Combination and Recombination in Genetic Algorithms

Richard A. Watson Jordan B. Pollack

Dynamical and Evolutionary Machine Organization Group
Volen Center for Complex Systems – Brandeis University – Waltham, MA – USA
richardw@cs.brandeis.edu

Abstract. Recombination is supposed to enable the component characteristics from two parents to be extracted and then reassembled in different combinations – hopefully producing an offspring that has the good characteristics of both parents. This can work only if it is possible to identify which parts of each parent should be extracted. Crossover in the standard GA, for example, takes subsets of genes that are adjacent on the genome. Other variations of the GA propose more sophisticated methods for identifying good subsets of genes within an individual. Our approach is different; rather than devising methods to enable successful extraction of gene subsets from the parents, we utilize variable size individuals which represent subsets of genes from the outset. By allowing each individual to represent a building-block explicitly, the normal action of selection can identify good building-blocks without also promoting garbage genes. Then putting together two individuals (creating an offspring that is twice the size), straight forwardly produces the sum of the parents characteristics. This process is more properly combination than recombination since these building blocks have not been previously combined with any other. This paper summarizes our research on this approach and describes improved methods that reduce the domain knowledge required for successful application.

1 Introduction

Divide and conquer techniques in problem solving are familiar and intuitive; first find the solution to sub-problems and then re-use these to find solutions to the whole problem. Dynamic programming, for example, offers exponential speed-up over linear methods by caching sub-solutions and re-using them to evaluate whole solutions. The original motive of crossover in the building-block hypothesis [Goldberg 1989] appeals to the same intuition. But whereas dynamic programming requires the storage of all intermediate results, the GA stores a finite population selected by a heuristic – the fitness function. If recombination in the GA can be made to work, the same kind of speed-up can be expected over mutation-based algorithms that cannot re-combine sub-solutions.

Watson and Pollack [2000a] describes a formal building-block problem (which we will also use in this paper), for which mutation based algorithms cannot be guaranteed to succeed in less than time exponential in N (the size of the problem), whereas an idealized recombinative algorithm has an upper bound in time to solution of $N\log^2 N$. In

practice, a GA with recombination succeeds in less than $N^2\log^2 N$. This problem was first introduced in Watson et al [1998] where we showed that the success of the GA was dependent on tight genetic linkage – that is, the bits forming a building-block must be close together on the genome – as assumed in our analytic solution times.

In Watson & Pollack [1999a] we started investigation into solving the poor linkage version of the problem where the order of genes is randomized, and crossover is unable to extract and recombine good building blocks. In that paper we introduced an alternative algorithm, the Incremental Commitment GA (ICGA), based on the feature of underspecification found in the Messy GA [Goldberg et al 1989]. In the Messy GA individuals are initialized to specify exactly k bits, where k is "the size of the highest-order non-linearity expected in the problem". This enables an individual to represent exactly one building-block explicitly, whereas in the standard GA an individual may represent many building-blocks but each one is, at best, implicitly identified by the proximity of genes on the genome. We showed that the feature of underspecification is sufficient to enable success on a poor-linkage problem and that the other features of the Messy GA, in particular the moving-locus features (described below), are not required.

However, our ICGA was not without its own complications. There were two specific problems. First, the approach requires a diversity maintenance technique to keep the population from converging. As in earlier work with the standard GA, we used a resource-based fitness sharing method that utilized considerable knowledge of the problem structure. Specifically, it maintained a 'resource level' for each building block in the problem. Second, the approach requires a method for preventing strings from growing in length too quickly. We used a size-penalty augmentation to the fitness function but this was complicated by the distorted fitness values given by the fitness sharing mechanism.

In this paper we report some progress in these respects. First we report that an off-the-shelf diversity maintenance technique, deterministic crowding [Mahfoud 1995], works for our problem. This technique does not require any knowledge of the problem's internal structure. Second, this method simplifies our size-penalty function and does not require us to modify the fitness contributions of building-blocks. These improvements bring us a little closer to a general algorithm based on combination but, as we shall discuss, there are other issues yet to be resolved.

