Proceedings of the 1996 ACM/SIGAPP Symposium on Applied Computing, February 18-20, 1996, Philadelphia, PA., Pages 269-276, ACM Press.

REDUCING DISRUPTION OF SUPERIOR BUILDING BLOCKS IN GENETIC ALGORITHMS *

Arthur L. Corcoran
Octel Network Services
17060 Dallas Parkway, Suite 214
Dallas, TX 75248
artc@ons.octel.com
http://euler.mcs.utulsa.edu/~corcoran

Roger L. Wainwright
Dept. of Mathematical and Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, OK 74104-3189, USA
rogerw@penguin.mcs.utulsa.edu
http://euler.mcs.utulsa.edu/faculty/wainwright.html

Key Words: Genetic Algorithms, Building Blocks, Reduction, Masking, Relative Bonding Strength

ABSTRACT

The strength of genetic algorithms is the ability to recombine short, highly-fit schemata (i.e., building blocks) to produce longer, more highly-fit schemata. Unfortunately, this benefit can be diminished by the effects of hitchhiking, allele loss, disruption, etc. In many cases this is due to the absence of information to indicate the relative strengths of the genes or the bonding between the genes. Consequently, beneficial building blocks are just as likely to be disrupted as harmful ones. In this paper, we describe several methods which can be used to preserve sequences and distributions of beneficial building blocks. Most research to date attempts to minimize the negative aspects of disruption. This research emphasizes the positive aspect of identifying and manipulating optimal sequences.

INTRODUCTION

In this paper, we present several methods which can be used to identify and preserve sequences and distributions of optimal or near-optimal building blocks in genetic algorithms. Attention is given to identifying highly-fit schemata within a chromosome and manipulating and preserving such schemata. The concept of a *sliding window* is presented to identify highly-fit sequences. In some cases these highly-fit sequences can be moved to the front of the chromosome. This is called *reduction*. This has the added benefit of reducing the size of the problem under investigation.

Some problems may not have easily identifiable optimal subproblems. However it may be easy to determine the strength of the bonds between adjacent genes or a sequence of consecutive genes. We develop the concept of a bit mask associated with a chromosome that identifies highly-fit sequences. We also introduce the concept of *relative bonding strength* which exploits the strong bonding between genes in a chromosome. We investigate how to identify and preserve these strong bonds during crossover. We also provide some practical applications illustrating the success of the concepts presented in this paper.

Sequences

A *sequence* is a contiguous group of genes within a chromosome. That is, it is a schema in which the defining length and the order are equal. Highly-fit sequences correspond to beneficial building blocks. Once identified, these beneficial building blocks can be preserved. The genetic algorithm can then manipulate the remaining genes to provide better results in less time. We have developed several techniques for identifying and preserving highly-fit building blocks.

Sliding Window

The *sliding window* is one approach which we have developed to identify highly-fit sequences. The sliding window is defined by *left* and *right* pointers which correspond to the two end genes in a sequence, as illustrated in Figure 1.

*Research partially supported by OCAST Grant AR2-004 and Sun Microsystems, Inc.

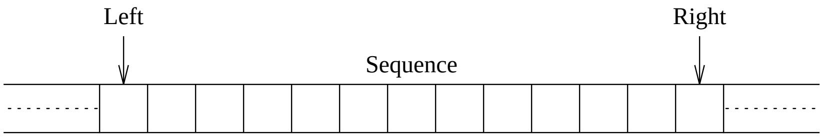

Figure 1: Sliding window

Initially when identifying highly-fit sequences the window is positioned on the first gene (*left* and *right* point to the first position). Beginning with the first gene in the chromosome, genes are added to the window on the right or removed from the window on the left as needed to form candidate sequences. A candidate sequence is then evaluated with a problem-specific function provided by the user to determine its fitness. If the sequence under investigation is considered *short* of a highly-fit sequence according to the problem-specific evaluation function, then genes are added to the right of the window one at a time and the sequence is reevaluated. If the sequence is considered *long* for a highly-fit sequence then genes are removed from the left of the window one at a time and the sequence is reevaluated. Sequences which are neither *short* nor *long* are considered highly-fit.

