Empirical observations on the roles of crossover and mutation

Annie S. Wu
Naval Research Laboratory
Code 5514
Washington, DC 20375
aswu@aic.nrl.navy.mil

Robert K. Lindsay
Mental Health Research Institute
University of Michigan
Ann Arbor, MI 48109
lindsay@umich.edu

Rick L. Riolo
Prog. Study of Complex Systems
University of Michigan
Ann Arbor, MI 48109
rlriolo@umich.edu

In *ICGA-7*, 1997

Abstract

There is a great deal of information to be gained from studying the details within a GA run. This paper investigates the roles of crossover and mutation by observing the actions and effects of individual occurrences of each genetic operation. The observations are compared with some of the common expectations of these operators.

1 INTRODUCTION

The most common way to test the usefulness of a genetic operator has typically been to compare the performance of a genetic algorithm (GA) with that operator to the performance of a GA without that operator. If the former is better, then the operator is said to be useful; if the latter is better, then the operator is considered not useful. While such "macroscopic" observations and comparisons reveal important information to the study of GAs, the "microscopic" details of a GA run also hold interesting information on how the GA works (Nordin and Banzhaf 1995; Spears and De Jong 1991; Spears 1993; Thierens and Goldberg 1993; Wu and Lindsay 1996). To really understand the role of a genetic operator, it is necessary to look within a GA run at the actions and effects of the operator from generation to generation.

Two commonly used operators in the GA are crossover and mutation. Traditionally, crossover has been considered the primary operator of the GA, thought to contribute to both the generation of new potential solutions and the propagation of known partial solutions from generation to generation. The combination of crossover with the use of a population makes possible a unique search strategy described by the Building Block Hypothesis (Goldberg 1989; Holland 1992): sample the solution space in parallel for highly fit partial solutions or building blocks and combine them to form more complete solutions; repeat recursively until a complete solution is found. Mutation, on the other hand has traditionally been considered a "background" operator whose main contribution is thought to be to assure that no information is permanently lost from the population of potential solutions. Recent studies on both crossover and mutation, however, suggest that the contributions of these operators may not be so clearly defined (Eshelman and Schaffer 1993; Fogel and Stayton 1994; Hinterding, Gielewski, and Peachey 1995; Jones 1995; Schaffer, Caruana, Eshelman, and Das 1989; Spears 1993). This paper describes an empirical study of the roles of crossover and mutation, focusing on the effects of individual crossover and mutation events.

2 WHAT WE THINK THEY DO

Before dissecting a GA run, let us outline some of the typical expectations of crossover and mutation. As the primary sources of exploration, crossover and mutation play important roles in the GA search process. Many of these roles relate to *building blocks*: segments of an individual that affect that individual's fitness.

- Crossover is expected to propagate existing building blocks to the next generation. By manipulating "chunks" of an individual rather than individual bits, crossover is expected to be able to transfer entire building blocks from parent to offspring. Crossover is able to make potentially large changes from parents to offspring with minimal disruption of building blocks.
- Crossover is expected to combine existing building blocks. Lower order building blocks are more likely to be found through random sampling and

can be searched for in parallel on different members of the GA population. Higher order building blocks are easier to construct from combining lower order building blocks than from random sampling.

- Both crossover and mutation are expected to construct new building blocks and disrupt existing building blocks.
- Mutation is expected to introduce diversity. Since crossing over identical segments provides no new information, once the entire population possesses the same value at a particular bit position, only mutation is able to insert new values of that bit into the population.

3 EXPERIMENTAL DETAILS

Two programs were implemented for this study: a GA program and a visualization/analysis program. The GA program was executed on the Royal Road (RR) functions and the details of these runs were saved. The visualization program was then used to interactively analyze this saved data.

3.1 THE GENETIC ALGORITHM

A canonical, generational GA was used to generate data for analysis. The parameters for the runs to be discussed are as follows:

Population size:	128
Genome length:	64
Crossover type:	one point
Crossover rate:	0.7
Mutation rate:	0.005
Selection method:	roulette wheel
Parent selection:	with replacement

This GA was run on multiple RR functions and the entire runs saved for analysis.