The remainder of this paper is organized as follows: Section 2 discusses recombination with respect to the problem of identifying gene subsets in problems of poor linkage. Section 3 summarizes the characteristics of our test problem. Section 4 describes our use of deterministic crowding and discusses why this method is appropriate for problems of this class. Section 5 gives results using deterministic crowding for the standard GA and the ICGA, on both the regular test problem and the problem with random genetic linkage. We discuss the trade-offs between recombination and combination, and Section 6 concludes.

2 Recombination and poor linkage

Crossover in the standard GA [Holland 1975] relies on the heuristic of bit adjacency to identify appropriate subsets of genes during recombination. Clearly, we cannot in general rely on this heuristic, and when the adjacency of the genes is not correlated with gene interdependency, the subset of genes extracted from a parent by crossover is

unlikely to contain a meaningful component of the parent's abilities [Altenberg 1995]. The Messy GA [Goldberg et al 1989] and the Linkage Learning GA [Harik & Goldberg 1996] use a *moving locus* representation of genes – each gene is represented by a locus/allele pair. This enables genes to be re-ordered on the genome and potentially allows interdependent genes to collect together. Thus, it is reasoned, subsequent crossover operations can extract meaningful subsets of genes. Other algorithms, such as the Gene Expression GA [Kargupta 1997] and Selective Recombination [Vekaria & Clack 1998], use more sophisticated methods for identifying subsets of interdependent genes. Other research abandons the idea of recombination as described by the Building Block Hypothesis, and instead advocates the use of mutation only algorithms, or variants of 'recombination' such as uniform crossover [Syswerda 1989, Radcliffe 1991] that amplify the similarities between individuals rather than combining differences [Chen 1999].

Our research pursues the strong form of recombination described by the Building Block Hypothesis, and we have shown that, for at least one class of problem, recombination combining differences is necessary and sufficient for successful operation of the GA [Watson & Pollack 2000b]. We also showed that the progress of search under recombination is provably superior to mutation-only alternatives (if appropriate diversity is maintained in the population) [Watson & Pollack 2000a]. These results depend on the validity of the assumption of tight linkage. But our research continues to pursue a better understanding of GA recombination by addressing problems of poor genetic linkage [Watson & Pollack 1999a, and this paper] where the assumption of gene adjacency does not hold and the standard GA fails.

Our approach to this problem takes inspiration from one feature of the Messy GA. This is the feature of underspecification – individuals that specify only a subset of genes. The interesting property of this approach is that individuals represent building-blocks explicitly; and it is the normal operation of selection in the GA, operating on these sub-strings, that permits the successful identification of good building-blocks. In contrast, fully-specified individuals, as used in the standard GA may contain good building blocks but selection acting on these individuals will also promote garbage genes riding on the same string (see "parasites" [Goldberg et al 1989], and "hitch-hikers" [Forrest & Mitchell 1993], [Vekaria & Clack 1999]). Consequently, the crossover operator must, one way or the other, use internal structure such as gene order (standard GA, Messy GA, Linkage Learning GA), or additional data structures (Gene expression GA, Selective Recombination), to express which subsets of genes within a parent represent good building blocks and to exclude garbage genes from recombination events. Figure 1 summarizes our discussion.

```
A: 01010011
B: 10000110
---------------
C: 11010011
```

Figure 1a: Parents, A and B, each contain a useful subset of genes (three genes per parent, shown in bold). The desired offspring, C, should take the good genes from both parents as shown. But simple crossover cannot achieve this.

```
A: -1---0-1
B: 1-0-0---
-----------
C: 110-00-1
```

Figure 1b: Combination of fixed-locus partially-specified individuals is independent of gene position. Here we represent unspecified genes, or don't cares, by "-" and the offspring is created by taking specified genes from either parent where available.

```
A: -1--00-1
B: 100-0--0
-----------
C: 110-00-1
```

Figure 1c: Where conflicts in specified genes occur we resolve all conflicts in favor of one parent. In this example we allow the first individual to dominate. We call this operator 'dominant splice' [Watson & Pollack 1999a].[1]

In our previous work [Watson & Pollack 1999a] we verified that recombination using the (fixed-locus) recombination operator in Figure 1c can enable successful combination of building blocks on a problem with random genetic linkage. In this paper we continue to investigate whether our approach can be made more practical by reducing the problem dependent aspects of our earlier experiments.