The boundary between 'too short' and 'too long' is problem dependent. In the strictest sense, an exact match to some criteria is required for the sequence to be considered highly-fit. This corresponds to preserving only optimal sequences. However, fuzziness can be introduced by allowing a small tolerance when matching the window. This corresponds to the preservation of near optimal sequences.

While the sliding window itself is a linear scan of the chromosome, the evaluation of the sequences are problem dependent and may not be very efficient. In many circumstances, the entire sliding window algorithm may be performed in conjunction with the GAs evaluation function. In this way, duplication of effort could be reduced or eliminated.

Reduction

Once highly-fit sequences have been identified, the chromosome can be manipulated to preserve the highly-fit sequences during crossover and mutation. In the case of movable, optimal sequences, *reduction* of the chromosome can take place. That is, optimal sequences can be moved to the front of the chromosome so they can be preserved more easily during crossover. The remaining genes form a chromosome of reduced length.

Figure 2a illustrates a chromosome to be reduced. Genes to the left of the *reduction boundary* are optimal sequences resulting from previous reductions. The optimal sequence to the right of the *reduction boundary* is appended to the optimal sequence at the left, and the *reduction boundary* is adjusted. This results in the reduced chromosome shown in Figure 2b.

The intent of reduction is to concatenate all of the optimal sequences together in one place. Hence when an optimal sequence is located it is moved to the front of the chromosome along with the others. This usually means that the entire collection of optimal sequences is highly-fit according to the problem dependent criteria. However, it is essential that the problem under investigation have easily identifiable optimal subparts or this technique may not be very useful. For example, consider the classic bin packing problem. Suppose a sequence of packages can be found that completely fill a bin. The optimally packed bin can be set aside, and the problem is effectively reduced in size. Almost all routing, layout, scheduling and packing problems have easily identifiable subparts and are candidate problems for the techniques developed in this paper.

There are several advantages of chromosome reduction. It effectively shortens the chromosome and concentrates on a smaller portion of the problem. Thus, more work can be done in the same amount of time. It isolates the superior sequences so they cannot be disrupted. Another very important advantage is that the traditional crossover operators (with minor modification) can still be used without fear of causing disruption of superior sequences.

A possible disadvantage of chromosome reduction is the possibility of premature convergence which misses the global optimum. For example, consider the one dimensional bin packing problem where the list of objects (3,3,4,6,7,7) are to be packed into bins of size 10. The first three objects in the list are used to create an optimally filled bin, and are set aside by reduction. The remaining three objects each require a bin, resulting in a total of four bins. However, the optimal packing uses three bins. This example illustrates the 'worst case' which is used to prove the performance bound for the next fit packing strategy. Fortunately, the genetic algorithm maintains a diverse population of chromosomes. It may be just as likely for optimal sequences to be present in the pool as for the worst case. This is why the genetic algorithm generally outperforms deterministic approximation algorithms like First Fit Decreasing. However, it does suggest that as the genetic algorithm converges, it may be useful to allow reduced chromosomes to be disrupted to some extent. Even

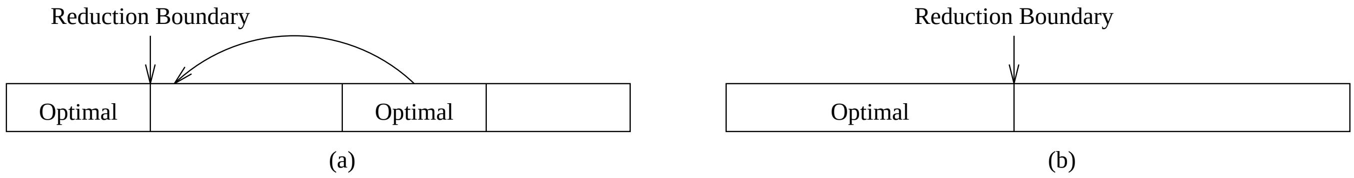

Figure 2: Chromosome reduction

when premature convergence cannot be avoided, reduction should at least improve the bounds over the deterministic strategies.