3.2 THE ROYAL ROAD FUNCTIONS

The RR functions are a class of functions that were designed for studying building block interactions (Jones 1994; Mitchell, Forrest, and Holland 1991). With respect to the RR functions, a *building block* refers to a low order, low defining length group of bits on an individual that together contribute to that individual's fitness. The RR functions have multiple levels of hierarchically arranged building blocks. All building blocks are defined and assigned fitness contributions before a run begins. The optimum individual is defined in advance. The basic or lowest-level building blocks are shortest in length and upper-level building blocks consist of combinations of lower-level building blocks. The fitness of an individual is the sum of the fitness contributions of all building blocks that exist on that individual. The additional fitness support provided by the intermediate-level building blocks are expected to encourage the recombination and preservation of the basic building blocks, in effect laying out a "royal road" for the GA to follow to the optimum individual.

The RR functions were designed to be easy functions for the GA to solve. The hierarchical building block structure was expected to encourage the recombination of low order building blocks into higher order building blocks. Empirical tests, however, showed that the GA seemed to find RR functions with multiple levels of building blocks more difficult than RR functions with only a single level of building blocks (Forrest and Mitchell 1993). An examination of the distribution of building blocks from generation to generation revealed the reason for this unexpected result. The added fitness provided by the upper level building blocks appeared to be "too effective", causing non-optimal bits to "hitchhike" on individuals with highly fit building blocks, potentially disrupting building blocks on less fit individuals. Studies by Wu and Lindsay (1995, 1996) found that the addition of non-coding segments in the RR functions can reduce the disruption of building blocks during a GA run and that alternate representation styles may improve the diversity of the GA populations. All of these conclusions were possible because the RR functions allow researchers to examine the details of building block interactions during a GA run.

The pre-defined structure of the RR functions allows detailed monitoring of GA progress and makes these functions excellent candidates for our analysis. Table 1 shows the RR function used in this study. This RR function has eight basic (level 0) building blocks, fifteen total building blocks, and four levels. Each building block that exists on an individual contributes one point to that individual's fitness. A building block *exists* on an individual if its optimum pattern exists on that individual. A building block is *inherited* if it exists on an individual and one or both of its parents. A building block is *constructed* if it exists on an individual, but not on either of its parents. A building block is *disrupted* if it exists on an individual but not on any of its offspring. The goal of the GA is to find an individual with copies of all of the building blocks of the RR function.

Though the RR functions are not representative of many "real" problems, the pre-defined structure and

i	Schema b_i
0	11111111**
1	********11111111**
2	****************11111111**
3	************************11111111********************************
4	********************************11111111************************
5	**11111111****************
6	**11111111********
7	**11111111
8	1111111111111111**
9	****************1111111111111111********************************
10	********************************1111111111111111****************
11	**1111111111111111
12	11111111111111111111111111111111********************************
13	********************************11111111111111111111111111111111
14	1111111111111111111111111111111111111111111111111111111111111111

Table 1: The RR function, 8x8.RR2. Each $b_i, i = 1, 2, ..., 15$ shows the optimum pattern for one building block. $*$'s represent *don't care* bits. For each b_i, there is a value l_i which indicates the level of that building block, and a value c_i which indicates the fitness contribution of that schema. Here, $c_i = 1$ for all i and l_i ranges from 0 to 3 for the lowest to highest order building blocks.

relative simplicity of these functions allows us to investigate basic GA events such as how a building block is constructed or disrupted without having to worry about complex representation and fitness calculation issues. The observations made from this study can then be used as guidelines on how to approach more realistic problems.

3.3 THE VISUALIZATION PROGRAM

The visualization program allows for the interactive analysis of GA runs. It expects one file of data per generation of a GA run containing exhaustive details such as a copy of each individual of the generation, records of all crossover and mutation locations, and pointers to each individual's parents and offspring. This program is able to provide us with a "snapshot" of any particular instance of a GA run. Program functionality ranges from interactively displaying detailed information on populations or individual members of a population to collecting and calculating statistics from multiple runs.

4 OBSERVATIONS

The validity of the expectations described in section 2 is difficult to measure from observations of GA run outcomes alone. The difficulty arises from the difference between the *means* to an end and the *end* itself. Correct expectations are likely to lead to good GA performance, but good performance does not necessarily indicate that these expectations are correct. Using the visualization program described in section 3.3, we examined the effects of crossover and mutation on the building blocks of the RR function. This study focuses on the basic (level 0) building blocks. We compare our observations to the expectations from section 2.