3 Hierarchical if-and-only-if (HIFF) and Shuffled H-IFF (SHIFF)

To investigate the principles of recombination and the identification of building-blocks, it is necessary to use a test problem where the building-blocks are clear. We first introduced our test problem 'Hierarchical if-and-only-if' (HIFF) in [Watson & Pollack 1998]. Previous building-block problems in the literature (e.g. Royal Roads [Mitchell et al 1992] and concatenated trap functions [Deb & Goldberg 1992]) did not model interdependency between building-blocks. The separable nature of these problems means that each sub-problem can be solved independently and serially, and certain varieties of mutation-based hill-climbers show that a GA is not required for this type of problem [Mitchell and Forrest 1993], [Jones 1995].

HIFF is not a separable problem – the building-blocks have strong non-linear interdependencies. Although each building-block is identifiable via its fitness contribution, a successful algorithm must maintain competing solutions for each block and search combinations of blocks to find complete solutions. This discovery and combination process continues through a hierarchical structure which is consistent in the nature of the problem at each level [Watson and Pollack 1999b]. Equation 1, below, gives the fitness of a string B under the canonical version of HIFF.[2] This function interprets a string as a binary tree and recursively decomposes the string into left and right halves. Each block at each level in the hierarchy has two solutions – all ones and all zeros – and the function has two corresponding global optima.

---

[1] In the current work we drop the stipulation that the dominant parent should be the most fit.

[2] That is, where blocks are defined over a binary string, and blocks are assembled from pairs of sub-blocks. Other variants that use variable sized alphabets, variable numbers of sub-blocks per block, and uneven fitness contributions, (amongst others), are defined in [Watson & Pollack 1999b].

$$F(B)=\begin{cases} 1, & \text{if } |B|=1, \text{ and } (b_i=0 \text{ or } b_i=1) \\ |B| + F(B_L) + F(B_R), & \text{if } (|B|>1) \text{ and } (\forall i\{b_i=0\} \text{ or } \forall i\{b_i=1\}) \\ F(B_L) + F(B_R), & \text{otherwise.} \end{cases} \quad \textbf{Eq.1}$$

where B is a block of bits, $\{b_1,b_2,...b_k\}$, $|B|$ is the size of the block=k, b_i is the ith element of B, B_L and B_R are the left and right halves of B (i.e. $B_L=\{b_1,...b_{k/2}\}$, $B_R=\{b_{k/2+1},...b_k\}$). The length of the string evaluated must equal 2^p where p is an integer (the number of hierarchical levels). Notice that this function gives no reward to nulls and therefore naturally evaluates partially specified strings.

This equation states the fitness of a string assuming tight genetic linkage i.e. two bits that form a block at the lowest level are adjacent on the genome, and two size-2 blocks that form a block at the next level are also adjacent, and so on. Unlike previous tight-linkage building-block problems HIFF is not solvable by any form of mutation-based hill-climber [Watson and Pollack 1998, Watson and Pollack 2000a]. However, with the proviso that diversity in the population is maintained appropriately, the standard GA succeeds easily. In this paper we confirm this result using deterministic crowding rather than our problem-dependent fitness-sharing method used in our earlier work.

Here, however, the focus is on *Shuffled HIFF*. In Shuffled H-IFF (SHIFF) the position of bits in the problem is randomly re-ordered. This means that the bits constituting a block may be anywhere on the string – but for any given instantiation of the problem the position is fixed. Shuffled HIFF is not solvable by the standard GA because it is not possible to extract meaningful subsets of bits from a parent using adjacency-dependent crossover (e.g. one or two-point crossover). In principle, uniform crossover could provide the crossover that we need but the probabilities are equivalent to random guessing at each locus where the parents disagree on allele values [Watson & Pollack 1999a]. It is also not solvable by the Messy GA or Gene Expression GA because both these methods assume an upper limit on the highest order non-linearity in the problem. In other words, they assume that the problem consists of separable building-blocks. The hierarchically consistent nature of HIFF means that blocks at all levels are strongly and non-linearly dependent on one another – whether a block should be all-ones or all-zeros depends on whether the neighboring block is based on ones or zeros.

