other control parameter for the 60 problems we examined. Interactions between population size and crossover, or between population size and mutation rate, did appear in our experiments, but the size of those interactions was small.

Next, we consider the four right-most (Level 1 to Level 4) columns in Table 2, which provide additional insights about specific GA control parameters. First, note that the GA performed much better when the population re-randomized after 200 generations (best level on 77% of the problems) or when there was no population re-randomization (best level on 40% of the problems) than when population re-randomization occurred every 50 or 100 generations. While showing that population re-randomization can improve GA performance, this result also indicates that re-randomizing a population too frequently destroys the ability of the GA to converge to good problem solutions.

This raises the question of determining the best re-randomization frequency, which remains for investigation in a future expansion of this experiment, where reboot proportion might be set to levels 0, 0.4, 0.5, and 0.6. If such experiments reveal no additional insights, then perhaps the GA had already converged to good solutions somewhere between 100 and 200 generations, so that re-randomization points occurring at or beyond 200 generations would have no effect on GA performance. Investigating this question would require future experiments where reboot proportion would be set to levels 0, 0.25, 0.3, and 0.35.

Second, the highest percentage value (8%) we used for elite selection yielded substantially better GA performance than lower percentages (2% and 4%), which performed indistinguishably from no elite selection. While Grefenstette (1986) also found elitism to be useful, his study moved only a single elite individual to the next generation. Our results suggest few benefits accrue from moving only a single elite individual from one generation to the next. In fact, our results suggest that a substantially larger proportion of elite individuals is required for a GA to reap significant benefits. On the other hand, the degree of exploration may decrease as elite selection percentage increases, leading to diminishing returns and to less effective outcomes. This raises the question of determining the best elite selection percentage, which remains for future investigation, where elite selection percentage might be set to levels 0, 8%, 10%, and 12%.

Third, the data provide some evidence that coarsening parameter discretization can improve GA success rate. Specifically, a precision scaling of 1/2 (as fine as specified by the user) was the best level setting on 42% of the test problems. Apparently, for these problems, coarsening parameter discretization reduces the search space, which can aid the ability of a GA to converge to a higher fitness value. This result might, however, be due to the fact that the maximum or minimum values were integers for 36 of the test problems. Also, this finding might not hold for GA applications outside the domain of numerical optimization. Other coding questions could also be investigated. For example, additional experiments could be conducted to compare the effectiveness of simple binary encoding against other encoding methods, e.g., Gray encoding, permutation encoding, and real-value encoding; however, including additional encoding methods would require extending the GA, which currently supports only simply binary encoding.

Finally, while the study reported here encompasses 60 test problems, we caution readers not to over interpret our results. Our findings hold only for the GA algorithm described in Section 2, and only for the range of levels investigated in our experiments, and only for the 60 numerical optimization problems on which we evaluated the GA. Within these bounds, we observe that our results suggest some general findings with respect to control parameters for a classic binary-encoded GA applied to numerical optimization.

We found crossover and mutation most influential in GA success. This suggests that GA control parameters associated with exploration have more influence on GA success than control parameters associated with exploitation. Further, we found that crossover is an essential element for successful GA performance, as disabling crossover exhibited poor GA success in our experiments. Three crossover points yielded much better GA success than fewer crossover points. We suspect that increasing the number

of crossover points will yield diminishing returns at some point, because collections of bits representing key genetic material would be sliced up and lost. Determining the threshold for diminishing returns requires additional experimentation. More experiments are also needed to determine the influence of crossover probability on GA success, because our experiments used probabilities of only 0 and 1. In addition, further experiments could be conducted to compare the effectiveness of uniform crossover against slicing crossover; however, including uniform crossover would entail extending the GA, which currently supports only slicing crossover. Finally, as argued by Baeck (1996), Charbonneau (2002), and DeJong (2007), we found adaptive mutation, when compared to fixed mutation rates, led to substantially better GA performance.

Regarding population size, we observed that 200 individuals led to much better GA success than lower population sizes. This confirms the findings of Digalakis and Margaritis (2001), the researchers who investigated GA control settings over 14 different numerical optimization problems. As suggested by DeJong (2007), we suspect that increasing population size further will lead to diminishing returns, but confirming this (and establishing a threshold) requires additional experimentation.

We also observed that re-randomizing a population too frequently can destroy the ability of a GA to converge to good solutions. Our experiments found that re-randomizing after 40% of the intended number of generations led to better GA success than more frequent re-randomization, but more experiments are needed to determine the effects of higher and intermediate re-randomization thresholds.

Considering selection method, we observed that stochastic uniform sampling (SUS) led to much better GA success than a 2-tournament selection algorithm, regardless of the r value. Overall, we found selection method to be less important to GA success than number of crossover points, mutation rate, re-randomization point, and population size. These findings conflict somewhat with those of Rojas et al. (2002), who studied six test problems and three selection methods (roulette wheel, elitist roulette wheel and deterministic) and reported selection method among the most important GA control parameters. We suspect that SUS is a robust selection method that provides reasonable utility in classic GAs, though further experiments comparing SUS to q-tournament selection ($q > 2$) and other proposed selection methods might prove informative. Conducting such comparisons would require extending the GA, which currently provides only two selection methods: SUS and 2-tournament.