4.1 PROPAGATION OF BUILDING BLOCKS

Does crossover propagate existing building blocks to the next generation? This question requires that we differentiate between building blocks that are constructed during recombination and building blocks that are inherited intact. The source of every building block on every individual is recorded. Five different sources are possible:

1. The building block is randomly generated as part of the initial population (only valid for building blocks from generation zero).
2. The building block is generated by a crossover event that combined parts of the building block from two parent individuals.
3. The building block is generated by a mutation event.
4. The building block is generated via both crossover and mutation.
5. The building block is inherited from an individual from the previous generation.

Analysis of multiple GA runs found that the majority of building blocks on any individual (except for those from generation zero) are inherited from the parents of that individual, confirming that crossover does propagate building blocks from one generation to the next. Figure 1 plots, for a typical run, the total number of building blocks in each generation (divided by 100) and the number of those building blocks that are constructed (using any of the first four methods). Less than one percent of the building blocks in each generation are constructed; most are inherited.

4.2 RECOMBINATION OF BUILDING BLOCKS

The ability to propagate building blocks from one generation to the next allows the GA to carry the information found in one generation to future generations. What we would like to see next is for crossover to actively combine different building blocks from different individuals onto one offspring individual. Such an observation would indicate that the GA is taking full

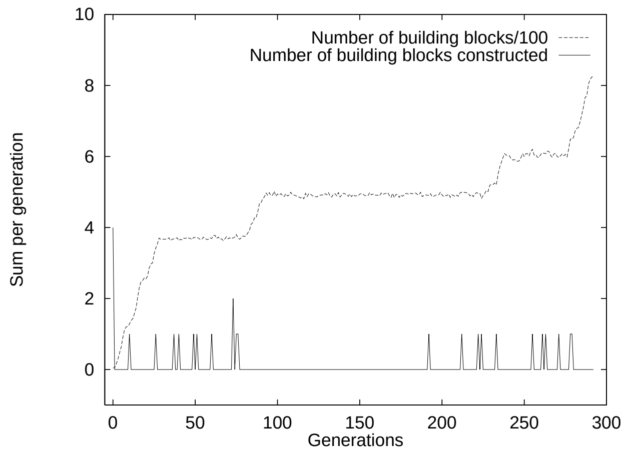

Figure 1: Total number of building blocks (divided by 100) and the number of those building blocks that are constructed in each generation of a typical run.

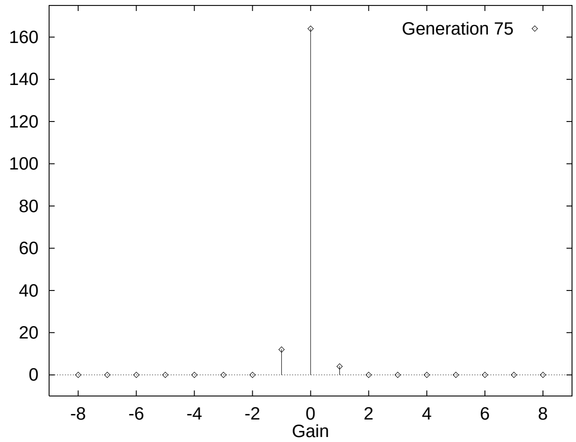

Figure 2: Typical histogram of the gain in the number of building blocks from parent to offspring due to crossover in one generation.

advantage of the combination of crossover and a population.

We calculate the *gain* or increase in the number of building blocks from parent to offspring for each crossover event. Negative gain values indicate loss of building blocks. Figure 2 shows a typical histogram of the gain values from every crossover event of a generation. Each time crossover occurs, a gain value is calculated for every parent-to-offspring transition (resulting in a total of four values per crossover). The shape of the gain histograms remains relatively constant throughout a run. The majority of crossover events result in no change in the number of building blocks from parent to offspring. These results agree in part with results from (Nordin and Banzhaf 1995).