However, the ICGA using partially specified individuals and a combination operator as described in Figure 1c can solve Shuffled HIFF. In the following section we discuss deterministic crowding as a diversity maintenance method for this algorithm. This method removes the considerable problem dependence used in our previous work on Shuffled HIFF.

4 Deterministic Crowding

Deterministic Crowding (DC) is a diversity maintenance technique developed by [Mahfoud 1995]. DC is naturally implemented in a steady state GA as described in Figure 2.

The key feature of the DC method is *restricted competition*. Competition between individuals is limited to individuals that are similar. Similarity is involved in two ways:

firstly, competing individuals are restricted to parent/offspring pairs so their genetic make-up is not arbitrary; Secondly, the pairing rule is designed explicitly to minimize the difference between offspring and potential replacees. This restricted competition means that the relative fitness of individuals does not matter unless they are similar enough to compete. We may imagine two sub-populations exploring different peaks in the fitness landscape: if one peak is higher than the other then using unrestricted competition the population will tend to converge on that peak and exploration of the second peak will be precluded. Using DC the two sub-populations will not compete. Note that for this to be the case the parents must be selected at random. If they are selected proportionate to fitness then the low-fitness sub-population, to follow our two-peak scenario, will not receive reproductive opportunities.

- Initialize population.
- Repeat until stopping condition:
  - Pick two parents at random from the population, p1 & p2.
  - Produce a pair of offspring using recombination, c1 & c2.
  - Pair-up each offspring with one parent according to the pairing rule below.
  - For each parent/offspring pair, if the offspring is fitter than the parent then replace the parent with the offspring.

  Pairing rule: if H(p1,c1)+H(p2,c2) <H(p1,c2)+H(p2,c1) then pair p1 with c1, and p2 with c2, else pair p1 with c2, and p2 with c1, where H gives the genotypic Hamming distance between two individuals.

Figure 2: A simple form of a GA using deterministic crowding as used in the following experiments.

Another notable feature of deterministic crowding is elitist replacement – offspring will only replace their parents if they are fitter. Actually, we relax this slightly, and allow offspring to replace parents if they are fitter or equal in fitness. This elitism is sensible given that we believe any competing individuals to be on the same peak – if this is so, we need not entertain exploratory individuals that are inferior.

It is important to realize that deterministic crowding does not use *restricted mating* as used in other diversity maintenance techniques (e.g. 'thresholding' [Goldberg et al 1989]). In DC, any individual may mate with any other regardless of their similarity or dissimilarity. This important distinction from other niching methods is ideal for recombination as described by the Building Block Hypothesis and for the HIFF problem. Since recombination is supposed to combine characteristics formerly not seen together, it makes little sense to restrict mating to similar individuals. A separate investigation shows that it is the combination of *dissimilarities* that is essential in this class of problem and that the common wisdom of recombining only *similar* individuals is misguided. The combination of differences is sufficient for finding solutions even in the somewhat perverse case where recombination is deliberately restricted so that only genes that *disagree* in the parents can be inherited [Watson & Pollack 2000b].

Deterministic crowding is thus an appropriate method for the ICGA; it segregates the competition between sub-populations to maintain different building-blocks, yet it still allows these building-blocks to be combined together.

5 Experiments

This section gives experimental results of the GA and ICGA using deterministic crowding described in Figure 2. The ICGA differs from the regular GA in that is uses the partially specified individuals and the dominant splice combination operator described in Figure 1c. We also need to employ a method to prevent strings from growing in size too quickly and thereby collecting garbage genes that would defeat our combination operator.

