Balancing the Effects of Parameter Settings on a Genetic Algorithm for Multiple Fault Diagnosis

Robert F. Chevalier

Artificial Intelligence Center
The University of Georgia

Balancing the Effects of Parameter Settings on a Genetic Algorithm for Multiple Fault Diagnosis

Robert F. Chevalier

Artificial Intelligence Center
The University of Georgia

Abstract

Genetic algorithms can be a very successful means of solving Multiple Fault Diagnosis problems. This success, however, depends on finding the right balance between exploring the search space and exploiting discovered individuals. We reveal the results of several experiments to find the most successful combinations of parameter settings and discuss the effects of varying individual parameters in light of their contribution to the genetic algorithm's overall balance of exploration and exploitation. We also show a method for predicting the degree of effectiveness of certain combinations of parameter settings.

1 Introduction

A Genetic algorithm (GA) makes use of evolutionary techniques observed in biology to find solutions to complex optimization and search problems. The tools of a GA are selection, crossover and mutation. Like its biological counterpart, a GA uses these operators to explore the search space while exploiting the successful solutions that it encounters. There are many variations of the standard GA operators, most of which can be applied in varying degrees. Each operator contributes in some way to either the amount that a GA will explore the search space or the degree to which it will exploit individuals.

We will examine the impact of specific operators on a Multiple Fault Diagnosis (MFD) problem. By varying parameter settings we will observe which different operators work best together and at what particular values. It will be shown that the best results can be obtained by maintaining a balance between operators that promote exploration and operators that promote exploitation. The results will be judged both on their reliability and the speed with which they are able to find a solution. We'll start with the basic operators of a simple GA and then introduce some more sophisticated operators. We will also measure our results against the Mixing Model [Thie96] and describe a method that makes use of this model to predict the success of specific combinations of parameter settings.

2 Multiple Fault Diagnosis

Multiple Fault Diagnosis is the process of determining the most likely cause of one or more observed symptoms. This cause, or diagnosis, may be the result of several faults, all occurring at the same time. The MFD problem which we will be looking at involves 10 possible symptoms and 25 possible faults. Since any combination of these 25 faults may be the cause of the observed symptom set, there are 2^{25} (33,554,432) possible diagnoses. Obviously an exhaustive search is out of the question because of this complexity. Instead, we make use of a simple GA to find solutions.

Before the GA can be set loose on the search space, it needs to be given a method for comparing possible solutions that it will encounter. The method that we employed is a modified version of the *relative likelihood* function [Peng87a, Peng87b]. The *relative likelihood* function defines the finite set of possible disorders as D and the finite set of possible symptoms as M. It then makes use of a tendency matrix which defines the likelihood (c_{ij}) that a particular diagnosis (d_j) causes a particular symptom (m_i), given diagnosis (d_j). Additionally, it makes use of a vector of prior probabilities (p_j) for each fault. This data may be derived from historical trends or an analysis of the relationship between the individual faults and symptoms performed by domain experts. Given this information, the likelihood that a diagnosis (DI), which is a subset of D, causes a particular symptom set (M^+), which is a subset of M, is defined as the product of three terms. These terms are

the likelihood that DI covers the given M^+ (L1), the likelihood that DI does not cover more than the given M^+ (L2), and the likelihood that a common disorder in DI is the cause of M^+. These terms are defined as follows:

$$L1 = \prod_{m_i \in M+} \left(1 - \prod_{d_j \in DI} (1 - c_{ij})\right)$$

$$L2 = \prod_{dj \in DI} \prod_{m_i \in effects(d_j) - M+} (1 - c_{ij})$$

$$L3 = \prod_{d_j \in DI} \frac{p_j}{(1 - p_j)}$$

A drawback of this *relative likelihood* function is that L1 will be forced to zero in cases where the DI does not cover M^+ and L2 will be forced to 0 in cases where the DI covers more than the observed M^+. This limitation is overcome using a *modified relative likelihood* [Pott90]. The *modified relative likelihood* function changes any $c_{ij} = 0$ to a number close to but not equal to 0. Likewise, it changes any $c_{ij} = 1$, to a number close to but not equal to 1. This change allows the GA to get some sense of the fitness of a diagnosis that otherwise would have been given a fitness of 0.

3 Experiment Setup

There were three phases of our experiment. Our first step was to establish a baseline by selecting an initial group of parameter settings to run on our GA. Kenneth De Jong determined in his 1975 PhD dissertation [DeJo75] that generally good results could be found on a wide range of problems by making use of a population size of about 100, a crossover rate of 0.60, and a mutation rate of 0.001. We decided to build off of these findings. As a result, our baseline parameter settings were defined as follows:

Population Size (n) = 100
Crossover Rate (p_c) = 0.6
Mutation Rate (p_m) = 0.001
Selection Method = roulette wheel
Elitism = true
Stopping Criteria = 10 stable generations

The goal of our next set of experiments was to find the optimal standard parameter settings. To do this, we ran the GA using every possible combination of the following settings:

n = 40 – 180 in increments of 20
p_c = .4 - .8 in increments of .05
p_m = .001 - .1 in increments of .011
Selection Method = roulette wheel and tournament[1]
Elitism = true and false
Stopping Criteria = 5 – 25 stable generations in increments of 5

Finally, we altered the simple GA so that it was capable of using uniform crossover, rank based selection and a modified mutation scheme. We then ran the GA substituting in these non-standard variants.