DISTRIBUTIONS

In this Section we present several ways to preserve optimal subsequences and prevent their disruption during crossover and mutation. Since highly-fit sequences are not always adjacent and often cannot be moved, more general mechanisms for preserving highly-fit schemata are presented.

Masks

The most natural way to preserve individual genes in a chromosome is to provide a bit mask the same length as the chromosome. Each bit in the mask indicates whether the corresponding gene is highly-fit or not. For example, suppose that in the chromosome ABCDEFGH, genes A, B, D, and F are considered highly-fit. The associated bit mask,

```
A B C D E F G H
1 1 0 1 0 1 0 0
```

makes this fact explicit by indicating the highly-fit genes with '1' bits.

When using a bit string representation for the chromosome, the utility of a bit mask is not immediately apparent. For example, the highly-fit values in the mask indicate that the corresponding genes should remain unchanged. Furthermore, this implies that the alleles for the remaining genes should be changed. Thus, by simply inverting these bits the optimal can be trivially found. While this may be true in simple problems, there are two issues that may prevent this when applied to practical problems.

The first issue is the presence of competing schemata with dissimilar and contradicting bit patterns, which are common in deceptive and multivalued problems. For example consider the following schemata and respective fitnesses:

$$\begin{array}{lll}
S_1 = \texttt{0000****} & f(S_1) = & 4 \\
S_2 = \texttt{****0000} & f(S_2) = & 4 \\
S_3 = \texttt{00000000} & f(S_3) = & 8 \\
S_4 = \texttt{11111111} & f(S_4) = & 16
\end{array}$$

Consider a candidate chromosome:

```
 String:  00001111
   Mask:  11110000
Fitness:      4
```

Inverting the bits that are not highly-fit produces the chromosome:

```
 String:  00000000
   Mask:  11111111
Fitness:      8
```

indicating that the optimal has been found. However, this is the *complement* of the true optimal in this contrived case.

The other issue is the probability of finding highly-fit schemata in a chromosome. A solution to a needle in a haystack type problem will not be found any quicker with a mask than without a mask. However, once a highly-fit schema has been found, the mask will ensure its survival. The mask will not otherwise effect the genetic algorithm.

In the case of movable, optimal sequences, the bit mask can be used to simulate chromosome reduction without the need for the *reduction boundary* pointer. The optimal sequences are moved to the front as before and the mask bits set for every gene to the left of where the *reduction boundary* would normally be. This results in a mask with all set bits grouped to the left and the unset bits to the right. In some instances an optimal sequence may not be able to be moved. In this case the mask can be set in the current location of the sequence and still be used effectively to preserve the sequence. The mask concept can be especially useful for near optimal sequences as well. In this instance the bit mask is used to identify adjacent genes with strong bonds.

One minor problem with masks is that traditional crossover operators must be adapted to use the information it provides. This generally requires a simple modification. Several suggestions are given below:

- For many crossover functions a postprocessing step could be added to ensure the masked genes survive. The crossover could create children as it normally does, and the postprocessing operation could undo any damage created. This is especially useful in single and multipoint crossover techniques.

- The uniform order crossover (UOX) operator of Davis [5] uses randomly generated bit masks. These could easily be replaced by the user defined bit mask from each parent chromosome. An element of randomness could be preserved by performing a logical OR between the random and user defined masks. We define this operator as UOX/MASK.
- For order based crossovers, such as those described by Starkweather *et al.* [11], the above methods may prove impractical. In these cases the crossover operator may need to be rewritten in order to take advantage of the user defined bit mask.

The mask concept is not limited to just bit masks. Integers or real values could be used just as easily. For example, the fitness for a gene can be represented as a real number or a normalized fraction in the range [0..1]. A mapping function could be used to adaptively discretize real valued masks and transform them into integer or bit masks. Non-bit values provide more information, and may be more useful in discriminating between distinct schemata or to distinguish groups of related genes. Furthermore, the relative strength of the bond between two adjacent genes is more easily represented by non-bit masks.