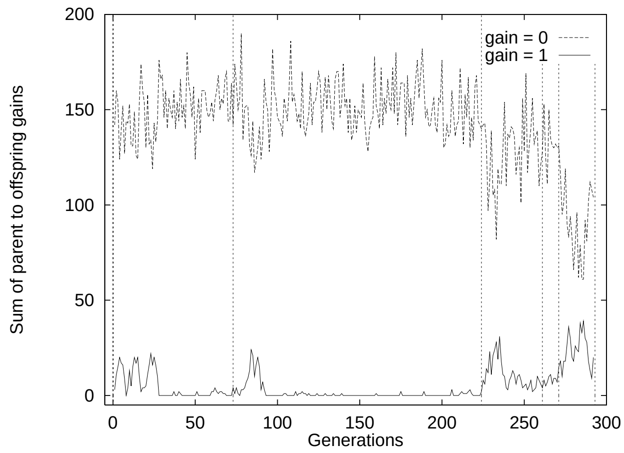

Figure 3: Number of parent to offspring transitions in each generation in which the gains in the number of building blocks are zero and one. Vertical lines indicate generations in which new building blocks are discovered.

Nordin and Banzhaf compared the fitnesses of the parents and offspring from a crossover event and found that crossover typically either caused little change or a strong negative shift in fitness.

The loss of one building block from parent to offspring is the next most common event. It occurs at a fairly constant rate throughout each run, involving approximately 10% to 15% of the crossovers in each generation. The gain of one building block from parent to offspring also occurs a significant number of times, but seems to be concentrated at various intervals throughout a run. These concentrations coincide with those times that new building blocks are discovered in the generation. Figure 3 shows the number of parent to offspring transitions that result in a gain of either zero or one building block. The vertical lines indicate generations in which new building blocks are discovered. Three building blocks existed in the initial population. Immediately after each new building block is discovered, there is a noticeable increase in the number of crossovers with gain of one building block. As inferred by Wu and Lindsay (1996), crossover is very effective at quickly propagating newly discovered building blocks to all members of the population as soon as those building blocks are discovered. We can see from Figure 1 that each growth spurt in the number of building blocks in the population increases this number by approximately 128 or one per individual in the population. Unfortunately, this efficient propagation comes at the cost of a loss of diversity in the population. As the population becomes more homogeneous, crossover becomes less productive. While a parallel

Num.	Total	Building blocks constructed				
Gen.	BB	Total	Init.	Xover	Mut.	X&M
171	98707	24	2	11	11	0
695	504333	39	5	19	13	2
515	328354	42	4	20	17	1
371	269260	39	4	19	15	1
181	117054	34	9	10	13	2
293	136622	26	4	10	8	4
562	359374	29	5	10	13	1

Table 2: Distribution of how building blocks are constructed in example runs.

search is still being performed on the remaining building blocks, the GA as a whole is essentially searching for one building block at a time, and not taking advantage of its population. This observation is further reinforced by the fact that gains and losses of more than one building block from parent to offspring are relatively rare. Though crossover does seem to be good at combining different building blocks onto one individual, the lack of building block diversity in a typical population prevents the GA from taking full advantage of this ability.

4.3 CONSTRUCTING NEW BUILDING BLOCKS

While it is important for the GA to save or exploit the partial solutions that it finds, it is also necessary to continue exploration of the solution space. Both crossover and mutation are expected to contribute to the construction of new building blocks: crossover by bringing together pieces of building blocks from different individuals onto one individual, mutation by randomly changing bits. We have already seen in section 4.1 that this constructive streak is extremely small.

Table 2 shows the distribution of how building blocks are constructed in a set of example runs. Column one shows the number of generations until an optimal solution is found; column two shows the total number of building blocks that existed during each run, summed over all individuals and all generations; and column three shows the number of building blocks constructed (i.e. not inherited) during each run. Columns four through seven of this table correspond to items one through four, respectively, from section 4.1. Except for generation zero, where all building blocks are generated as part of the initial population, what little constructive tendencies that exist seem to be divided relatively evenly between crossover and mutation, with crossover constructing an average of 14 building blocks per run and mutation, 13 building blocks per run. Interestingly, the number of building blocks constructed by crossover or mutation remain relatively constant in spite of the large differences in the lengths of the runs.