In the Messy GA, accumulation of garbage genes is prevented by using two distinct phases of operation: in the first phase, individuals are restricted to a fixed size and selection identifies good building-blocks. In the second phase, when all good building-blocks have been found, individuals grow in size without restriction. This method is only applicable when there are no higher-order linearities in the problem, as is the case in the test problems used for Messy GAs. Since HIFF has dependencies between blocks at all levels in the hierarchy we cannot use this technique. Instead we must use a method that allows blocks to be accumulated through many incremental stages. To this end, we employ a size-penalty augmentation to the fitness function. This enables the size of strings to grow gradually, only committing to gene alleles when these genes return significant fitness contributions, hence, Incremental Commitment GA.

In HIFF, the maximum fitness, MF, of a string of size N, is the product of N and the number of hierarchical levels in the string. i.e. $MF(N)=N(\log_2(N)+1)$. Accordingly, individuals in our algorithm will receive fitness $F'(B)=F(B)-MF(|B|)$. This ensures that good, small individuals are preferred over individuals that specify several sub-optimal building-blocks. Clearly, this method requires knowledge of how the fitness of strings grows with the size of strings (number of specified bits). But such knowledge says nothing of where the blocks are or what their solutions might be – there is still considerable work to be done by the algorithm in discovering and searching combinations of building-blocks.

To compare the *combination* of individuals in our Incremental Commitment GA using dominant splice (ICGA), with *recombination* found in the standard GA, we will also test a regular GA using deterministic crowding. The regular GA is tested with one-point crossover (GA one-point) and uniform crossover (GA uniform).[3] All three algorithms are applied to the Shuffled HIFF, and the GA one-point is also applied to the regular HIFF for comparison. Since the GA uniform and ICGA have no locus-dependent features, their performance is identical on the Shuffled HIFF and regular HIFF. In all cases, crossover/combination is applied with probability 0.7, and mutation is applied with 0.03 probability of assigning a new random allele (0, 1, null; with equal probability). In the ICGA algorithms, individuals are initialized to specify one random gene. A population size of 1000 is used in all cases. Data points are given every 1000 iterations of the crowding algorithm, which is every 2000 evaluations. Performance is measured by the fitness of the best individual (in the preceeding 2000 evaluations) averaged over 30 runs for each algorithm. The problem size is 64 bits which gives a maximum fitness of 448. To enable comparison, the fitness of individuals in the ICGA is measured before the size penalty is applied.

---

[3] Note that we cannot try the size-penalty augmentation with the regular GA since individuals are fully specified and therefore all have the same size.

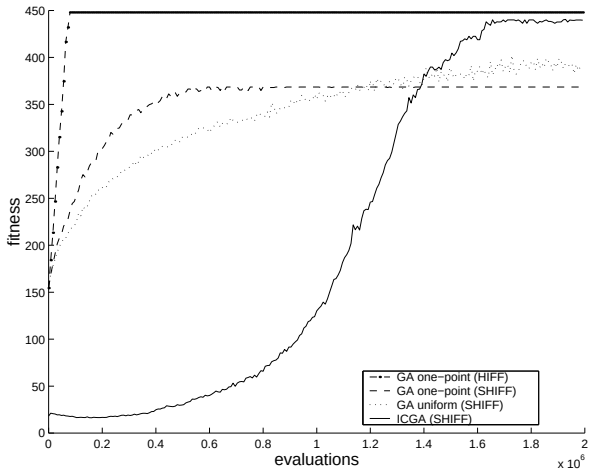

Figure 3: Performance of Incremental Commitment GA, and regular GA (using one-point and uniform crossover), on Shuffled HIFF. GA one-point is also shown on regular HIFF for comparison.

Figure 3 shows clearly that the regular GA using deterministic crowding and one-point crossover succeeds quickly on regular HIFF. However, the Shuffled HIFF prevents successful recombination using one-point crossover. Also, uniform crossover is unable to combine building-blocks reliably (regardless of shuffling). The macro-mutations [Jones 1995] induced by uniform crossover in a diverse population keep the fitness low at first. But eventually they enable better exploration.

