Symbiotic Combination as an Alternative to Sexual Recombination in Genetic Algorithms

Richard A. Watson Jordan B. Pollack

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

Abstract. Recombination in the Genetic Algorithm (GA) 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. However, this can only work if it is possible to identify which parts of each parent should be extracted. Crossover in the standard GA 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 parents, we utilize variable-size individuals which represent subsets of genes from the outset. Joining together two individuals, creating an 'offspring' that is twice the size, straight-forwardly produces the sum of the parents' characteristics. This form of component assembly is more closely analogous to combination of symbiotic organisms than it is to sexual recombination. Whereas sexual recombination, modeled by crossover, occurs between similar individuals and exchanges subsets of genes, symbiotic combination, as modeled in our operator, can occur between entirely unrelated species and combines together whole organisms. This paper summarizes our research on this approach to recombination in GAs and describes new methods that illustrate its potential.

1 Introduction

The natural process of *symbiogenesis* [Merezhkovsky 1909] is the creation of new species from the genetic integration of symbionts, organisms in symbiotic relationship. Symbiogenesis has enabled some of the major transitions in evolution [Maynard-Smith and Szathmary 1995], including the origin of eukaryotes which include all plants and animals. This kind of genetic integration is quite different from the transfer of genetic information in sexual reproduction. Sexual recombination occurs between similar organisms (i.e. of the same species) and involves the exchange of parts of the genome in a mutually exclusive manner. That is, every gene acquired from one parent is a gene that cannot be acquired from the other parent. In contrast, symbiotic combination can occur between genetically unrelated organisms (i.e. different species) and can involve the integration of whole genomes. The resultant composite may have all the genes from one organism and at the same time acquire any number of genes from the second organism.

Our research has been investigating whether this form of genetic combination has any potential to inspire an alternate form of evolutionary algorithm. This has first required us to address some old arguments about the utility of ordinary crossover in the standard GA

[Holland 1975]. To this end, we developed a hierarchical building-block problem that exemplifies the utility of sexual recombination as demonstrated by crossover [Watson et al 1998, Watson & Pollack 1999b, Watson 2000]. Moreover, our experiments also illustrated a well-known limitation of crossover – that is, it 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 parents' abilities [Altenberg 1995]. Figure 1 illustrates the problem of crossover in a *poor-linkage* problem - i.e. where the bits of a block (a set of interdependent genes) are distributed.

A: 0**1**010**0**1**1**
B: **1**0**0**0**0**110
C: **110**1**00**1**1**

Figure 1: Sexual recombination (crossover). Parents, A and B, each contain a useful subset of genes that are distributed on the genome (three genes per parent, shown in bold). The desired offspring, C, should take the good genes from both parents as shown. One-point or n-point crossover cannot do this. In principle, uniform crossover could provide the crossover that we need but the probability of this is equivalent to random guessing at each locus where parents disagree on allele values [Watson & Pollack 1999a].

Thus we see that building-blocks in the standard GA are, at best, only represented *implicitly* by virtue of the proximity of genes on the genome. In poor-linkage problems we need some *explicit* mechanism to represent the unity of bits comprising a block. This is where a mechanism based on symbiotic combination might be useful. Put simply, if each organism represents just one building-block, then a combination operator can put different organisms together straight-forwardly producing an organism twice the size with both building-blocks intact. A simple representation for an individual in this scheme utilizes 'unspecified' genes - then symbiotic combination can be described as a genetic operator acting on partially-specified individuals (Figure 2).

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

Figure 2: 'Symbiotic combination'. Left) Combination of partially-specified individuals produces an offspring that is twice the size of the parents with the sum of their characteristics. Here we represent unspecified genes, or don't cares, by "-". Each parent represents a single building-block explicitly and the offspring is created by taking specified genes from either parent where available. Right) Where conflicts in specified genes occur we resolve all conflicts in favour of one parent, e.g. the first parent.