Relative Bonding Strength

Relative Bonding Strength (RBS) is a concept developed as part of this research which is used for both identifying and preserving distributions in a chromosome. While the mask concept is especially useful for preserving distributions based on position, RBS preserves distributions based on relative order. In this respect, RBS is similar to reduction, however, RBS does not require the chromosome to be partitioned. With RBS, each gene has a bond to one or more other genes in the chromosome. The bond may or may not be directed or symmetrical. Strong bonds are less likely to be broken than weaker ones. Bond strengths can be represented using floating point, integer, binary, or any other useful representation.

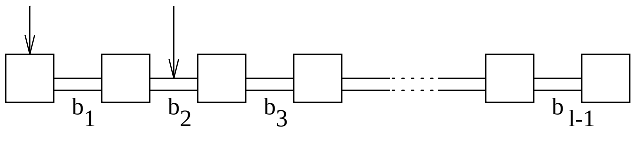

Figure 3: Chain bond

Figure 3 illustrates a chromosome of length l with a single, undirected bond between each gene. The strength of the bond is represented by a *bond vector*, $\bar{b} = (b_1, b_2, b_3, \ldots, b_{l-1})$. This bond represents a *chain* of genes. A *ring* of genes is formed by adding a bond, b_l to the bond vector, $\bar{b} = (b_1, b_2, b_3, \ldots, b_{l-1}, b_l)$. The extra bond connects the first and last genes in the chain. Ring bonds are used in problems such as the traveling salesman problem, where a sequence of genes represent a circular tour. Directed bonds can be represented by adding a second bond vector: the first vector indicating bonds in one direction and the second vector indicating the bond in the opposite direction. Alternatively, each element of the undirected bond vector b_i could be replaced by a pair of directed bonds $(b_{i,i+1}, b_{i+1,i})$. Bond $b_{i,i+1}$ indicates the directed bond strength from gene i to $i+1$; bond $b_{i+1,i}$, from gene $i+1$ to i.

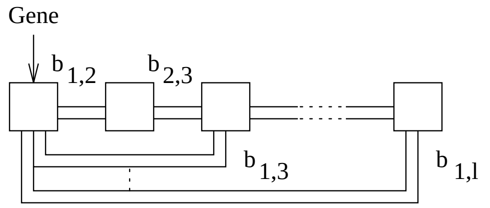

Figure 4: Multiple bonds per gene

Figure 4 illustrates a chromosome in which there exists a bond between every pair of genes. The bond strengths are represented using a *bond matrix*, where $b_{i,j}$ is the strength of the bond directed from gene i to gene j. In the case of undirected bonds, $b_{i,j} = b_{j,i}$ for every $i, j = 1, 2, \ldots, l$. Matrix bonds are especially useful in problems such as the Package Placement Problem, Fuzzy Logic Controllers, Neural Networks, etc.

These structures can be used to identify distributions, but it is also important to be able to preserve and manipulate them. Bond values could be integer, floating point, or a bit value. The simplest technique is to use a binary representation for the bond. A one bit indicates the bond should be preserved and a zero bit allows the link between the respective genes to be severed. Floating point representation for the bonds provides more opportunity for variation. The bonds could be converted to binary using a function of the bond strength. For example, bonds stronger or weaker than the average could be converted to ones or zeroes, respectively. The function could be skewed in a linear or nonlinear fashion as needed. To reduce the computational effort required to find the average, a small random sample of the bonds could be averaged. An integer bond strength representation could be used to discretize the floating point bonds, or to separate them into groups. It could extend the binary bonds to groups of bonds or bond strengths, or to enumerate disjoint sequences.

RBS can be used to guide crossover. For example, asexual crossover should not disrupt stronger bonds, but rather should move genes with weaker bonds. As another example, the bond vector could be converted to a bit mask based on the variance from the true or small sample average. This mask could be combined with the mask used by Uniform Order Crossover (UOX) to ensure the strong bonds survive. This variant will be referred to as Mask-UOX. RBS is a general mechanism which formalizes the many attempts to improve crossover in a GA. For example, RBS uses a bond vector or matrix for each chromosome in the population. As a special case, the vector or matrix can be the same in each chromosome. In this case, the vector or matrix can be removed from the chromosomes and be used as a global data structure. This is similar to the global precedence vector used by Blanton and Wainwright [1] for the Vehicle Routing Problem, which they developed with great success.