Actually, the performance of uniform crossover is better than expected - it succeeds in solving the 64 bit problem in 16 of the 30 runs. It appears that the convergence-sensitive macro-mutation [Eshelman et al 1996, Watson and Pollack 2000b] enabled by uniform crossover, together with a technique like crowding to maintain the blocks discovered, is quite effective at exploring the search space. This is only the case when using deterministic crowding, large populations, and long runs, as in these experiments. Our previous research (comparing performance to the GA with one-point crossover on regular HIFF) did not notice that uniform crossover occasionally succeeded because uniform crossover takes about 12 times more evaluations to succeed (when it does). Note that uniform crossover cannot recombine building-blocks from two different parents – where parents disagree, gene values are effectively randomized [Radcliffe 1993] – and its exploration cannot be guided by information from lower-level blocks appropriately for this problem. Its exploration pressure is merely to go to places it has not already been. Accordingly, we see that its progress slows down as search continues.

This is in sharp contrast to the performance of the ICGA which is very slow to discover small building-blocks but ensues more rapidly as search continues. This unusual profile can be explained as follows: in the early stages the algorithm must search out which particular pairs of bits make a block. There are many size-2 blocks to be found and their reward is small. As the size of blocks discovered increases, the number of blocks, and the number of combinations to be searched decreases. Moreover, the fitness of larger blocks is greater. Thus the improvement in fitness speeds-up as search progresses through hierarchical levels.

The contrast of uniform crossover and the ICGA illustrate the trade-off between

recombination and combination. Uniform crossover is quick to discover reasonable solutions. An individual that contains a good building-block is promoted even if it also contains garbage on the rest of the string. Eventually, it finds individuals that contain many reasonably-sized building blocks but these building-blocks are not necessarily compatible. Uniform crossover (or one-point crossover on Shuffled HIFF) is subsequently unable to extract building-blocks from two different individuals and recombine them because there is no mechanism to identify which genes within an individual constitute a block. The partially specified individuals in the ICGA operate differently. They cannot accumulate garbage genes or sub-optimal blocks so their fitness grows slowly. But when blocks are discovered, they are represented explicitly; so as bigger blocks are discovered, they can be easily combined together. The ICGA pays the price in the early stages of search, so that the information contained in individuals has more utility in the later stages of search. Overall the trade-off in our test problem lies in favor of combination.

6 Discussion and Conclusions

The results above illustrate that combination of partially-specified individuals can provide an alternate method for successful building-block assembly in problems of poor genetic linkage. Other schemes attempt to extract meaningful components from parents that may also be carrying garbage genes. Our method uses the ordinary selection method of the GA to identify good subsets of genes by allowing individuals to represent subsets explicitly.

The approach of combination rather than recombination may cause us to think about representing problems differently when using an evolutionary algorithm. Consider the evolution of neural networks. It seems unlikely that we could expect recombination to take sub-networks from parents and recombine them in a way that is meaningful. However, it is not so hard to imagine the creation of small networks that provide useful behaviors, and that when these are assembled together to form larger networks, they provide an informed exploration of the larger search space. Note that the problem need not be separable for this to work, as the structure of HIFF illustrates.

We have not yet turned this approach into a practical algorithm. The main hurdle is the size-penalty which requires knowledge of how fitness increases with the size of individuals. However, this paper greatly reduces the required problem knowledge with respect to previous work. In the meantime, our experiments illustrate some important principles in (re)combination methods.

Acknowledgments

The first author is indebted to Martin Oates for tireless investigations of alternate diversity maintenance techniques, and to the Santa Fe Institute for providing the opportunity to discuss this work with like-minded researchers, in particular, Christopher Ronnewinkel who suggested that deterministic crowding might be the mechanism for which we were looking. Thanks also to the members of DEMO at Brandeis.

References

Altenberg, L, 1995 “The Schema Theorem and Price’s Theorem”, *FOGA3*, editors Whitley & Vose, pp 23-49, Morgan Kauffmann, San Francisco.

Chen, S, 1999, “*Is the Common Good? A New Perspective in Genetic Algorithms and Genetics*”, PhD diss., Robotics Institute, Carnegie Mellon University.

Deb, K & Goldberg, DE, 1989, “An investigation of Niche and Species Formation in genetic Function Optimization”, *ICGA3*, San Mateo, CA: Morgan Kauffman.