Previous work [Watson & Pollack 1999a, 2000a], implemented the 'Incremental Commitment GA', ICGA, using this operator. The ICGA can be described as a simplified and generalized version of the Messy GA [Goldberg et al 1989], which also uses combination of partially specified individuals. But our implementation of the ICGA had important limitations, as we shall discuss. In this paper we introduce a much richer model of symbiotic combination: The Symbiogenic Evolutionary Adaptation Model, SEAM. In addition to symbiotic combination, as used in the ICGA, the key features of SEAM are group evaluation of individuals which removes the need for partial evaluation, and a

'Pareto coevolution' method that segregates competition to maintain diversity and prevent large sub-optimal individuals from replacing small optimal individuals. These new features overcome the limitations of the earlier ICGA and facilitate the application of symbiotic combination more effectively.

The remainder of this paper is organized as follows: Section 2 summarizes relevant background work. Section 3 details our Symbiogenic Evolutionary Adaptation Model. Section 4 gives results for the standard GA, ICGA and SEAM on the hierarchical building-block problem with random linkage. Section 5 concludes.

2 Background: Approaches to Building-Block Combination

Our previous work describes a formal building-block problem that exemplifies the class of problems for which recombinative algorithms like the GA are well-suited. This problem, which we call Hierarchical If-and-Only-If (HIFF), was first introduced in [Watson et al 1998] (Equation 1). 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. 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]. On this class of problem, mutation based algorithms cannot be guaranteed to succeed in less than time exponential in N (the size of the problem in bits), whereas an idealized recombinative algorithm has an upper bound in time to solution of order $N\lg^2 N$ [Watson 2000]. However, Equation 1 states the fitness of a string assuming tight 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. The success of the regular GA is dependent on tight genetic linkage – and the focus of our ongoing research is on solving the problem with randomized linkage.

$$F(B)=\begin{cases} 1, & \text{if } |B|=1, \text{ and } (b_1=0 \text{ or } b_1=1) \\ |B| + F(B_L) + F(B_R), & \text{if } (|B|>1) \text{ and } (\forall i\{b_i=0\} \text{ or } \forall i\{b_i=1\}) \\ F(B_L) + F(B_R), & \text{otherwise.} \end{cases} \qquad \textbf{Eq.1}$$

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.

In [Watson & Pollack 1999a] we started investigation into solving *Shuffled HIFF*, (SHIFF) where the position of bits in the problem is randomly re-ordered. One approach to address problems of poor linkage is seen in the Messy GA [Goldberg et al 1989] and the Linkage Learning GA [Harik & Goldberg 1996] which 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. However, there is one feature of the Messy GA that is quite simple, and potentially effective, yet is generally under-emphasized. This is the feature of *underspecification* – individuals that specify only a subset of genes.

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, a crossover operator using fully-specified individuals must, one way or another, express which subsets of genes represent good schemata to exclude garbage genes from recombination events. The interesting property of approaches using partially-specified individuals is that individuals represent schemata 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 the problem class we address, large building-blocks at higher levels of the hierarchy produce significant high-order interdependencies. For a multi-level building-block problem like this we must use a method of gradually increasing specification that allows blocks to be accumulated through many incremental stages (unlike the two-phase method of underspecification seen in the Messy GA). To enable this, previous work employed 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 [Watson & Pollack 1999a]. Using the ICGA 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 which are often the focus of related work, are not required.