[1] A tournament size of 2 was used for these initial standard variant experiments.

In all three phases, we used the following method to test each combination of parameter settings: we ran the GA against all 1023 possible symptom sets, compared the GA's solution for each run against the pre-established "best" solution, and calculated the reliability based on the number of matching solutions. This process was repeated five times and the highest of the five reliabilities was recorded as the reliability for that particular combination of parameter settings. Also recorded was the number of fitness evaluations run. This figure was then used to determine the speed with which the GA was able to find a solution.

The GA that we used to run our experiments was written in C#. Although the GA was written from scratch, its logic was based loosely on the pseudo-code found in [Gold89, Enge02].

4 Efficiency Value

Each combination of parameter settings in our experiments needed to be judged based not only on its ability to produce correct answers (its reliability) but also on speed (the number of required fitness evaluations). In order to compare various parameter combinations objectively, we would need to define a third criteria which would be a concrete representation of both the reliability and speed. In a similar manner to that used in [Kolj06] to develop a "cost function", we defined the following efficiency value which is the result of a linear combination of the reliability and the number of fitness evaluations:

$$efficiency = reliability - weight\left(\frac{fitness\,evaluations}{100{,}000{,}000}\right)$$

We divide the number of fitness evaluations by 100,000,000 in order to normalize the value and then multiply this number by a constant weight value before subtracting it from the reliability. The weight value is used to assign a relative importance to speed with regard to reliability. Determining an appropriate value for this weight required a quick analysis of the experimental results. Keeping in mind Goldberg's [Gold85] expression for optimal population size[2], we calculated that for our MFD problem the optimal population size should be approximately 60 individuals. We then adjusted the weight value so that the average efficiency for results with a population size of 60 was higher than the average of any other population size. Our final formula for calculating efficiency is as follows:

$$efficiency = reliability - 3\left(\frac{fitness\,evaluations}{100{,}000{,}000}\right)$$

5 Results

The best result from our 5 GA runs using the baseline settings obtained a reliability of 0.1925708 and it required 2,142,306 fitness evaluations to complete. Although the baseline settings produced results relatively quickly, as evidenced by the low number of fitness evaluations, the reliability was unacceptably low and as a result the efficiency value of the baseline was only 0.1283016.

The standard variant parameter settings produced efficiency values in the range of -1.3553242 to 0.7252409. The fact that the baseline efficiency falls in the upper half of this range is somewhat misleading. Although some of the standard variants did indeed fare worse than the baseline, the majority outperformed it as can be seen by observing the histogram data in Figure 1.

[2] Goldberg derived the following expression for optimal population size given chromosomes of length, l: $pop = 1.65 \cdot 2^{0.21(l)}$

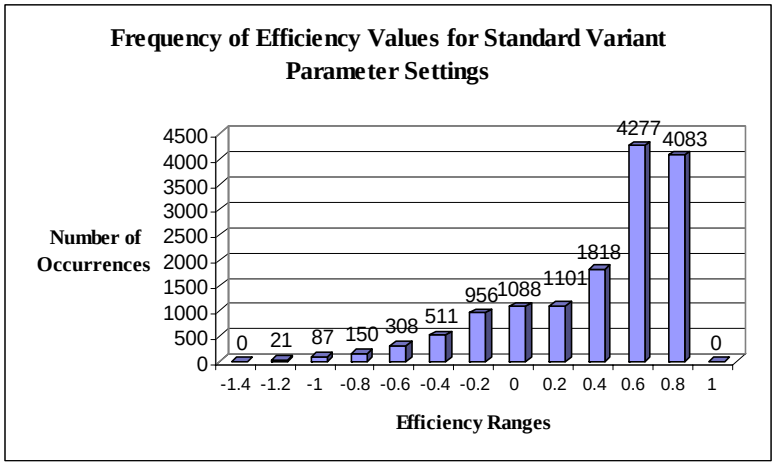

Figure 1. The baseline efficiency value falls in the range of 0 - .2 which means that at least 10,178 of the 14,400 standard variant parameter settings outperformed it.

5.1 Optimal Settings

Before observing the optimal standard variant settings in terms of efficiency, we first examine the top performers based solely on speed and then based solely on reliability. This should give us a better understanding of their relationship to efficiency. Table 1 shows the top performers in terms of speed.