Relative Bonding Strength and Vehicle Routing

There are many applications where the bonding between genes in a chromosome can be applied. As a simple example, consider the Vehicle Routing Problem (VRP). The VRP problem involves determining minimum cost vehicle routes for a fleet of vehicles that originate and return from a central depot. Additional constraints include a time window when each customer can be serviced. In addition, each customer has a specified capacity of goods to receive. Minimum distance routes must be determined for each vehicle such that all customers are serviced within their allowed time window and the capacity of each vehicle is not exceeded. Examples include bank deliveries, refuse collection, school bus routing, postal deliveries, overnight deliveries, and most routing and scheduling problems. For more details using GAs with VRP see [1]. In order to simplify this problem, consider a single vehicle with no time window or capacity constraints. This degenerates to the Traveling Salesman Problem (TSP).

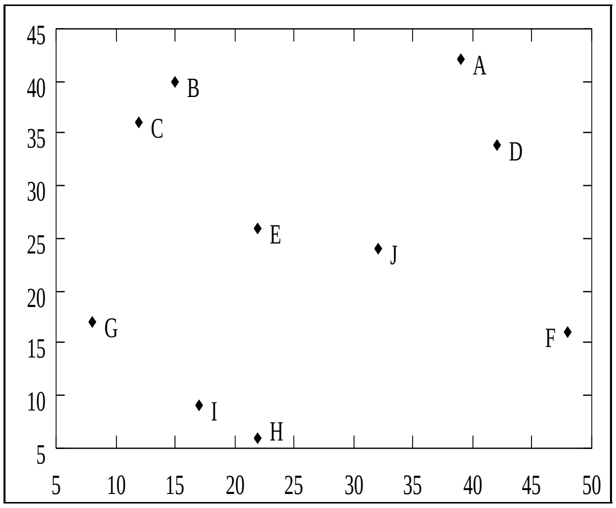

Figure 5: Customers for VRP

Consider the ten customers A through J depicted in Figure 5. The customers were placed randomly in a 50 by 50 grid. The resulting customers are located at: A(39,42), B(15,40), C(12,36), D(42,34), E(22,26), F(48,16), G(8,17), H(22,6), I(17,9), and J(32,24). An example tour is shown in the chromosome below.

Tour Length = 213

A	B	C	D	E	F	G	H	I	J
24	5	30	22	28	40	18	6	21	19

The value shown below each gene (customer) indicates the Euclidean distance between that customer and the next one. That is, the Euclidean distance between customers D and E is 22, and between J and A is 19. For convenience all values have been rounded to integers. The Euclidean distance serves as the relative bonding strength measurement between adjacent genes.

The relative bonding strength may be used in several ways to assist in the crossover operation depending on the application and the crossover operator. In the TSP, one option is to identify the two weakest bonds for the purpose of an asexual crossover. An asexual crossover is simply a swap of two selected genes. In this case, the weakest bonds are CD and FG, hence the asexual crossover operator could be defined to exchange D and G producing the following child:

Tour Length = 183

A	B	C	G	E	F	D	H	I	J
24	5	10	17	28	19	34	6	21	19

The exchange points for the asexual crossover could be determined by a selection bias similar to the criteria used to select chromosomes. That is, according to some bias factor, select genetic bonds that have a tendency for weakness just as chromosomes are selected from a population that tend to be more fit. Suppose a bias factor was used to select weaker bonds on the original chromosome A through J resulting in the selection of AB and CD. These turn out to be the second and fourth worst bonds in the chromosome. In this case B and D are exchanged resulting in the following child:

Tour Length = 191

A	D	C	B	E	F	G	H	I	J
9	30	5	15	28	40	15	6	21	19

For long chromosomes several less exhaustive strategies could be used for locating weak bonds. For example, randomly select an entry point in a chromosome and canvass the relative bond strengths for the next x genes (sample window) to determine the maximum value. Then select the first gene after the window larger than the maximum. This could be done twice to determine two exchange points. Depending on the application, canvassing the average or minimum relative bond strengths of a window might be appropriate. The window size, x, could be a fixed value or a percent of the chromosome length.