However, our ICGA was not without its own complications. The first, is that 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. Since then we have found an off-the-shelf diversity maintenance technique, deterministic crowding [Mahfoud 1995], that works very well for our problem [Watson & Pollack 2000a]. Deterministic crowding, DC, is naturally implemented in a steady state GA as described in Figure 3.

```
• 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 3: Pseudo-code for a simple form of a GA using deterministic crowding.

The most important feature of DC is that offspring only compete with their own parents. Additionally, offspring and potential replacees are paired to maximize their similarity. This segregated competition introduces tolerance for lower-fitness individuals in different niches and reduces the pressure for convergence. Note that DC does not segregate mating like other diversity maintenance techniques, e.g. ‘thresholding’ [Goldberg et al 1989]. In DC, an individual may *mate* with any other regardless of their similarity or dissimilarity, but it only *competes* with similar individuals.

The advantage of DC with respect to our previous method of diversity maintenance is that it does not require any knowledge of the problem’s building-block structure. It also alleviates complications in the size-penalty that arose from the distorted fitness values given by the fitness sharing mechanism. However, the ICGA still needs a size-penalty that requires knowledge of how fitness is expected to grow with string length. Moreover, the ICGA also depends on being able to evaluate partially specified strings. These are handled quite naturally in our test problem but, in general, an objective function may not be able to evaluate a string that is not fully-specified.

Goldberg et al [1989] suggest that one way to overcome the need for partial evaluation is to use *competitive templates*, bit-strings that are used to fill-in the unspecified bits of an individual. The template provides a context in which the partial individual can be evaluated. However, Goldberg et al also show that the use of random strings as templates would produce too much ‘background noise’ to identify the relatively small fitness contribution of a low-order schema represented by an individual. And they also argue that the use of a single random template used for all evaluations (reducing the influence of random noise) cannot evaluate a schema in an appropriately diverse range of contexts to assess its proper value. They propose that one way forward is to use a ‘locally optimized’ template provided by some other search method. However, this approach assumes that the “highest order non-linearity expected in the problem” is bounded and, in fact, quite low-order, as Goldberg et al assume. In this case, an appropriately optimized template is quite easy to find. But, in problems with strong high-order interdependencies, like the class we address, the task of providing an appropriately optimized template is only one step easier than solving the whole problem. Nonetheless, a form of templating will be useful in solving the problem of partial evaluation in our new algorithm, SEAM.

3 The Symbiogenic Evolutionary Adaptation Model, SEAM

This section introduces the ‘Symbiogenic Evolutionary Adaptation Model’, SEAM, which utilizes three main ideas. First, as in the ICGA, SEAM uses symbiotic combination that combines *whole* small organisms, rather than sexual recombination that recombines *parts* of fully-specified organisms. Second, we use groups of other individuals from the population to provide the templates and preclude the need for partial evaluation. And the third new component of SEAM is to use what we call ‘Pareto coevolution’ which segregates competition to maintain diversity, and prevents large sub-optimal strings from replacing small optimal strings (removing the need for a size-penalty).

Group templating

The use of other organisms to provide templates is inspired by the co-adaptation of symbiotic organisms in an ecosystem. We think of the templates as different environmental *contexts,* or niches, provided by different combinations of neighboring organisms. Algorithmically, the templates test the performance of a given schema in the context of many other different schemata provided by other optimized individuals. In this way we do not need to use a different search technique to provide the templates as Goldberg et al propose – rather, the organisms that are used as templates, and the organisms that use the templates, are all created by the same unified process. As more fit large individuals are discovered by the algorithm, they provide better templates for discovering individuals for the next hierarchical level in the problem. This use of individuals that are co-adapted to fill-in for one-another provides effective templates and precludes the need for partial evaluation in the algorithm. Group evaluation is used in a couple of existing algorithms, [Moriarty 1997, Potter 1997], and we have also investigated related effects of group evaluation ourselves [Watson & Pollack 1999c]. But the technique has not been connected to the use of templates in the Messy GA, nor has it been integrated with genetic operators that combine organisms together permanently.

Naturally, this method of templating will return different fitness scores depending on which individuals are chosen for the template. As Goldberg et al caution, an accurate measure of fitness for the individual in question might require prohibitively many trials. To alleviate the background noise of a template, individuals in SEAM are assessed in pair-wise competitions. That is, two individuals, *A* and *B*, are evaluated using the same additional individuals to provide the template/context for evaluation (Figure 4).