TABLE 1. Top Ten Standard Variant Parameter Settings in Terms of Fitness Evaluations.

n	p_c	p_m	Selection Method	Elitism	Stable Generations	# Fitness Evaluations	Reliability	Efficiency
40	.40	.001	Roulette	False	5	391,559	0.0068426	-0.0049041
40	.40	.001	Roulette	True	5	395,824	0.0048875	-0.0069871
40	.45	.001	Roulette	False	5	409,610	0.0068426	-0.0054456
40	.45	.001	Roulette	True	5	412,413	0.0078201	-0.0045522
40	.50	.001	Roulette	False	5	427,114	0.0087976	-0.0040157
40	.50	.001	Roulette	True	5	433,016	0.0048875	-0.0081028
40	.55	.001	Roulette	True	5	445,951	0.0068426	-0.0065359
40	.55	.001	Roulette	False	5	447,098	0.0107526	-0.0026602
40	.60	.001	Roulette	False	5	461,045	0.0087976	-0.0050336
40	.60	.001	Roulette	True	5	469,106	0.0078201	-0.0067352

It can be seen that the entries in this table required very few fitness evaluations to find a solution. However, this was achieved at the expense of reliability. This poor reliability is reflected in the overall efficiency for each entry. Conversely in Table 2, the top performers in terms of reliability all have poor fitness evaluation numbers.

TABLE 2. Top Ten Standard Variant Parameter Settings in Terms of Reliability.

n	p_c	p_m	Selection Method	Elitism	Stable Generations	# Fitness Evaluations	Reliability	Efficiency
180	.70	.023	Tournament	True	15	21,281,474	1	0.3615557
160	.75	.023	Tournament	True	20	22,857,297	1	0.3142810
160	.55	.023	Tournament	True	25	23,627,017	1	0.2911894
160	.80	.023	Tournament	False	20	24,208,834	1	0.2737349
180	.65	.023	Tournament	True	20	24,270,213	1	0.2718936
180	.80	.012	Tournament	False	25	24,421,147	1	0.2673655
180	.60	.023	Tournament	False	20	24,621,372	1	0.2613588
180	.70	.023	Tournament	True	20	24,877,019	1	0.2536894
180	.75	.023	Tournament	True	20	25,492,166	1	0.2352350
160	.65	.023	Tournament	False	25	25,766,469	1	0.2270059

Notice that the effect on the overall efficiency is not quite as pronounced as it was in Table 1. This is because reliability is weighted more heavily than speed in the efficiency formula. Finally, Table 3 displays the top performers in terms of overall efficiency.

TABLE 3. Top Ten Standard Variant Parameter Settings in Terms of Efficiency.

n	p_c	p_m	Selection Method	Elitism	Stable Generations	# Fitness Evaluations	Reliability	Efficiency
40	.80	.067	Roulette	False	20	5,997,998	0.9051808	0.7252409
40	.75	.056	Roulette	True	25	6,526,881	0.9110459	0.7152395
40	.75	.056	Roulette	False	20	5,659,136	0.8836754	0.7139013
40	.75	.067	Roulette	False	25	7,029,516	0.9247311	0.7138457
60	.55	.056	Roulette	True	15	6,459,725	0.9042033	0.7104115
40	.80	.056	Roulette	False	20	5,685,802	0.8807429	0.7101688
60	.60	.045	Roulette	True	15	6,050,543	0.8914956	0.7099793
40	.80	.067	Roulette	False	15	4,944,236	0.8582600	0.7099329
40	.80	.045	Roulette	True	25	6,117,710	0.8934506	0.7099193
40	.70	.023	Tournament	True	20	6,192,994	0.8954056	0.7096158

It should be noted that the entries in this table have many common parameter settings. They all have relatively low population sizes, but high crossover and mutation rates. The one exception to this rule is the last entry which has a mutation rate of only 0.23. This entry, however, uses Tournament selection which compensates for a low mutation rate because it is not as susceptible to premature convergence as roulette wheel selection. Consequently, more of the search space is able to be explored, even with a small mutation rate. This complimentary relationship between parameters can be found in almost all successful GA configurations.

5.2 Effects of Mutation Rate

Since mutation is an exploratory operator, we know that raising the mutation rate contributes to the amount of the search space that the GA attempts to examine. We may then conclude that the higher this rate is, the more likely a GA is to discover the global optimum. This is true to a point; however, as we can see in Figure 2, the efficiency actually starts to drop as the mutation rate reaches its highest values.

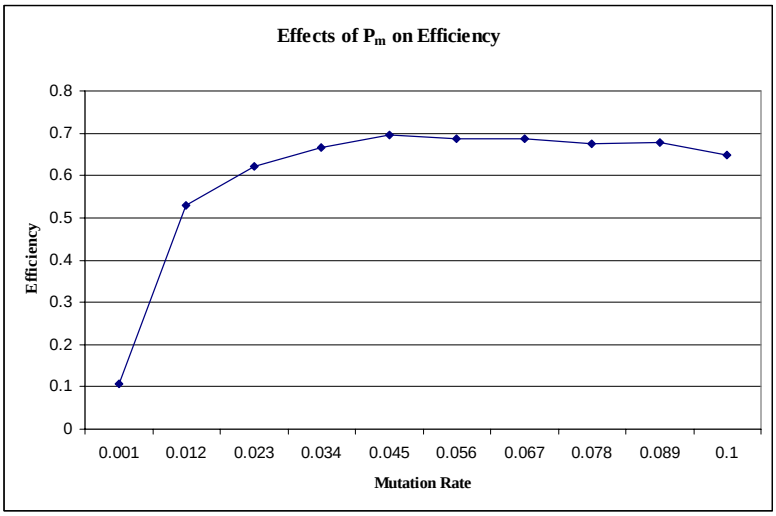

Figure 2. The above data was obtained by using the following parameter settings: n = 40, p_c = .6, p_m = .001 - .1, selection = roulette wheel, stable generations = 15, elitism = false.