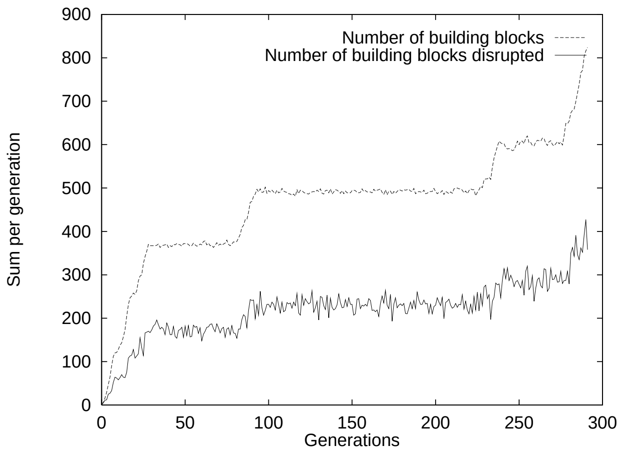

Figure 4: Total number of building blocks and the number of those building blocks that are disrupted in each generation of a typical run.

4.4 DISRUPTING EXISTING BUILDING BLOCKS

In addition to constructing new building blocks, the pseudo-random sampling of the solution space by crossover and mutation will also disrupt building blocks. Figure 4 plots for a typical run the total number of building blocks in each generation and the number of those building blocks that "die" or are not passed on to the next generation. Approximately 50% of the building blocks in any generation do not survive to the next generation. There are four ways that a building block can die:

1. A building block can die if the individual it is on has no offspring, i.e. is not selected to reproduce.

2. A building block can die if a crossover occurs within its boundaries. Note that this does not include instances where crossover falls within a building block that exists on both parents. In that event, the building block does not die since both offspring will receive complete copies of that building block.

3. A building block can die if mutation changes a bit within that building block

4. A building block can die if both crossover and mutation occur within its boundaries.

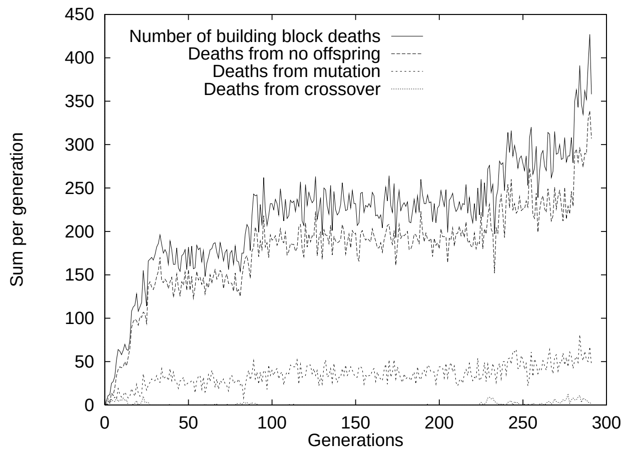

Figure 5: Plots of how building blocks are disrupted in a typical run.

Figure 5 shows a typical plot of how the building blocks of each generation of a run are disrupted. The top plot shows the total number of building blocks that die in each generation. The other plots correspond to items one through three above and show the distribution of how building blocks die. The majority of building blocks that do not survive into the next generation die because their individual failed to reproduce. Mutation is significantly more disruptive than crossover. Death by crossover alone or by crossover and mutation is relatively infrequent. These results agree with previous work, both empirical (Schaffer and Eshelman 1991) and theoretical (Spears 1993), which found mutation's potential for disruption to be much higher than that of crossover. If 50% of all building blocks in a generation are not passed onto the next generation (see Figure 4), the plots in figure 5 indicate that approximately 35% to 40% of building block die because their individual is not selected to reproduce. Accordingly, we find that in each generation the number of individuals that have no offspring averages approximately 40% throughout an entire run.

4.5 MAINTENANCE OF DIVERSITY

If we count the number of copies of each RR basic building block in each generation, we find that for most generations of a run, all of the individuals possess approximately the same building blocks. That is, a building block is typically found on either all individuals or no individuals of a generation. Why this low level of diversity? Though the crossover rate not yet at its maximum value, Figure 2 shows that the majority of crossover events do not alter the number of building blocks from parent to offspring. Mutation appears to be as active as crossover in the construction of building blocks and more active than crossover in the disruption of building blocks. Further increasing the mutation rate destroys the GA's ability to preserve building blocks. If crossover and mutation cannot provide the necessary diversity, are there other sources?

Figure 4 shows that in any generation, approximately 50% of the building blocks will not survive to the next generation. Figure 1 shows that in any generation, almost all building blocks are inherited from the previous generation. This means all of the building blocks in each generation are selected from a pool consisting of only half the building blocks of the previous generation. Though crossover and mutation provide some added diversity in each generation, the amount is very small. As a result, the GA is sampling from an increasingly limited and biased solution space as a run progresses. These observations suggest that the GA's selection method may be partially responsible for the level of diversity in a GA population. While selection is typically associated with the preservation and propagation of information in the GA, perhaps its role in encouraging diversity, for example in selecting a group of parents that are not only fit but also contain a variety of building blocks, should also be evaluated.