```
A: -11---00           B: --1010--
1: -0--1-0-           1: -0--1-0-
2: 10-0-11-           2: 10-0-11-
3: 1-1--1--           3: 1-1--1--
4: --0-01-1           4: --0-01-1
-----------------------------------
A′:11101100           B′:10101001
```

Figure 4: Left) A given partially-specified individual, *A*, is evaluated by building a template from several other partially-specified individuals, 1 though 4. Specified genes are provided by *A* where available, and unspecified positions are filled-in with genes from 1, and so on through 4, using additional individuals until all genes are specified. The resultant string *A'*, is the string evaluated for *A*. Right) The same individuals 1 through 4 are used to evaluate a second individual *B*. The difference in fitness between *A'* and *B'* indicates which is better in this context.

Note that any given context may favor *A* more than *B*, for example, depending on whether *A* happens to be better adapted to that particular context or not. So we will still need to perform evaluations in many different contexts to determine the superiority of *A* and *B*. However, this will not require prohibitively many evaluations.

Pareto coevolution

The group evaluation used in templating makes SEAM a coevolutionary system – the task of being a successful organism is dependent on the composition of the population. The normal coevolutionary procedure is to average performance over many trials, in this case, many different contexts. However, preliminary investigations using averaged scores caused convergence and failure of the algorithm. This problem has prevented us from progressing our model of symbiotic combination with group templating for some time. Here we introduce a new method of coevolution that we can use in SEAM to overcome this problem. This new method, which we call 'Pareto coevolution', incorporates ideas from Pareto optimization methods that are well-established for optimization in problems with multiple objectives [e.g. Horn 1997]. Pareto optimization recognizes that performance over different objectives, say 'financial cost' and 'construction time', cannot be combined to give an overall performance unless we know how to convert one ‘dimension’ to another – in this case, we need to know what our time is worth. Pareto optimization techniques may be used when the relative weighting of different dimensions is not known. Specifically, Pareto optimization is built on the principle of *Pareto domination*. A solution is said to Pareto dominate another solution if it is superior or equal in all dimensions (and superior in at least one dimension).

The idea behind Pareto coevolution is to use different coevolving opponents as the dimensions for determining dominance. Specifically, an individual dominates some other individual if it performs no worse than that individual against each and every opponent. In this manner, the performance of an individual will be assessed on the basis of which particular opponents it does well against and not just an average score. This allows individuals to adapt to different sets of opponents and promotes diversity.

Pareto coevolution is simply the application of this form of dominance in any Pareto optimization technique. We suggest that Pareto coevolution may be valuable in a variety of existing coevolutionary games and with a variety of Pareto optimization methods. But in the next subsection we show how to apply the technique to our specific domain of function optimization via group evaluation, and how to implement a very simple form of Pareto optimizer that is sufficient for our needs.

Integrating the features of SEAM

In SEAM the coevolution is subtle – the result of an evaluation depends on the other members of the population selected for the template – but there is no overt opponent in this setup. Nonetheless, we can use contextual groups as dimensions to determine domination and apply Pareto coevolution. SEAM uses this principle directly in determining the outcome of the pair-wise competition illustrated in Figure 4. Specifically, *A* dominates *B*, if it is superior or equal to *B* in all groups tested. Figure 5 shows how to utilize this rule in a very simple Pareto optimizer.