6 Conclusions and Future Work

We defined an experiment design and analysis method to determine the relative importance and most effective setting (among those studied) for each control parameter in a GA. The method is general enough to adapt and apply to determine the same information for a wide variety of search techniques. The method can also be used to determine effective starting parameter values for use in meta-search techniques, such as sequential parameter optimization. We demonstrated the method applied to a classic binary-encoded GA, evaluated against 60 numerical optimization problems, providing findings spanning four times as many problems as any previously reported study of GA control parameters. We confirmed some findings from previous studies in the literature, and we raised questions about others. We found that two exploration control

parameters (i.e., crossover and mutation) most significantly influence GA success, followed by population size and population re-randomization points, while exploitation control parameters (i.e., elitism and selection method) showed less influence. We determined that adapting mutation rate based on population diversity provided better GA success than selecting a fixed mutation rate.

Based on our experiment, we identified the need for further study of various forms of crossover, including crossover probability and uniform crossover. We also outlined future experiments to investigate whether GA success would be improved by re-randomizing a population more frequently. We noted that future experiments might focus on determining the population size at which improvement in GA success begins to level off, as well as comparing additional selection and encoding methods. The experiment design and analysis method we defined can be used to conduct such follow-on studies.

Elsewhere (Mills et al., 2013), we applied the GA described in Section 2 to search for failure and performance-degradation scenarios in a parallel population of cloud-computing simulators. In that application, we used the findings reported in this paper to select settings for six GA control parameters: population size = 200, selection method = stochastic uniform sampling, elite selection percentage = 8%, reboot proportion = 0.4, number of crossover points = 3, and mutation rate = adaptive. (We decided to set precision scaling = 1, since that parameter had least influence on GA success.) While the cloud-computing search problem differed substantially from the numerical optimization problems we used to calibrate the GA control parameters, the parameterized GA proved quite effective at finding failure and performance-degradation scenarios in a cloud-computing simulator.

7 Appendix A: List of Numerical Optimization Problems

We provide four tables, each listing 15 of the 60 numerical optimization problems used in this experiment. For each problem, we give the numerical identifier (ID) we assigned, the function name (and an abbreviation), the number of parameters (Dim) in the problem, the problem type (encoding explained below), the theoretical maximum (or minimum) value (where known), the best value found by the GA, and the source for the problem. The overwhelming majority (56) of the test-set problems are continuous and differentiable; only the three Step functions and the Corana function are discontinuous and non-differentiable. Using the scheme from Jamil and Yang (2013), we further classify the test problems in the type column, using two bit positions. The first bit position denotes separability (separable = 1 and non-separable = 0) and the second bit position denotes modality (uni-modal = 1 and multimodal = 0). A mathematical description for many of these problems can be found at Adorio (2005).

ID	Function	Dim	Type	Max (Min)	GA Best	Source
1	Chemical Yield (Cy)	2	11	unknown	89.7505	Box, Hunter & Hunter 2005
2	Defective Springs (Ds)	3	11	100	99.9982	Box & Bisgard 1990
3	Chemical Reactor (Cr)	5	11	100	100	Box, Hunter & Hunter 2005
4	Morris10 (Mo)	10	11	unknown	112.9689	Saltelli et al. 2004
5	Morris20 (Mo)	20	11	unknown	129.894	Saltelli et al. 2004
6	SchafferF6 (Sf)	2	01	(0)	0	Adorio 2006
7	ShekelM10 (Sh)	4	00	(-10.536284)	-10.53628	Adorio 2006
8	ShekelM5 (Sh)	4	00	(-10.1532)	-10.1532	Adorio 2006
9	ShekelM7 (Sh)	4	00	(-10.403)	-10.40282	Adorio 2006
10	Axis Parallel Hyper Ellipsoid (Ax)	15	11	(0)	9.4e-5	Rahnamayan et al. 2008
11	Axis Parallel Hyper Ellipsoid (Ax)	30	11	(0)	1.21e-3	Rahnamayan et al. 2008
12	Quartic Without Noise (Qn)	100	11	(0)	1.841e-3	Rahnamayan et al. 2008
13	Quartic With Normal Noise (Qn)	100	10	unknown	-3.842	Rahnamayan et al. 2008
14	Quartic With Uniform Noise (Qn)	100	11	(0)	1.67e-1	Rahnamayan et al. 2008
15	Sphere Model (Sp)	15	11	(0)	1.3e-5	Rahnamayan et al. 2008

Table 4: Test problems #1-#15: problem name (and abbreviation), number of dimensions (i.e., problem variables), type (separability and modality), maximum (or minimum) value, the best value discovered by the GA, and the problem source.