4.6 NON-CODING MATERIAL

One way to protect existing building blocks from crossover's destructive effects is with the insertion of "non-coding bits". Non-coding bits refer to bits on an individual that have no effect, positive or negative, on the fitness of that individual. A number of studies have examined the effects of non-coding bits on GA performance (Forrest and Mitchell 1993; Levenick 1991; Nordin, Francone, and Banzhaf 1995; Wu and Lindsay 1995; Wu and Lindsay 1996), including one that specifically reported a decrease in the destructive effects of crossover (Nordin and Banzhaf 1995).

Wu and Lindsay (1995, 1996) reported that, with the addition of non-coding bits, the building blocks of the RR function are less likely to be disrupted and reconstructed multiple times. We verify this conclusion by comparing the data from sections 4.3 and 4.4 with data from GA runs on a RR function with non-coding bits. The new RR function has the same number and size building blocks as our original function, but uses a genome length of 224 bits. The 160 extra non-coding bits are separated into blocks of twenty and placed in between the basic (level 0) building blocks of our original RR function.

Table 3 shows the number of building blocks disrupted by crossover and mutation for both sets of runs.

Genome length 64			Genome length 224		
# Gen.	Xover	Mut.	# Gen.	Xover	Mut.
171	543	7433	919	225	45196
695	1420	37528	530	230	27630
515	860	24621	441	207	21833
371	790	20181	298	116	12982
182	628	8741	317	171	14284
293	350	10196	491	150	21742
562	867	26805	736	257	36407

Table 3: The disruption of building blocks in runs with and without non-coding bits.

Num.	Total	Building blocks constructed				
Gen.	BB	Total	Init.	Xover	Mut.	X&M
919	567970	25	4	2	18	1
530	351093	19	3	2	13	1
441	276078	21	4	2	15	0
298	162816	22	2	6	14	0
317	183160	24	7	3	13	1
491	274573	28	3	6	17	2
736	457081	22	3	2	17	0

Table 4: Distribution of how building blocks are constructed in runs with non-coding bits.

Without non-coding segments, an average of 780 building blocks are destroyed by crossover in one run. With non-coding segments, this average drops noticeably to 194. The addition of non-coding segments does appear to protect building blocks from the destructive effects of crossover. Alone, this observation implies that adding non-coding bits would result in an immediate speed-up in GA performance. Further analysis, however, suggests that we should expect this decrease in destruction to be accompanied by a parallel decrease in construction (Wu and Lindsay 1996), since the addition of non-coding bits decreases crossover's overall activity within the building block regions of an individual. A comparison of the column five data from tables 2 and 4 shows that crossover constructs an average of 14 building blocks in runs without non-coding bits and an average of 3 building block in runs with non-coding bits. Thus, while non-coding bits do protect existing building blocks from crossover's destructive effects, they also slow down the discovery of new building blocks.

5 CONCLUSIONS

While it is important to study average GA performance over many runs, it is also important to understand the details of what is happening during a run. This paper describes observations made on the effects of individual crossover and mutation events on a set of GA runs. The following conclusions may be drawn from our observations.

Though crossover is relatively inactive in the construction and disruption of building blocks, it is doing its job of propagating building blocks and it is good at recombining building blocks. Unfortunately, the GA is not able to take full advantage of crossover's recombination ability due to the low building block diversity in most generations. Mutation provides some of the necessary diversity to a GA population; it is considerably more active than crossover at destroying building blocks and as good or better than crossover at constructing new building blocks. Increasing the genome length, and consequently the amount of non-coding material makes crossover less active within building block boundaries, but does not affect mutation's constructive or destructive effects on building blocks.

Consistent with the low number of building block constructions in GA runs, the majority of building blocks on any given individual (except for individuals from the initial population) are inherited from the individual's parents. Approximately fifty percent of all of the building blocks in a generation do not survive into the next generation; the majority die because their individual was not selected to reproduce. The remaining fifty percent of building blocks becomes the pool from which almost all of the building blocks from the next generation are selected, causing the GA to sample from an increasingly limited and biased solution space as a run progresses.