```
•  Initialize population to random single-bit individuals.
•  Repeat until stopping condition:
     •  Pick two parents at random from the population.
     •  Produce an offspring using symbiotic combination.
     •  If the offspring dominates both parents (see below) then replace the parents
        with the offspring.
   To determine whether A dominates B: Repeat for t trials:
          •  Build a complete template from randomly selected individuals.
          •  If B superimposed on this template receives a higher score than A
             superimposed on this template then A does not dominate B.
```

Figure 5: Pseudo-code for a simple implementation of SEAM.

Segregating competition by the use of contextual niches in SEAM maintains appropriate diversity without the need to use the genotypic similarity metric usually used in the pairing-rule of deterministic crowding. More importantly, these methods preclude the need to use a size-penalty function. The size-penalty was used in the ICGA to prevent organisms from prematurely filling-up with incompatible blocks. In SEAM, if just one of the contexts tested for the parent includes a *compatible* block, then an offspring formed by joining with an *incompatible* block in that position will be rejected. By insisting that a composite must perform as well as the parents in all contexts, we prevent incompatible blocks from being joined, and individuals being filled with sub-optimal schemata.

The algorithm in Figure 5 differs considerably from established Pareto optimization methods: specifically, like deterministic crowding, an offspring only survives through competition with its parents. This restriction is applied for efficiency – the offspring is intended to combine the characteristics of the parents so, if this combination is successful, the offspring should be at least as good as either parent in any context. This method proves sufficient for our problem class but preliminary investigations suggest that a more conventional Pareto optimization approach may broaden the applicability of SEAM. Additionally, our implementation of this algorithm also reduces computational expense by removing duplicates from the population. So, when we replace the parents we insert only one copy of the offspring, reducing the size of the population by one with each offspring that is successful. At present, duplicates in the initial population are identified by genetic comparison but, in principle, we can test for identity in their contextual performance.

4 Experiments

This section gives experimental results of the GA, ICGA, and SEAM applied to a 64-bit Shuffled HIFF. The GA is implemented using the deterministic crowding algorithm of Figure 3, and is tested using uniform crossover (GA-uniform) and one-point crossover (GA-onepoint). A population size of 1000 is used; crossover is applied with probability 0.7; and mutation is applied with 0.03 probability of assigning a new random allele (0 or 1, with equal probability). The ICGA differs from the GA in three respects: 1) it uses partially specified individuals, initialized to one random bit, and mutation assigns 0, 1, or null, with equal probability; 2) it uses the combination operator described in Figure 2, and; 3) it uses a size-penalty augmentation to the fitness function. 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 the ICGA receive fitness $F'(B)=F(B)-MF(|B|)$. SEAM uses the algorithm outlined in Figure 5. A population size of 1000 is used in initializing SEAM but the removal of duplicates quickly reduces this to (approximately) the 128 unique individuals (for a 64-bit problem). Symbiotic combination (Figure 2) is applied always, no mutation is required. The number of trials, t, used in testing the dominance of two individuals is at most 50, but most tests fail in less than 10 trials. Performance in Figure 6 is measured by the fitness of the best string evaluated (in the preceding 2000 evaluations) averaged over 30 runs for each algorithm. The problem size of 64 bits gives a maximum fitness of 448.

We see that the regular GA, using either crossover operator, tends to converge on sub-optimal solutions. The disruption caused by uniform crossover [Watson & Pollack 2000b] makes it worse than one-point crossover at first, but ultimately allows exploration that outperforms one-point. Actually, uniform crossover succeeds in 16 of the 30 runs, which is better than expected [Watson & Pollack 2000a], but those that do succeed take about 1,200,000 evaluations to do so. The ICGA is very slow to start because, unlike the GA, it must discover building-blocks explicitly – one per individual. But, eventually the ICGA shows that this method of partial-specification and symbiotic combination does allow successful combination of building-blocks in poor-linkage problems (about 1,700,000 evaluations permits 100% success). However, as noted, the ICGA uses a problem-specific size-penalty and partial-evaluation to achieve this. In contrast, SEAM performs very rapidly and successfully without using a size-penalty or partial evaluation. Group templating and Pareto coevolution introduced in SEAM prove to be very effective at enabling effective symbiotic combination. Control experiments, not shown, confirm that the use of either random templates instead of group evaluation, or replacement based on superior average performance instead of Pareto dominance, both cause the algorithm to fail. In either case, strings quickly fill with sub-optimal blocks and the combination operator is prevented from operating.