Altering the mutation rate effects the balance of parameter settings, initially providing a better balance, but then ultimately becoming more disruptive until the exploitive effects of selection are completely overwhelmed [Eshe89]. The mutation rate is not the only operator contributing to this balance of exploration and exploitation. For instance, if we alter population size and then range over the same set of mutation rates, we

would expect different results. This affect is demonstrated in Figure 3, which compares the above results with those obtained at a higher population size.

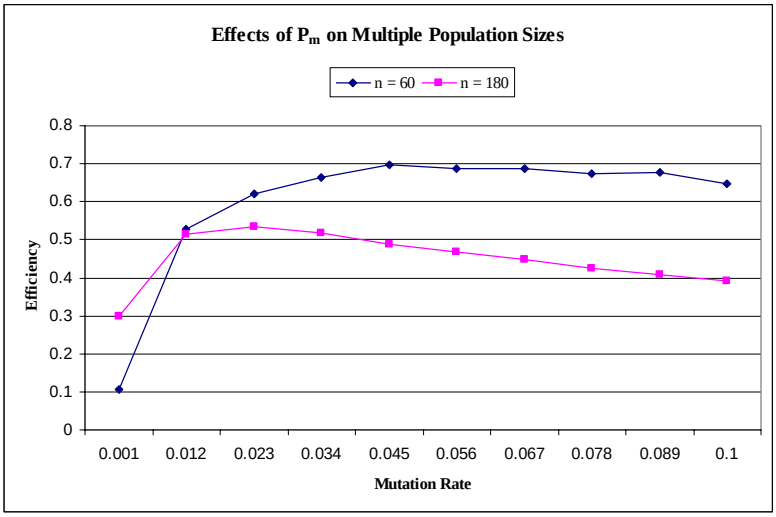

Figure 3. At a lower population size, good performance is more sensitive to changes in the mutation rate.

Notice that with a larger population size, the sensitivity to mutation rate has decreased. At a population of 60, the range of efficiency values is from 0.1 to 0.7, however, with a population of 180, the efficiency values only range from 0.3 to around 0.5. As noted in [Scha89], the inverse relation between population and mutation reflects the fact that increasing either one increases exploration; they may be traded off while maintaining a more or less constant level of exploration.

5.3 Roulette Wheel Selection vs. Tournament Selection

The fundamental difference between selection methods can be observed in their varying degrees of selection pressure. Selection pressure is the intensity with which a GA tends to either eliminate an individual or give it an adaptive advantage. Generally speaking roulette wheel has a higher selection pressure than that of tournament. This is due to its inherent bias towards the most fit individuals in a population. Whereas, roulette wheel selection depends on an individual's relative fitness, tournament selection depends on an individual's rank and is not affected by the fitness distribution [Khar05]. With a higher selection pressure roulette wheel exerts greater exploitive power, and consequently less exploratory power, than that of tournament selection. As a result, their tolerance for shifts in the balance of exploration and exploitation are different. Figure 4 shows that as the mutation rate gets higher, the efficiency initially increases for both selection methods, with tournament selection outperforming roulette wheel. As the mutation rate continues to rise, however, the balance shifts in favor of roulette wheel selection. At this point, tournament's efficiency dramatically drops off while the efficiency of roulette wheel continues to increase a bit longer before gradually declining.

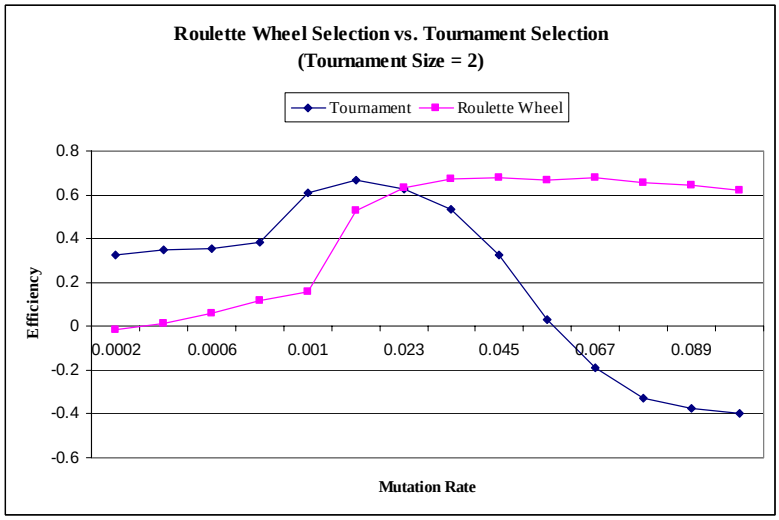

Figure 4. With all other parameter settings equal, Tournament selection performs better at lower mutation rates (assuming a small tournament size) and Roulette Wheel selection performs better at higher mutation rates. The other relevant parameter settings are: n = 80, p_c = .6, p_m = .0002 - .1, stable generations = 15, elitism = false.