ID	Function	Dim	Type	Max (Min)	GA Best	Source
16	Sphere Model (Sp)	30	11	(0)	6.9e-5	Rahnamayan et al. 2008
17	Sphere Model (Sp)	60	11	(0)	3.15e-3	Rahnamayan et al. 2008
18	Beale (Be)	2	01	(0)	0	Adorio 2006
19	Bohachevsky (Bo)	2	00	(0)	0	Adorio 2006
20	Griewank (Gk)	2	00	(0)	0	Adorio 2006
21	Griewank (Gk)	10	00	(0)	1.769e-2	Adorio 2006
22	Michalewitz (Mz)	15	00	(-14.49)	-13.5745	Adorio 2006
23	Michalewitz (Mz)	30	00	(-28.98)	-27.428176	Adorio 2006
24	Michalewitz (Mz)	60	00	(-57.96)	-53.464618	Adorio 2006
25	Plateau (Pl)	5	00	(5)	5	Ingber 1993
26	Step (St)	15	11	(0)	0	Rahnamayan et al. 2008
27	Step (St)	30	11	(0)	0	Rahnamayan et al. 2008
28	Step (St)	60	11	(0)	0	Rahnamayan et al. 2008
29	Paviani (Pa)	10	00	(-45.7784)	-45.7784	Adorio 2006
30	Levy (Le)	15	00	(0)	-1.92e-1	Rahnamayan et al. 2008

Table 5: Test problems #16-#30: problem name (and abbreviation), number of dimensions (i.e., problem variables), type (separability and modality), maximum (or minimum) value, the best value discovered by the GA, and the problem source.

ID	Function	Dim	Type	Max (Min)	GA Best	Source
31	Levy (Le)	30	00	(0)	6.1e-1	Rahnamayan et al. 2008
32	Levy (Le)	60	00	(0)	8.473	Rahnamayan et al. 2008
33	Corana (Co)	4	10	(0)	0	Adorio 2006
34	Hosaki (Hk)	2	00	(-2.3458)	-2.3458	Ali et al. 2005
35	Gear (Gr)	4	00	(2.7e-12)	2.7e-12	Adorio 2006
36	Multimod (Mm)	30	01	(0)	0	Adorio 2006
37	Easom (Ea)	2	10	(-1)	-1	Adorio 2006
38	Camel 3-hump (Cm)	2	00	(0)	0	Ali et al. 2005
39	Camel 6-hump (Cm)	2	00	(-1.0316285)	-1.031628	Ali et al. 2005
40	Hartman (Ha)	3	00	(-3.86)	-3.86	Adorio 2006
41	Hartman (Ha)	6	00	(-3.32)	-3.32	Adorio 2006
42	Schwefel (Sw)	3	10	(0)	0	Adorio 2006
43	Kowalik (Kw)	4	00	(0.00030748610)	5.58e-4	Adorio 2006
44	Watson (Wa)	6	01	(≈2.28767e-3)	2.413e-3	Brent 2002
45	Zettl (Zt)	2	00	(-0.003791)	-0.03791	Adorio 2006

Table 6: Test problems #31-#45: problem name (and abbreviation), number of dimensions (i.e., problem variables), type (separability and modality), maximum (or minimum) value, the best value discovered by the GA, and the problem source.

ID	Function	Dim	Type	Max (Min)	GA Best	Source
46	Ackley (Ak)	10	00	(0)	3.196e-3	Adorio 2006
47	Goldstein-Price (Gp)	2	00	(3)	3	Adorio 2006
48	Rosenbrock (Rk)	15	01	(0)	9.568	Ali et al. 2005
49	Powell (Pw)	4	01	(0)	3.82e-28	Ali et al. 2005
50	Shubert Problem (Sb)	2	10	(-186.7309)	-186.7309	Ali et al. 2005
51	Periodic (Pe)	2	10	(0.9)	0.9	Ali et al. 2005
52	Salomon Problem (Sm)	10	00	(0)	0.099873	Ali et al. 2005
53	Sinusoidal Problem (Sn)	10	10	(-3.5)	-3.499856	Ali et al. 2005
54	Deckkers & Aarts (Dk)	2	10	(-24777)	-24776.518	Ali et al. 2005
55	McCormick (Mk)	2	00	(-1.9133)	-1.913223	Ali et al. 2005
56	Colville (Cv)	4	00	(0)	0	Rahnamayan et al. 2008
57	Zakharov (Zh)	15	00	(0)	0.0748	Rahnamayan et al. 2008
58	Branin's (Br)	2	00	(0.3979)	0.397888	Rahnamayan et al. 2008
59	Perm (Pm)	10	10	(0)	1.2e-5	Rahnamayan et al. 2008
60	Neumaier3 (Nu)	10	00	(-210)	-209.71224	Ali et al. 2005

Table 7: Test problems #46-#60: problem name (and abbreviation), number of dimensions (i.e., problem variables), type (separability and modality), maximum (or minimum) value, the best value discovered by the GA, and the problem source.