The expectations of the individual contributions of crossover and mutation seem to be correct, but the interaction of their roles and the role of selection require further investigation. Though the GA is able to construct and maintain building blocks in its populations, the lack of building block diversity prevents effective recombination of the building blocks. These observations suggest that it may be useful to consider less severe selection schemes or alternate methods such as crowding or sharing that encourage a less homogeneous population.

6 DISCUSSION

In addition to gaining a better understanding of how the GA works, an additional goal of this paper is to encourage a more fine-grained study of the GA. Suggested extensions to this work emphasize two directions: analysis of the construction and destruction of the upper level building blocks of the RR function and analysis of a wider range of GA control parameter values and a wider variety of problems.

This paper concentrates on the effects of genetic operations on the basic building blocks of the RR functions. It is also important to understand how well the GA is able to combine these basic building blocks. This question is addressed to some extent in the crossover gain analysis (section 4.2). An analysis is currently being performed on the construction and disruption of upper level Royal Road building blocks.

Variation in the GA control parameters would likely change some of the observed behaviors. For example, changing the population size or scaling individual fitnesses are expected to affect selection pressure and the inheritance of building blocks. Alternate types of crossover, such as uniform crossover, are expected to have varying effects on the propagation of information (Eshelman, Caruana, and Schaffer 1989; Spears 1993).

Though the structure of the RR functions make them ideal for these analyses, one must be careful not to generalize too much from experiments on a single set of functions. For such analyses to be useful, it is necessary to expand to other functions, both artificial and real.

Acknowledgments

This work was supported by the National Science Foundation (Grant IRI-9224912), the Program for the Study of Complex Systems at the University of Michigan, the National Research Council, and the Office of Naval Research. The authors wish to thank John Holland, Bill Spears, and the anonymous reviewers for their helpful comments.

References

Eshelman, L. J., R. A. Caruana, and J. D. Schaffer (1989). Biases in crossover landscape. In *ICGA-3*, pp. 10–19.

Eshelman, L. J. and J. D. Schaffer (1993). Crossover's niche. In *ICGA-5*, pp. 9–14.

Fogel, D. B. and L. C. Stayton (1994). On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems 32*, 171–182.

Forrest, S. and M. Mitchell (1993). Relative building-block fitness and the building-block hypothesis. In *Proceedings of the Foundations of Genetic Algorithms Workshop 2*, pp. 109–126.

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

Hinterding, R., H. Gielewski, and T. C. Peachey (1995). The nature of mutation in genetic algorithms. In *ICGA-6*, pp. 65–72.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Jones, T. (1994). A description of Holland's Royal Road function. *Evolutionary Computation 2*(4), 409–415.

Jones, T. (1995). Crossover, macromutation, and population based search. In *ICGA-6*, pp. 73–80.

Levenick, J. R. (1991). Inserting introns improves genetic algorithm success rate: Taking a cue from biology. In *ICGA-4*, pp. 123–127.

Mitchell, M., S. Forrest, and J. H. Holland (1991). The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 245–254.

Nordin, P. and W. Banzhaf (1995). Complexity compression and evolution. In *ICGA-6*.

Nordin, P., F. Francone, and W. Banzhaf (1995). Explicitly defined introns and destructive crossover in genetic programming. In *Advances in Genetic Programming II.* MIT Press.

Schaffer, J. D., R. A. Caruana, L. J. Eshelman, and R. Das (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In *ICGA-3*, pp. 51–60.

Schaffer, J. D. and L. J. Eshelman (1991). On crossover as an evolutionarily viable strategy. In *ICGA-4*, pp. 61–68.

Spears, W. M. (1993). Crossover or mutation? In *Proceedings of Foundations of Genetic Algorithms Workshop 2*, pp. 221–237.

Spears, W. M. and K. A. De Jong (1991). On the virtues of parameterized uniform crossover. In *ICGA-4*, pp. 230–236.

Thierens, D. and D. Goldberg (1993). Mixing in genetic algorithms. In *ICGA-5*, pp. 38–55.

Wu, A. S. and R. K. Lindsay (1995). Empirical studies of the genetic algorithm with non-coding segments. *Evolutionary Computation 3*(2), 121–147.

Wu, A. S. and R. K. Lindsay (1996). A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation 4*(2), 169–193.