Since tournament's selection pressure is directly related to its tournament size, it is possible to manipulate the tournament selection operator and force it to behave more like roulette wheel selection. Notice how the tournament selection curve begins to resemble the roulette wheel curve as we progressively employ higher tournament sizes in Figures 5, 6, and 7.

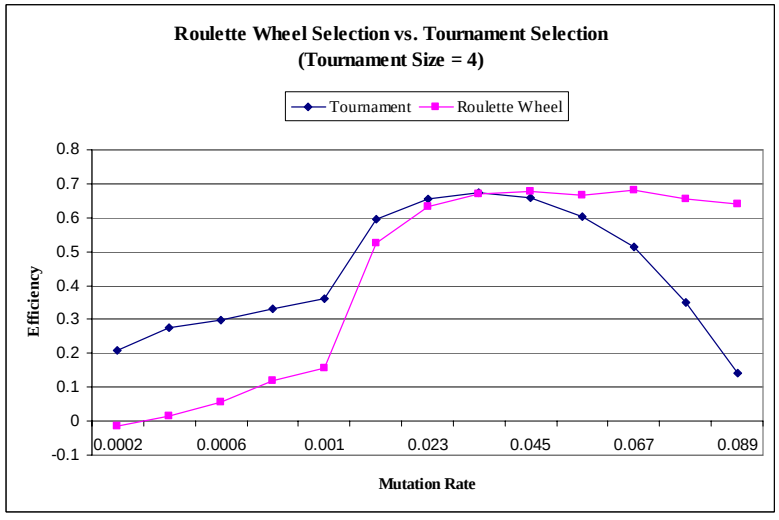

Figure 5. n = 80, p_c = .6, p_m = .0002 - .1, stable generations = 15, elitism = false.

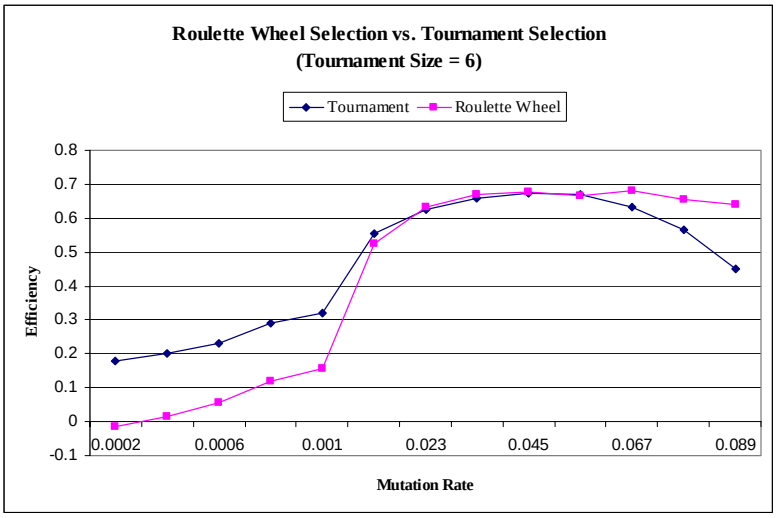

Figure 6. n = 80, p_c = .6, p_m = .0002 - .1, stable generations = 15, elitism = false.

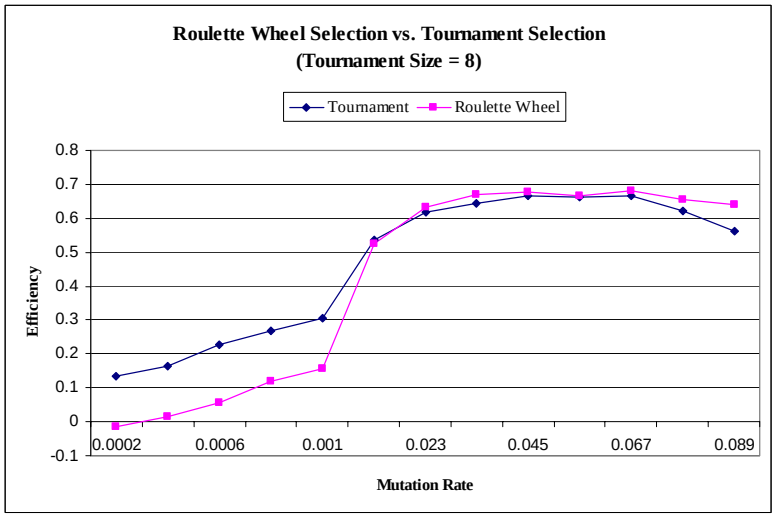

Figure 7. n = 80, p_c = .6, p_m = .0002 - .1, stable generations = 15, elitism = false.

5.4 Uniform Crossover

Traditional 2-point crossover takes parent individuals, randomly selects two cutoff points along the chromosomes, and then swaps all of the genes that fall in between those two points. Uniform crossover instead looks at each of the chromosomes' genes and swaps them with a probability equal to some specified swap rate.