The relative bond strength measurement can be converted to a bit mask. For example, an average relative bond strength value for a chromosome can be calculated and used to convert all values weaker than the average to zero, and all values stronger than the average to one. The goal is to generate approximately 50% one bits. To save time a sample window could be used to determine a pseudo average, as described above. Consider the following random tours of customers from Figure 5. The first parent has an average relative bond strength just over 28, and the second parent has an average relative bond strength just under 28. The corresponding bit mask was generated accordingly.

Parent 1: Tour Length = 277

C	A	G	D	J	B	H	F	I	E
28	40	38	14	31	34	28	32	18	14
1	0	0	1	0	0	1	0	1	1

Parent 2: Tour Length = 274

B	D	I	A	E	G	C	J	H	F
28	36	38	23	17	19	23	21	28	41
0	0	0	1	1	1	1	1	0	0

At this point several options are possible for the bit mask. Suppose a uniform order crossover (UOX) is applied. The traditional UOX, however, uses a randomly generated bit mask. Since a bit mask is used to depict the strength of the bonds between genes, we will use the Mask_UOX operator. Applying Mask_UOX to the parent chromosomes yields the following two children:

Child 1: Tour Length = 204

C	B	A	D	G	J	H	F	I	E
8	24	9	38	12	21	28	32	18	14

Child 2: Tour Length = 270

D	B	H	A	E	G	C	J	F	I
28	35	40	23	17	19	23	18	32	35

Notice a single isolated strong genetic bond (one-bit) can be disrupted during this crossover. However, a series of strong genetic bonds are preserved during Mask_UOX. Notice in all cases the new operators, making use of the relative bond strength values, resulted in children with improved fitness values. There is no guarantee this will always occur. However, this shows the added genetic information provided by the relative bond strength values can produce a significant benefit for many applications.

There are numerous strategies and uses for relative bond strength values. For example, the relative bond strength values can be used to develop several adaptive bit mask strategies. For example, in early generations it may be prudent to generate a relatively small percentage, y, of one bits from the relative bond strength values. Then as the population matures, this percentage can slowly increase in order to preserve a higher percentage of strongly bonded genes. As a second example, during early generations a relatively small percent, y, of the strongly bonded genes are identified and assigned one bits. Furthermore, another z percent of weakest bonded genes are identified and assigned zero bits. The remaining positions of the bit mask are assigned bit values randomly. As the population matures, y increases slowly, and z decreases slowly. Considerable work needs to be done in this area.

EXPERIMENTAL RESULTS

To further illustrate optimal and near-optimal sequences, as well as masks and RBS concepts, we present several experimental results which show the success of these methods.

Reduction and Bin Packing

Genetic algorithms have been used with great success in both two dimensional bin packing [4] and three dimensional bin packing [3]. These results showed that genetic algorithms greatly improve packing performance over traditional algorithms such as First-Fit Decreasing. Our research to date [3, 4] has investigated the possibility of getting even better results for the genetic bin packing algorithm by incorporating techniques for identifying, isolating, and preserving optimal sequences. The genetic algorithm package, LibGA [4], was used to implement reduction on a two dimensional bin packing problem. Coffman *et al.* [2] reports the best deterministic algorithm in this case is *split-fit* (SF), which guarantees performance bounds of 1.5 times optimal. Other algorithms include *first fit decreasing height* (FFDH) with bound of 1.7 times optimal and *next fit decreasing height* (NFDH) with bound of 2.0 times optimal. *Next fit* (NF) was used as the objective function for the following reasons. Unlike the other methods, it guarantees unique packings for every permutation of the objects. NF is by far the simplest of the algorithms to implement. As an objective function which is called upon repeatedly to evaluate chromosomes, its $O(n)$ efficiency proves more cost effective than the $O(n \log n)$ efficiency of the other methods. It avoids the unnecessary step of presorting the objects by decreasing height as required in NFDH and FFDH, where the presort negates the property of unique permutations.