8 Appendix B: Comparing Results from a 4^{7-2} Orthogonal Fractional Factorial Experiment Against Results from a Full Factorial Experiment for Three Test Problems

Earlier in this paper, we stated that orthogonal fractional factorial (OFF) experiment designs sample a subset of the parameter combinations from a full factorial (FF) experiment. We also stated that OFF designs give estimates for main effects, where estimate accuracy is influenced by the number of samples (n), the number of levels (ν), and the variance (σ) in the underlying observations. Main-effect estimates directly determine the relative rank among factors, as well as the best level for each factor, thus the accuracy of such estimates will influence findings and conclusions from OFF experiments. Further, uncertainty in main-effect estimates generated by OFF experiments also influence the computation of statistical significance using ANOVA. This latter point stands to reason, because ANOVA compares variation within groups of data points against variation among groups. Sampling methods typically exhibit increased variance over measurements taken over an entire population. This variance increase directly influences the computation of ANOVA statistics. In this appendix, we provide some data, from our experiments, regarding the effects of uncertainty arising when applying a 4^{7-2} OFF experiment to sample from a 4^7 FF experiment.

We selected three of the 60 test problems on which to compare results from the OFF design against results from a FF experiment. We chose a problem from each of the complexity categories among our numerical optimization problems: (1) problem #3 – Chemical Reactor (Cr: 5 dimensions), (2) problem #11 – Axis Parallel Hyper Ellipsoid (Ax: 30 dimensions), and (3) problem #24 – Michalewitz (Mz: 60 dimensions). For each factor of each problem, we compared the estimated effect (E_i)–including the derived factor ranking and most-effective level–and the ANOVA statistic (F_{cdf_i}). Our comparison demonstrates that: (1) OFF experiment designs do indeed produce estimates, (2) the main-effect estimates are fairly accurate, as are the derived factor rankings and best levels, and (3) the F_{cdf_i} estimates are somewhat less accurate, because for statistically significant factors in a FF experiment the F_{cdf_i} measures converge toward 100 due to decreased variance among the data points used to compute main effects. As we will demonstrate, this decrease in variance leads to increased statistical significance, even for small differences in main effects.

Table 8 compares results for main effects estimates (E_i), F_{cdf_i}, factor rank, and best level for the Chemical Reactor problem when using a 4^{7-2} OFF design and a FF experiment. Tables 9 and 10 provide similar results for two other problems: Axis Parallel Hyper Ellipsoid and Michalewitz.

Table 8 shows that most main-effect estimates from the OFF experiment are fairly close to the FF results for the Chemical Reactor problem. An exception is the estimate for factor x_4 (reboot proportion), where the OFF design indicates a substantially larger effect than the FF experiment. This difference also leads to some differences in factor rankings, which are derived from relative differences in main effects. The top two factors (x_5 and x_6) remain the same, but the difference in main effects estimate for x_4 leads to reordering in the ranking of three factors. The OFF and FF experiments produced the same best level setting for all but one factor, x_2 (selection method). Finally, in the FF experiment, the F_{cdf_i} values are at or near 100% for all factors, while factors x_2 and x_3 are nearer 50% in the OFF experiment. We explore these differences more thoroughly

Problem #3 Chemical Reactor (Cr) with 5 Dimensions

	E_i		F_{cdfi}		Rank		Best Level	
Factor	OFF	FF	OFF	FF	OFF	FF	OFF	FF
x_1	0.04	0.03	99.99	100	5	3	4	4
x_2	0.01	0.01	56.29	100	6	6	1	2
x_3	0.01	0.02	53.68	100	7	5	1	1
x_4	0.06	0.01	100	99.86	3	7	4	4
x_5	0.07	0.05	100	100	2	2	4	4
x_6	0.07	0.06	100	100	1	1	1	1
x_7	0.05	0.03	100	100	4	4	4	4

Table 8: Problem #3 – Chemical Reactor (Cr: 5 dimensions) comparison for each factor (x_1 through x_7) of main effects (E_i), statistical significance (F_{cdf_i}), rank, and best level from a 4^{7-2} OFF experiment against a 4^7 FF experiment.

later in this appendix.

Table 9 shows that most main-effect estimates from the OFF experiment are close to the FF results for the Axis Parallel Hyper Ellipsoid problem. Results from the OFF experiment report somewhat higher estimated effects for two factors: mutation rate (x_6) and precision scaling (x_7). Small differences in main-effect estimates lead to factors x_4 and x_6 swapping ranks 3 and 4. The OFF and FF experiments produced the same best level setting for all but one factor, population size (x_1), where the OFF experiment reported the size of 150 to be best, while the FF experiment found 200 to be best. Most of the F_{cdf_i} values were near 100%, except for x_1 and x_7. In the case of x_1, the variance in the data from the OFF design was sufficient to lower F_{cdf_i}. In the case of x_7, the estimated effect for the FF experiment was sufficiently low that the F_{cdf_i} rightly reflects a lack of statistical significance.

Table 10 shows that most main-effect estimates from the OFF experiment are quite close to the FF results for the Michalewitz problem. In fact, the OFF and FF experiments produced identical rankings and best levels for each of the seven factors. As typically the case, the FF experiment reported F_{cdf_i} values at 100%, while the variance in the data acquired from the OFF experiment design led to lower F_{cdf_i} values for two factors: x_4 and x_7.