In general, during crossover, all the schemata of an individual, including any schemata which may have been responsible for its selection, run the risk of being split apart and destroyed. The average amount of this disruption that is caused by a particular operator is referred to as its error rate [Sysw89]. A higher error rate will correspond to less exploitation and more exploration. The fact that uniform crossover provides an equal probability for each schema being split apart, regardless of its length, contributes to uniform crossover having a normally higher error rate than 2-point crossover. The effective difference in error rates, in our MFD problem, however, is not as substantial. This could be the result of large schema lengths for the MFD problem's optimal solutions. With large schema lengths, 2-point crossover would be just as likely to disrupt a schema as uniform crossover. Therefore, we would expect 2-point crossover and uniform crossover to provide similar results. In fact, we did not notice any improvement from the optimal standard variant settings. In Figure 8, we see a range of results which typifies the relation between uniform and 2-pt crossover in the MFD problem. It compares the results of using both 2-point and uniform crossover with roulette wheel selection, while varying both the population size and crossover rate. We see that the results are almost identical.

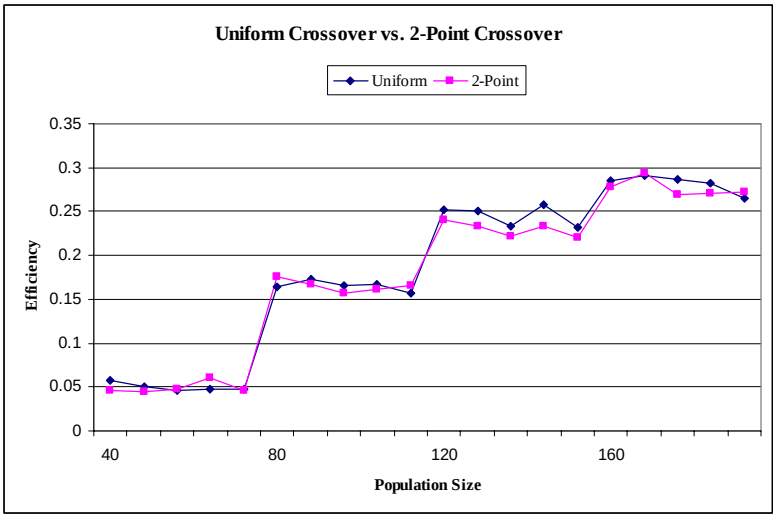

Figure 8. n = 40 - 160, p_c = .4 - .8, p_m = .001, selection method = roulette wheel, uniform swap rate = .5, stable generations = 15, elitism = false.

5.5 Rank Based Selection

Rank based selection is similar to roulette wheel selection in the sense that it randomly picks individuals with differing probabilities. The difference lies in how the probabilities are calculated. Roulette wheel selection assigns probabilities based on their relative fitness. Rank based selection assigns probabilities based on an individual's rank in the population. In this sense, it has much more in common with tournament selection. Again, we did not see any improvement to the top efficiency values. In Figure 9 we see a comparison of all three selection methods. Notice that the performance of rank based selection closely resembles that of tournament selection.

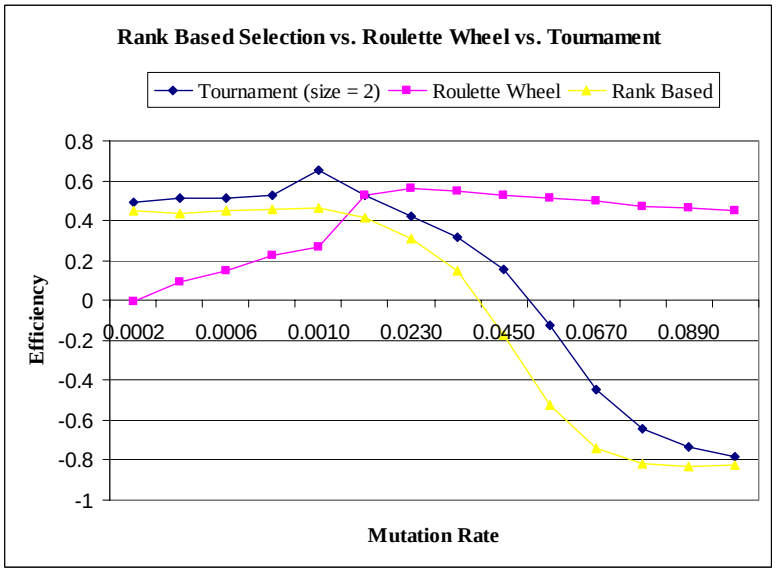

Figure 9. n = 160, p_c = .6, p_m = .0002 - .1, stable generations = 15, elitism = false.

5.6 Modified Mutation Rate

In another set of experiments, we attempted to introduce a new method for mutating offspring. We called this method the modified mutation scheme. Initially, the modified mutation scheme performs mutation on a new generation as it normally would with the mutation rate equal to some "base" mutation probability. Then, it takes the k worst individuals from the new generation and subjects them to the possibility of further mutation with some higher "modified" probability of mutation. The intent was to provide the advantages of increased exploration that come with higher mutation rates without the increased number of fitness evaluations that we would expect from applying such a high mutation rate to the entire population. In Figure 10 this method is compared to the results of running the GA using strictly the normal base rates as well as running the GA using strictly the higher mutation rate.