Each entry in Table 1 depicts the average fitness values obtained from executing the bin packing genetic algorithm ten different times using different random seeds. The pool size was 100 in every case. A steady-state GA and a generational GA were tested. The steady-state algorithm used a rank-biased selection similar to that in Genitor [12], with replacement by rank. The generational GA used typical parameters including roulette selection with elitism. The crossover used in both types of GAs was asexual. The data sets include several contrived, level-oriented data sets denoted as L25, L75, and

Data Set	Steady-State TRAD	Steady-State MOS	Generational TRAD	Generational MOS	Optimal	Best *OPT
L25	24.8	22.6	22.3	20.3	20	1.015
L75	76.5	73.8	69.7	67.7	60	1.128
L100	102.9	101.9	94.5	91.1	80	1.139
R25	29.8	28.9	27.7	26.4	24.7	1.069
R50	56.0	55.2	50.8	49.9	44.8	1.114
R100	106.5	103.9	94.7	94.5	82.6	1.144

Table 1: Bin packing results using TRAD versus MOS

L100, consisting of 25, 75, and 100 objects, respectively. These data sets when optimally packed consist of a series of optimally packed levels. Several random data sets, R25, R50, and R100, were generated which consisted of 25, 50, and 100 objects respectively. The results under 'TRAD' in the table are for the traditional GA. The results under 'MOS' are for the traditional GA augmented with a sliding window identification of optimal sequences and a subsequent reduction of the chromosome by the movement of the optimal sequence. MOS stands for 'Move Optimal Sequences'. The 'move to the front' has an added benefit in the case of bin packing. This movement ensures that the optimal sequences are now aligned on a level boundary. Most likely the sequence in its original position was not aligned on a level boundary!

Table 1 shows that our new technique (MOS) outperformed the traditional GA at every opportunity independent of whether a steady-state or generational GA was used. For example, for the L25 data set, the steady-state converged at 24.8 for traditional and 22.6 for MOS, and the generational resulted in 22.3 for traditional and 20.3 for MOS. The 20.3 obtained for the generational MOS was 1.015 times the optimal of 20. For the R25 data set, the steady-state converged at 29.8 for traditional and 28.9 for MOS, and the generational resulted in 27.7 for traditional and 26.4 for MOS. The estimated optimal packing heights for the random data sets were computed by dividing the sum total of the areas of the objects by the width of the bin. The 26.4 obtained for the generational MOS was 1.069 times the estimated optimal of 24.7. Notice the generational genetic algorithm outperformed the steady-state in every instance. It has been our experience that this is the case for most order-based problems that we have worked on. This may be best explained by considering the diversity in the population for each model. The generational model selects parents from one pool and places children in another. When enough children have been generated, the children replace the parents. This model tends to converge slowly, since it is difficult for the highly-fit members to dominate the population quickly. In the steady-state model, parents and children share the same pool. Consequently, highly-fit members can quickly dominate the population and cause the genetic algorithm to converge more quickly. The important point here is not that the steady-state algorithm performed worse, but that the heuristic was able to improve the performance of the genetic algorithm under both models. Note that in all cases, the results for the generational genetic algorithm with MOS were much better than the bounds for even the one dimensional bin packing problem. Also, in all cases, the genetic algorithm's results were obtained within three minutes.

Masks and Bin Packing

Recall the move optimal sequence (MOS) strategy that was used in bin packing can be implemented using a mask. In this special case of the mask, all of the set bits are adjacent and lie on the leftmost portion of the chromosome. This concept can also be used with other crossover methods. For example, uniform order crossover (UOX) uses a randomly generated mask to select genes for crossover. UOX/MASK is defined as the logical OR of a UOX random mask and the user defined mask. In this case, UOX/MASK uses MOS to set the user defined mask.