The reader should bear in mind that the comparisons we discussed above consider only three of the 60 test problems. The overall analysis averages data from all 60 test problems. Such averaging tends to allow minor estimate errors in individual problems to offset each other (unless there is some hidden bias in the procedure). This is another justification for ensuring that EA algorithms are evaluated on a sufficiently large sample of search problems.

To better understand the reasons underlying differences in main-effect estimates between an OFF and FF experiment, we delve more deeply into the Chemical Reactor problem (#3). Recall that uncertainty in main-effect estimates is a function of the

Problem #11 Axis Parallel Hyper Ellipsoid (Ax) with 30 Dimensions

	E_i		F_{cdf_i}		Rank		Best Level	
Factor	OFF	FF	OFF	FF	OFF	FF	OFF	FF
x_1	34.30	37.11	77.91	100	6	6	3	4
x_2	152.26	147.25	100	100	2	2	1	1
x_3	315.59	319.37	100	100	1	1	4	4
x_4	82.49	90.05	100	100	4	3	4	4
x_5	62.12	65.17	99.78	100	5	5	4	4
x_6	90.77	69.50	100	100	3	4	1	1
x_7	20.48	2.59	26.56	5.11	7	7	1	1

Table 9: Problem #11 – Axis Parallel Hyper Ellipsoid (Ax: 30 dimensions) comparison for each factor (x_1 through x_7) of main effects (E_i), statistical significance (F_{cdf_i}), rank, and best level from a 4^{7-2} OFF experiment against a 4^7 FF experiment.

number of samples (n), the number of levels (ν), and the variance (σ) in the underlying observations. For our experiments, n and ν are fixed, which yields an uncertainty estimate of 0.088 σ. This implies that uncertainty in main-effect estimates will be driven by σ, which is unknown for our experiment. Though σ is unknown, the confidence intervals around the main-effect estimates for our experiments may give some indication of the relative nature of σ.

Figure 7 shows a factor-analysis plot from the OFF experiment for the Chemical Reactor problem. The plot contains the same type of information we described when discussing Figure 1, but here we added vertical bars through the estimated mean $\overline{y}_{ij}$ for each level (ℓ_j) of each factor (x_i). The vertical bars indicate the 95% confidence interval, which implies that there is a small chance that the true mean falls outside the interval. The larger the interval, the greater the uncertainty in the data used to make the estimate. Figure 8 shows a factor-analysis plot (including 95% confidence intervals) from the FF experiment for the Chemical Reactor problem.

Recall Table 8, which shows that the main-effect estimates produced by the OFF experiment varied somewhat from the results of the FF experiment. For example, the estimated effect found by the OFF experiment for factor x_4 was substantially larger than the effect shown by the FF results. The large variance associated with the OFF data for that factor indicates the main-effect estimate would tend to be more uncertain. Comparing Figures 7 and 8, and their associated confidence intervals, reveals why the main-effect estimates from the OFF experiment differed from the FF results. For example, the mean value for x_4 level 3 was much lower when estimated from the data sampled in the OFF experiment than was the case when computed from the FF data. Further, mean values from the FF results exhibited much less uncertainty.

The uncertainty in main-effect estimates produced by an OFF design cannot be eliminated by simply iterating the OFF design, while varying the random number seed so as to generate independent samples that can be averaged. To demonstrate this, we iterated the 4^{7-2} OFF experiment 16 times for the Chemical Reactor problem,

Problem #24 Michalewitz (Mz) with 60 Dimensions

	E_i		F_{cdf_i}		Rank		Best Level	
Factor	OFF	FF	OFF	FF	OFF	FF	OFF	FF
x_1	5.19	5.18	100	100	5	5	4	4
x_2	7.41	7.23	100	100	3	3	1	1
x_3	14.39	14.46	100	100	1	1	4	4
x_4	1.54	2.70	79.97	100	6	6	4	4
x_5	5.24	5.35	100	100	4	4	4	4
x_6	9.98	9.83	100	100	2	2	2	2
x_7	0.93	1.10	39.58	100	7	7	4	4

Table 10: Problem #24 – Michalewitz (Mz: 60 dimensions) comparison for each factor (x_1 through x_7) of main effects (E_i), statistical significance (F_{cdf_i}), rank, and best level from a 4^{7-2} OFF experiment against a 4^7 FF experiment.

producing 16,384 datasets, which is equivalent in number to the 16,384 datasets produced by the FF experiment. Subsequently, for each factor, we averaged the main-effect estimates, F_{cdf_i} values, and ranks across the 16 OFF experiment iterations, and we took the mode of the best levels from the 16 iterations. We then compared (see Table 11) these results against the FF results.

As Table 11 shows, the main-effect estimates and ranks for each factor taken from the averaged OFF experiment results differ somewhat from the FF results. Similarly, though there is largely agreement in the best levels for each factor, the best level for factor x_4 differs. This difference is of no consequence because, as we discuss in the main body of the paper, levels 1 and 4 for factor x_4 yield comparable effectiveness across the 60 test problems.