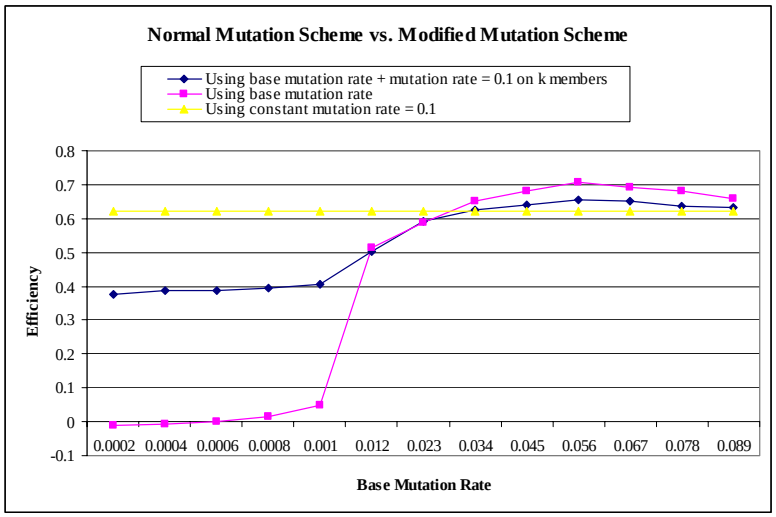

Figure10. n = 40, p_c = .6, p_m = .0002 - .089, modified p_m = .1, modified k value = 10, selection method = roulette wheel, stable generations = 15, elitism = false.

At no time does the modified mutation scheme ever outperform both of the normal mutation schemes at the same time. It does, however, reinforce the idea that operator influences are accumulated in a GA creating an overall balance between exploration and exploitation. The effect of the modified mutation scheme is really just to shift the balance a little bit more towards exploration. It has an accumulated mutation rate that is slightly more than its base rate and slightly less than its modified rate. This compromise is reflected in the results above.

6 Justification of Results

The basis for the effectiveness of any genetic algorithm is encompassed in the Schema Theorem and the Building Block Hypothesis. The Schema Theorem provides us with an estimation of the number of individuals containing a specific schema, or subset of genes, which appear in a population at any given time. For schemata with good attributes, this number should increase exponentially with each generation. The reason for this increase is that individuals containing good schemata will be chosen for reproduction by the GA's selection operator. Once these individuals are selected, the crossover and mutation operators effectively isolate the short, low order schemata which were responsible for the individual being selected in the first place. These schemata are referred to as building blocks. The Building Block Hypothesis states that by recombining these highly fit building blocks we can obtain highly fit individuals [Smit75].

The crossover and mutation operators of a GA work to explore the search space while the selection operator exploits past results. This process is typically able to find an optimal solution as long as the crossover operator has enough time to properly recombine the building blocks before the population converges to a sub-optimal solution. A dependency between crossover and selection can be observed in the sense that as the

selection operator chooses individuals whose schemata will be copied, it is also eliminating the other individuals, decreasing the diversity of the search space which is then available to the crossover operator for exploration. If a proper balance is not maintained between convergence time and exploration time, then the population will prematurely converge to a local optimum. Although we have only mentioned selection and crossover, in reality, all of the parameter settings contribute in one way or another to convergence and exploration times. Dirk Thierens introduces the Mixing Model [Thie96], to explain the proper balance required between parameter settings. The model is as follows, where n = population size, l = chromosome length, r = recombination rate, I = selection intensity and c is a constant value:

$$\frac{\sqrt{l}}{n} \leq c\frac{r}{I}$$

The method used to calculate the recombination rate will change depending on which type of crossover operator is being used. For a GA using uniform crossover:

$r = p_c p_r$, where p_c = crossover rate

and

$p_r = 2\ p_x\ (1\text{-}p_x)$, where p_x = uniform swap rate

The selection intensity is also dependent upon the selection method being used. When using tournament selection:

$$I \approx \frac{TournamentSize}{\sqrt{l}}$$

Our data seems to support the Mixing Model's relation. This was determined by taking the difference between the right side of the model and the left side of the model to obtain the following formula:

$$Mixing\ Model\ Difference = c\frac{r}{I} - \frac{\sqrt{l}}{n}$$

By applying this formula to each combination of parameter settings in our data set, we can determine whether or not they should produce satisfactory results. If the output of the formula is greater than zero then the parameter settings will be successful and if the output is less than zero, then the settings will be unsuccessful. In Figure 11, this direct correlation between the Mixing Model and our data can be seen. Notice that the cutoff value for the Mixing Model Difference correlates to an efficiency value of approximately 0.5000000 with this data. This is partly the result of our choosing a value of 0.275 for the constant c. By raising or lowering this constant we can effectively adjust the model's notion of a satisfactory result.