Deb, K & Goldberg, DE, 1992, “Sufficient conditions for deceptive and easy binary functions”, (IlliGAL Report No. 91009), University of Illinois, IL.

Eshelman, LJ, Mathias, KE, & Schaffer, JD, 1996, "Convergence Controlled Variation." In *Foundations of Genetic Algorithms 4*, Belew and Vose, eds. Morgan Kaufmann.

Forrest, S & Mitchell, M, 1993b “What makes a problem hard for a Genetic Algorithm? Some anomalous results and their explanation” *Machine Learning 13*, pp.285-319.

Goldberg, DE, 1989 “*Genetic Algorithms in Search, Optimisation and Machine Learning*”, Reading Massachusetts, Addison-Wesley.

Goldberg, DE, Deb, K, & Korb, B, 1989 “Messy Genetic Algorithms: Motivation, Analysis and first results”, *Complex Systems, 3*, 493-530.

Harik, GR, & Goldberg, DE, 1996, “Learning Linkage” in *FOGA 4*, Morgan Kaufmann, San Mateo, CA.

Holland, JH, 1975 “*Adaptation in Natural and Artificial Systems*”, Ann Arbor, MI: The University of Michigan Press.

Jones, T, 1995, *Evolutionary Algorithms, Fitness Landscapes and Search*, PhD dissertation, 95-05-048, University of New Mexico, Albuquerque. pp. 62-65.

Kargupta, H, 1997, “Gene Expression: The Missing Link In Evolutionary Computation” *Genetic Algorithms in Engineering and Computer Science*, Eds. Quagliarella, Q Periaux, J, and Winter, G.: John Wiley and Sons.

Mahfoud, S, 1995, “*Niching Nethods for Genetic Algorithms*”, PhD diss., Dept. General Engineering, University of Illinois. Also, IlliGAL Report No. 95001.

Mitchell, M, Forrest, S, & Holland, JH, 1992 “The royal road for genetic algorithms: Fitness landscapes and GA performance”, *Procs. of first ECAL*, Camb., MA. MIT Press.

Radcliffe, NJ. 1991, "Forma Analysis and Random respectful Recombination." *In Proc. Fourth International Conference on Genetic Algorithms.*

Syswerda, G, 1989, "Uniform Crossover in Genetic Algorithms." *In Proc. Third International Conference on Genetic Algorithms.*

Vekaria, K, & Clack, C, 1998, “Selective Crossover in Genetic Algorithms: An Empirical Study”. In Eiben et al. (eds). *Proceedings of the 5th Conference on Parallel Problem Solving from Nature*, number 1498 in Lecture Notes in Computer Science, 438-447. Springer-Verlag.

Vekaria, K, & Clack, C, 1999, “Hitchhikers Get Around”. To appear in: *Artificial Evolution (EA) 1999*, November 3-5, LIL, Universite du Littoral, Dunkerque, France.

Watson, RA, Hornby, GS & Pollack, JB, 1998, “Modeling Building-Block Interdependency”, *Parallel Problem Solving from Nature, proceedings of Fifth International Conference /PPSN V*, Springer 1998, pp.97-106 .

Watson, RA, & Pollack, JB, 1999a, “Incremental Commitment in Genetic Algorithms”, *Proceedings of 1999 Genetic and Evolutionary Computation Conference (GECCO 99).* Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, Smith, eds., Morgan Kauffmann, 710-717.

Watson, RA, & Pollack, JB, 1999b, “Hierarchically-Consistent Test Problems for Genetic Algorithms”, *Proceedings of 1999 Congress on Evolutionary Computation (CEC 99).* Angeline, Michalewicz, Schoenauer, Yao, Zalzala, eds. IEEE Press, pp.1406-1413.

Watson, RA, & Pollack, JB, 2000a, “Analysis of Recombinative Algorithms on a Hierarchical Building-Block Problem”, *Foundations of Genetic Algorithms (FOGA 2000)*, submitted.

Watson, RA, & Pollack, JB, 2000b, “Recombination Without Respect”, *Proceedings of 2000 Genetic and Evolutionary Computation Conference (GECCO 2000)*, submitted.