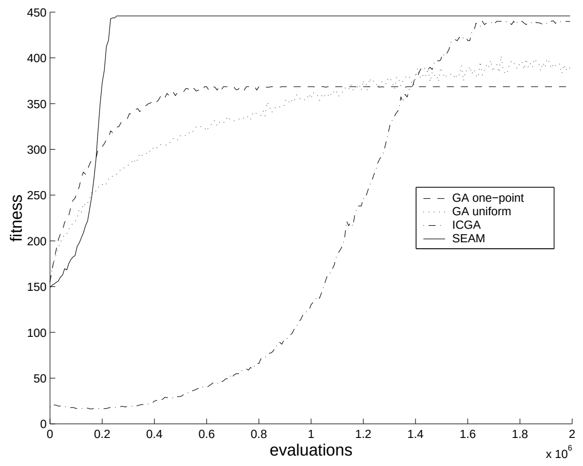

Figure 6: Performance of regular GA (using one-point and uniform crossover), ICGA, and SEAM, on Shuffled HIFF.

All algorithms except the GA-onepoint, perform the same on HIFF as on SHIFF, since they have no locus-dependent features. For reference, the GA-one point succeeds on regular HIFF, in all 30 runs, in less than 100,000 evaluations [Watson & Pollack 2000a]. Note that SEAM succeeds in solving SHIFF, a significantly harder problem, in a little over 200,000 evaluations.

5 Conclusions

To summarize, SEAM combines three new features with respect to a standard GA: 1) Partially-specified individuals and symbiotic combination (Figure 2) instead of sexual recombination. 2) Group evaluation (Figure 4) to provide contexts/templates that preclude the need to evaluate partially specified strings. 3) Pareto coevolution using different contexts to automatically define multiple dimensions for the problem space, thereby segregating competition to maintain diversity in the population, and prevent large sub-optimal individuals from replacing small optimal individuals. But although SEAM introduces several new concepts, it is algorithmically quite simple (Figure 5).

SEAM is the first known algorithm to solve Shuffled HIFF reliably. However, although SEAM is superior when applied to SHIFF, we have yet to compare SEAM’s performance with that of other algorithms on different problem domains. In the meantime, our experiments illustrate some important principles in (re)combination methods. SEAM demonstrates that symbiotic combination of partially-specified individuals can provide a successful alternative to sexual recombination for building-block assembly in problems of poor genetic linkage.

Acknowledgments

The first author is indebted to Martin Oates for tireless investigations of diversity maintenance techniques, Sevan Ficici for stimulating discussion on coevolution and its connection to Pareto optimization, and other members of DEMO at Brandeis.

References

Altenberg, L, 1995 "The Schema Theorem and Price's Theorem", *FOGA3*, editors Whitley & Vose, pp 23-49, Morgan Kaufmann, San Francisco.

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

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.

Horn, J, 1997, "Multicriteria Decision Making and Evolutionary Computation", in *Handbook of Evolutionary Computation*, T. Bäck, D.B. Fogel, and Z. Michalewicz (eds.), IOP Press, NY.

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

Maynard-Smith, J, and Szathmary, E, 1995 *The Major Transition in Evolution*, WH Freeman & Co.

Merezhkovsky KS, 1909 "The Theory of Two Plasms as the Basis of Symbiogenesis, a New Study or the Origins of Organisms," *Proceedings of the Studies of the Imperial Kazan University*, Publishing Office of the Imperial University. (In Russian).

Moriarty, DE, 1997, *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*, PhD thesis, University of Texas at Austin, USA.

Potter, M, 1997, *The Design and Analysis of a Computational Model of Cooperative Coevolution*, PhD thesis, George Mason University, Fairfax, Virginia.

Vekaria, K, & Clack, C, 1999, "Hitchhikers Get Around", *Artificial Evolution (EA) 1999*, November 3-5, LIL, Universite du Littoral, Dunkerque, France.

Watson, RA, Hornby, GS & Pollack, JB, 1998, "Modeling Building-Block Interdependency", *PPSN V, proceedings of Fifth International Conference,* Springer 1998, pp.97-106 .

Watson, RA, & Pollack, JB, 1999a, "Incremental Commitment in Genetic Algorithms", *Proceedings of GECCO 1999.* Banzhaf, et al. eds., Morgan Kaufmann, 710-717.

Watson, RA, & Pollack, JB, 1999b, "Hierarchically-Consistent Test Problems for Genetic Algorithms", *Proceedings of 1999 CEC.* Angeline, et al. eds. IEEE Press, pp.1406-1413.

Watson, RA, & Pollack, JB, 1999c, "How Symbiosis Can Guide Evolution". *Procs. of Fifth European Conference on Artificial Life*, Floreano, D, Nicoud, JD, Mondada, F, eds., Springer.

Watson, RA, 2000, "Analysis of Recombinative Algorithms on a Hierarchical Building-Block Problem", *FOGA 6*, Fogarty, Martin, Spears, eds. Springer, to appear (2001).

Watson, RA, & Pollack, JB, 2000a, "Combination and Recombination in Genetic Algorithms", technical report CS-00-209, Dept. Computer Science, Brandeis University.

Watson, R.A. & Pollack, J.B. 2000b, "Recombination Without Respect: Schema Combination and Disruption in Genetic Algorithm Crossover", *Procs. of GECCO 2000*, Whitley, D, et al (eds.), Morgan Kaufmann, 2000.