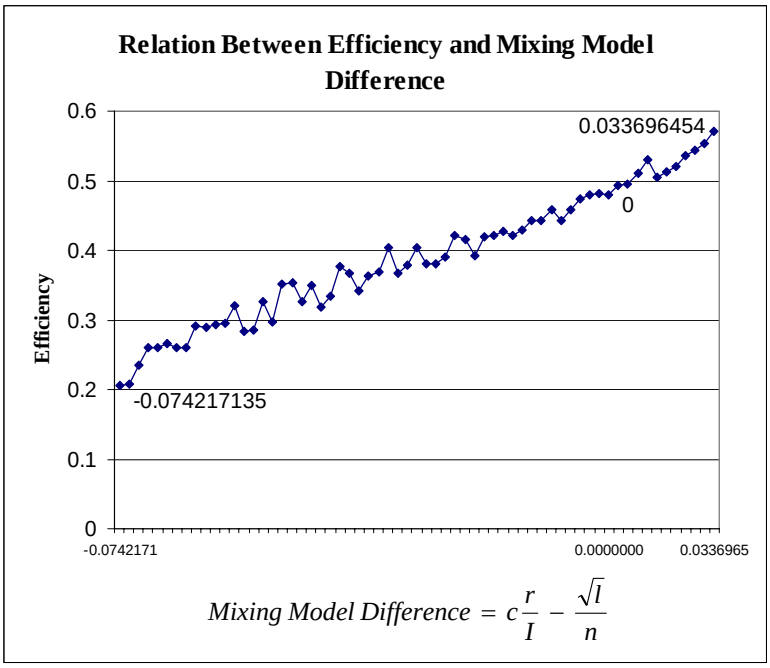

Figure 11. The above data was obtained by using the following parameter settings: n = 40, p_c = .45 - .8, p_m = .001, selection method = tournament (size = 2), uniform crossover (swap rate = .2-.8), stable generations = 15, elitism = false. Additionally the constant c was set to 0.275.

7 Conclusion

We explored using a simple GA to solve a Multiple Fault Diagnosis problem. It was shown that the optimal parameter settings generally require a small population size, paired with a large mutation rate. We introduced a new mutation scheme and examined its results along with two other non-standard variants. Although the overall efficiency was not improved upon by these new methods, they did help to frame performance in terms of a balance between exploration of the search space and exploitation of successful individuals. Finally, we showed the validity of the Mixing Model as a means of predicting the success of a given combination of parameter settings.

References

[DeJo75] DeJong, K.A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1975.

[Enge02] Engelbrecht, A (2002). *Computational intelligence*. Chichester, West Sussex: John Wiley & Sons.

[Eshe89] Eshelman, L. J., R. Caruana, R., J.D. Schaffer (1989). Biases in the crossover landscape. In J. Schaffer (Ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms* (pp. 10-19). San Francisco: Morgan Kaufmann Publishers.

[Gold85] Goldberg, D.E. Optimal Initial Population Size for Binary-coded Genetic Algorithms, TCGA Report Number 850001, The Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics University of Alabama, November 1985

[Gold89] Goldberg, D.E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

[Khar05] Kharbat, F, Bull, L, & Odeh, M (2005). Revisiting genetic selection in the XCS learning classifier system. In *The 2005 IEEE Congress on Evolutionary Computation*. *3*, 2061- 2068.

[Kolj06] Koljonen, J., J. Alander. (2006). Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms. In *Proceedings of the 12th Finnish Artificial Intelligence Conference STeP*. Helsinki.

[Peng87a] Peng, Y., J.A. Reggia. (1987). A probabilistic causal model for diagnostic problem solving, part I: integrating symbolic causal inference with numeric probabilistic inference. In *IEEE Transactions on Systems, Man, and Cybernetics* (pp. 146-162).

[Peng87b] Peng, Y., J.A. Reggia. (1987). A probabilistic causal model for diagnostic problem solving, part II: diagnostic strategy. In *IEEE Transactions on Systems, Man, and Cybernetics* (pp. 395-406).

[Pott90] Potter, W.D.B.E. Tonn, M.R. Hilliard, G.E. Liepins, R.T. Goeltz, S.L. Purucker. (1990). Diagnosis, parsimony, and genetic algorithms. In *Third International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert System.* New York, NY: Springer-Verlag.

[Scha89] Schaffer, J.D., R. Caruana, L.J. Eshelman, R. Das (1989). A study of control parameters affecting onine performance of genetic algorithms for function optimization. In J. Schaffer (Ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms* (pp. 51-60). San Francisco: Morgan Kaufmann Publishers.

[Smit75] Smith, R. *Genetic and evolutionary systems*. Department of Engineering Science and Mechanics University of Alabama, 1975.

[Sysw89] Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J. Schaffer (Ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms* (pp. 2-9). San Francisco: Morgan Kaufmann Publishers.

[Thie96] Thierens, D. (1996). Dimensional analysis of allele-wise mixing revisited. In H. Voigt, W. Ebeling, I Rechenberger, and H. Schwefel, (Eds.), *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature* (pp. 255-265). London: Springer-Verlag.