Figure 9 provides an overview of changes in factor rankings across the 16 iterations of the OFF design for the Chemical Reactor problem. Mutation rate (x_6) is consistently top ranked on all iterations, while selection method (x_2) and elite selection percentage (x_3) are consistently among the lowest ranked. The ranks of other factors fluctuate across the iterations, as dictated by relative changes in main-effect estimates. This example provides some insight into how results from OFF experiments can lead to differences in factor ranking, when compared with FF results. Further, the example suggests that OFF designs provide confident estimates of the most and least influential factors for any specific problem, while introducing uncertainty about the true ordering of factors exhibiting intermediate influence. Analyzing factor rankings across a large set of problems allows uncertainty on individual problems to be averaged, leading to reasonably confident factor rankings.

The main conclusions from this limited comparison of results from a 4^{7-2} OFF design against FF results are: (1) OFF experiment designs do indeed produce estimates, (2) the main-effect estimates are fairly accurate, as are the derived factor rankings and best levels, and (3) the F_{cdf_i} estimates are somewhat less accurate, because in a FF experiment F_{cdf_i} measures converge toward 100 due to decreased variance among the

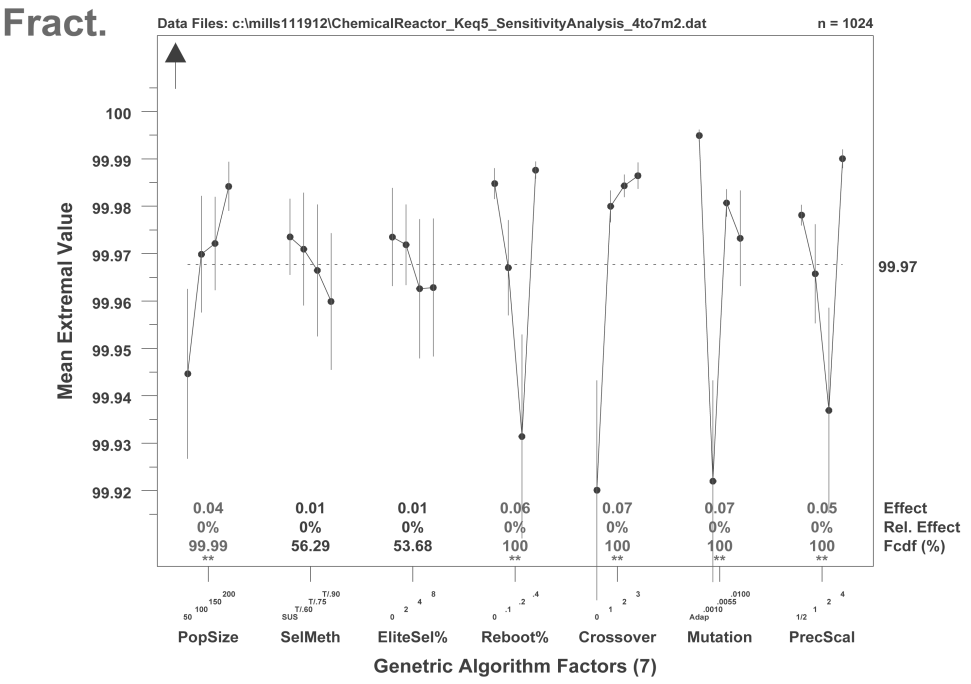

Figure 7: Factor-analysis plot from 4^{7-2} OFF experiment for numerical optimization problem #3: maximizing the output of a chemical reactor process. The 95% confidence intervals are given by vertical bars through each estimated mean $\overline{y}_{ij}$ for each level (ℓ_j) of each factor (x_i).

data points used to compute main effects. We also demonstrated that OFF designs produce confident rankings of the most and least influential factors for individual test problems. Confidence in the rankings of factors with intermediate influence relies on averaging results across a large collection of varied test problems.