In the case of bin packing, the performance of UOX was improved by combining UOX/MASK and MOS. Table 2 shows the results of running a GA with UOX versus UOX/MASK. These results are similar to the results of Table 1: the preservation of optimal sequences yielded better results in every instance. Also, the generational GA yielded better results than the steady-state GA. One surprising result occurred for the L100 data set with the generational GA. While asexual crossover with MOS remains the most successful crossover operator, UOX/MASK outperformed it for this data set UOX/MASK gave 90.6 (see Table 2), while asexual with MOS gave 91.1 (see Table 1). Asexual did outperform UOX/MASK for the other data sets, yet, preservation of optimal sequences narrowed the gap.

CONCLUSIONS

In this paper we have introduced the concept of sequences and how they are ignored and even disrupted by traditional GAs. We introduced the sliding window,

Data Set	Steady-State UOX	Steady-State UOX/MASK	Generational UOX	Generational UOX/MASK	Optimal	Best *OPT
L25	26.1	24.5	23.4	20.7	20	1.035
L75	82.8	78.4	77.6	68.6	60	1.143
L100	113.9	108	102.1	90.6	80	1.133
R25	31	30	28.9	28.4	24.7	1.150
R50	59.7	57.7	55.2	52.3	44.8	1.167
R100	118.2	111.5	107.1	95.4	82.6	1.155

Table 2: Bin packing results using UOX versus UOX/MASK

and described how it could be used to identify highly-fit sequences. We showed that when these sequences are optimal that the chromosome could be reduced and the sequences preserved by moving them to the front of the chromosome. We also introduced a more general method of preserving highly-fit genes through the use of a mask. When the bonds between genes are important, we showed how relative bonding strength could be used to preserve these bonds. Finally, we showed several instances where these concepts were applied with great success. Most research to date attempts to minimize the negative aspects of disruption. This research emphasizes the positive aspect of identifying and manipulating optimal sequences. In the future, we will continue to develop and apply these methods. In particular, more work needs to be done on problems where the probability of encountering highly-fit schemata is low.

Acknowledgements

This research has been supported by OCAST Grant AR2-004. The authors would like to thank Referee No. 1 for his thorough reading of the manuscript, which resulted in numerous improvements and clarifications. We gratefully acknowledge the support of Sun Microsystems, Inc.

References

[1] J. L. Blanton and R. L. Wainwright. Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms. In Stephanie Forrest, Editor, *Proceedings of the Fifth International Conference on Genetic Algorithms* (ICGA-93), Morgan Kaufmann Publisher, San Mateo, California, 1993, pages 452–459.

[2] E. G. Coffman, M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance Bounds for Level-oriented Two-dimensional Packing Algorithms. *SIAM Journal of Computing*, 9(4):808–826, November 1980.

[3] A. L. Corcoran and R. L. Wainwright. A Genetic Algorithm for Packing in Three Dimensions. In *Proceedings of the ACM/SIGAPP Symposium on Applied Computing*, pages 1021–1030, Kansas City, Missouri, March 1992.

[4] A. L. Corcoran and R. L. Wainwright. LibGA: A User-friendly Workbench for Order-based Genetic Algorithm Research. In *Proceedings of the ACM/SIGAPP Symposium on Applied Computing*, pages 111–118, Indianapolis, Indiana, February 1993.

[5] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[6] Stephanie Forrest and Melanie Mitchell. Relative Building-block Fitness and the Building-block Hypothesis. In D. Whitley, editor, *Foundations of Genetic Algorithms 2.* Morgan Kaufman, San Mateo, California, 1993.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989.

[8] John H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, Michigan, 1975.

[9] L. Knight and R. Wainwright. HYPERGEN – A Distributed Genetic Algorithm on a Hypercube. In *Proceedings of the Scalable High Performance Computing Conference*, Williamsburg, Virginia, April 1992.

[10] K. Li and K. H. Cheng. On Three-dimensional Packing. *SIAM Journal of Computing*, 19(5):847–867, October 1990.

[11] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A Comparison of Genetic Sequencing Operators. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76, San Diego, California, June 1991.

[12] D. Whitley and J. Kauth. GENITOR: A Different Genetic Algorithm. In *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, pages 118–130, Denver, Colorado, 1988.