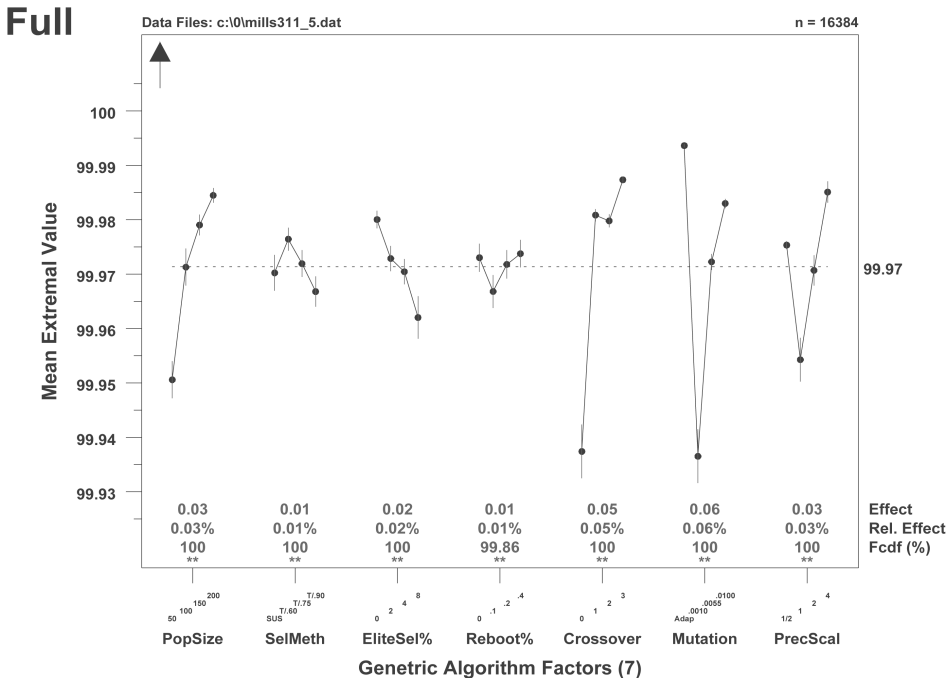

Figure 8: Factor-analysis plot from 4^7 FF experiment for numerical optimization problem #3: maximizing the output of a chemical reactor process. The 95% confidence intervals are given by vertical bars through each estimated mean $\overline{y}_{ij}$ for each level (ℓ_j) of each factor (x_i).

Problem #3 Chemical Reactor (Cr) with 5 Dimensions

	E_i		F_{cdf_i}		Rank		Best Level	
Factor	Average 16 OFFs	FF	Average 16 OFFs	FF	Average 16 OFFs	FF	Mode 16 OFFs	FF
x_1	0.05	0.03	100	100	3	3	4	4
x_2	0.02	0.01	80.92	100	6	6	2	2
x_3	0.02	0.02	88.83	100	6	5	1	1
x_4	0.04	0.01	99.95	99.86	4	7	1	4
x_5	0.04	0.05	99.31	100	4	2	4	4
x_6	0.07	0.06	100	100	1	1	1	1
x_7	0.05	0.03	99.98	100	3	4	4	4

Table 11: Problem #3 – Chemical Reactor (Cr: 5 dimensions) comparison for each factor (x_1 through x_7) of main effects (E_i), statistical significance (F_{cdf_i}), and rank averaged across 16 iterations of a 4^{7-2} OFF experiment against a 4^7 FF experiment. The best level for each factor of the 4^{7-2} OFF experiment was determined by taking the mode from across the 16 iterations.

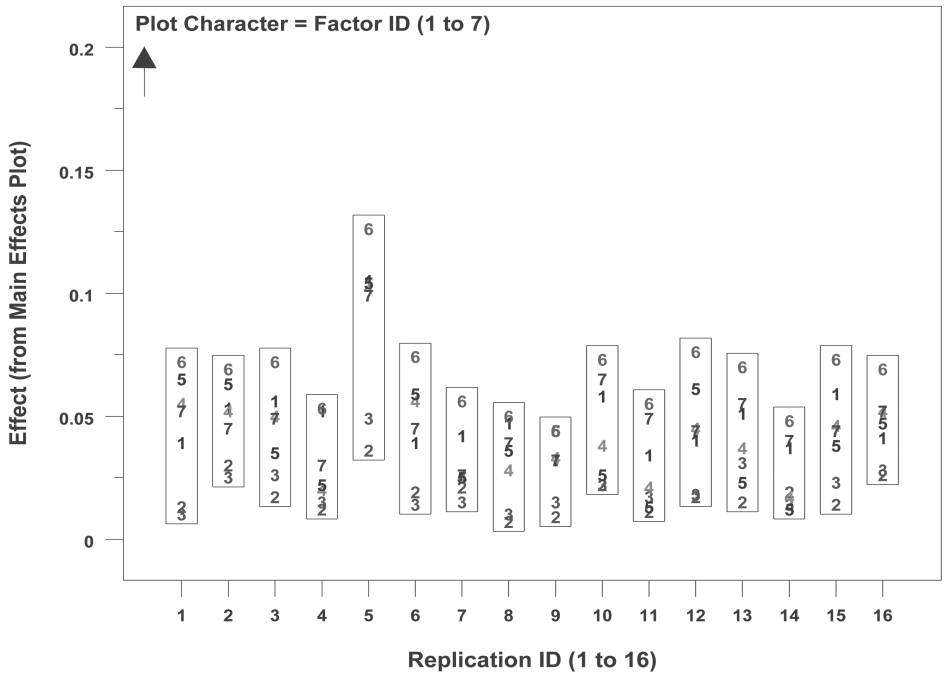

Figure 9: Main-effect estimates (y axis) for the seven genetic algorithm control parameters on 16 iterations (x axis) of Problem #3 – Chemical Reactor (Cr: 5 dimensions). For each iteration, the parameter identifiers (1 for x_1 to 7 for x_7) are sorted from largest (top) to smallest (bottom) main effect (E_i).

References

Adorio, E. P. (2005). A list of 58 numerical optimization test functions, along with associated equations and source code in c. `http://www.geocities.ws/eadorio/mvf.pdf/`.

Ali, M. M., Khompatraporn, C., and Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31:635–672.

Arenas, M. I. G., Valdivieso, P. A. C., Garcia, A. M. M., Guervos, J. J. M., Laredo, J. L. J., and Garcia-Sanchez, P. (2010). Three interconnected parameters for genetic algorithms. In et al., R. S., editor, *PPSN XI, Part II, LNCS 6239*, pages 452–461. Springer-Verlag.

Baeck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford University Press.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the 2nd International Conference on Genetic Algorithms and thier Applications*, pages 14–21. L. Erlbaum Associates.

Bartz-Beielstein, T. (2006). *Experimental Research in Evolutionary Computation.* Springer-Verlag.

Bartz-Beielstein, T., Lasarczyk, C. W. G., and Preuss, M. (2005). Sequential parameter optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 773–780 Vol.1.

Box, G. and Bisgaard, S. (1996). *Designing Industrial Experiments.* BBBF Books.

Box, G., Hunter, J., and Hunter, W. (2005). *Statistics for Experimenters 2nd edition.* John Wiley and Sons.

Boyabatli, O. and Sabuncuoglu, I. (2004). Parameter selection in genetic algorithms. *Systemics, Cybernetics and Informatics*, 2(4):78–82.

Brent, R. P. (2002). *Algorithms for Minimization Without Derivatives.* Dover Publications.

Charbonneau, P. (2002). An introduction to genetic algorithms for numerical optimization. *NCAR Technical Note*, page 74.

Cioppa, T. M. and Lucas, T. W. (2007). Efficient nearly orthogonal and space-filling latin hypercubes. *Technometrics*, 49(1):45–55.

DeJong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* PhD dissertation University of Michigan.

DeJong, K. A. (1999). Genetic algorithms: a 30 year perspective.

DeJong, K. A. (2007). Parameter setting in eas: a 30 year perspective. In Kacprzyk, J., editor, *Studies in Computational Intelligence*, volume 54, pages 1–18. Springer.

Diaz-Gomez, P. and Hougen, D. F. (2009). Three interconnected parameters for genetic algorithms. In Rothlauf, F., editor, *Proceedings of the 11th Conference on Genetic and Evolutionary Computation*, pages 763–770. ACM.

Digalakis, J. G. and Margaritis, K. G. (2001). On benchmarking functions for genetic algorithms. *International Journal of Computer Mathematics*, 77(4):481–506.

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

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathl. Comput. Modelling*, 18:29–57.

Jamil, M. and Yang, X.-S. (2013). A literature survey of benchmark functions for global optimization problems. *Intl. Journal of Mathematical Modelling and Numerical Optimization*, 4(2):150–194.

Janikow, C. and Michalewicz, Z. (1991). An experimental comparison of binary and floating point representations in genetic algorithms. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 151–157. Morgan Kaufmann.

Kapoor, V., Dey, S., and Khuran, A. P. (2011). An empirical study of the role of control parameters of genetic algorithms in function optimization problems. *International Journal of Computer Applications*, 31(6):20–26.

Kleijnen, J. P. C. (2010). Design and analysis of computational experiments: overview. In T. Bartz-Beielstein, M. Chiarandini, L. P. and Preuss, M., editors, *Experimental Methods for the Analysis of Optimization Algorithms*, chapter 3, pages 51–72. Springer-Verlag.

Mills, K., Dabrowski, C., Filliben, J., and Ressler, S. (2013). Combining genetic algorithms and simulation to search for failure scenarios in system models. In *Proceedings of the 5th International Conference on Advances in System Simulation*, pages 1–8. IARIA.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. The MIT Press.

Nunez-Letamendia, L. (2007). Fitting the control parameters of a genetic algorithm: an application to technical trading systems design. *European Journal of Operational Research*, 179(3):847–868.

Odetayo, M. O. (1997). Empirical study of the interdependencies of genetic algorithm parameters. In *Proceedings of the 23rd EUROMICRO Conference: New Frontiers of Information Technology*, pages 639–643. IEEE.

Rahnamayan, S., Tizhoosh, H., and Salama, M. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, 12(1):64–79.

Rojas, I., Pomares, H., Gonzalez, J., Merelo, J., Castillo, P., and Romero, G. (2002). Statistical analysis of the main parameters involved in the design of a genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 32(1):31–37.

Sahin, I. (2011). Random lines: A novel population set-based evolutionary global optimization algorithm. In *Genetic Programming Lecture Notes in Computer Science*, volume 6621, pages 97–107. Springer.

Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. (2004). *Sensitivity Analysis in Practice*. John Wiley and Sons.

Schaffer, J., Carruana, R., Eshelman, L., and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 51–60. Morgan Kaufmann.

Spears, W. M. and DeJong, K. A. (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26.

Taleb, N. (2010). *The Black Swan 2nd edition*. Random House.

Tate, D. M. and Smith, A. E. (1993). Expected allele coverage and the role of mutation in genetic algorithms. In Forrest, S., editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 31–37. Morgan Kaufmann.