Adapting to Complexity During Search in Combinatorial Landscapes*

Terry P. Riopka

Lehigh University, Department of Computer Science and Engineering
200 Packard Lab, 19 Memorial Drive West
Bethlehem PA 18015
riopka@eecs.lehigh.edu

Abstract. Fitness landscape complexity in the context of evolutionary algorithms can be considered to be a relative term due to the complex interaction between search strategy, problem difficulty and problem representation. A new paradigm for genetic search referred to as the Collective Learning Genetic Algorithm (CLGA) has been demonstrated for combinatorial optimization problems which utilizes genotypic learning to do recombination based on a cooperative exchange of knowledge (instead of symbols) between interacting chromosomes. There is evidence to suggest that the CLGA is able to modify its recombinative behavior based on the consistency of the information in its environment, specifically, the observed fitness landscape. By analyzing the structure of the evolving individuals, a landscape-complexity metric is extracted *a posteriori* and then plotted for various types of example problems. This paper presents preliminary results that show that the CLGA appears to adapt its search strategy to the fitness landscape induced by the CLGA itself, and hence relative to the landscape being searched.

1 Introduction

It is well known that problem representation can greatly affect the efficiency of search and can alter the apparent function complexity with respect to the search strategy of the algorithm being used. The application of evolutionary methods often leads to the dilemma of matching the particular search heuristics and problem representation used to the complexity of the problem being solved. Of great benefit would be a method for predicting which algorithm (and its parameters) are best suited to the problem at hand. One way to approach this problem is try to measure problem complexity using a metric like epistasis variance [Davidor,1991], Walsh sums [Heckendorn *et al.*, 1997], fitness-distance-correlation (FDC)[Jones & Forrest, 1995] or density of states [Rose *et al.*,1996] and then use the result to classify a given problem for analysis.

Unfortunately, metrics can be misleading as they rely on statistical sampling of the solution space, and except for FDC, completely ignore the question of the search algorithm. As [Jones and Forrest, 1995] have shown, the fitness landscape observed for

* Lecture Notes in Computer Science, 2611. In *Proc. of Applications of Evolutionary Computing,EvoWorkshop* 2003

a particular function is an artifact of the algorithm, or more accurately, of the neighborhood induced by the operators the algorithm employs.

Another way would be to take an adaptive approach, to devise an algorithm that modifies its own behavior in the process of solving a given problem. Examples of this approach range from the self-adaptive mutation rates used in evolutionary strategies[Back, 1997] to the evolving representations pioneered by [Goldberg, *et al.*, 1989]. A comprehensive list of such methods is too extensive to be discussed here, but may be found in a recent survey [Hinterding, *et al.,* 1997]. In the case of self-adaptation, operators are often adjusted based on an analysis of the progress of the algorithm. A frequent criticism is the lag in the time between operator adjustment and algorithm response. In the case of evolving representations, the intent is to discover and analyze variable interactions in order to efficiently deal with them. However, assumptions regarding problem complexity are required to make these methods work.

The Collective Learning Genetic Algorithm (CLGA) [Riopka and Bock, 2000][Riopka, 2002] is an example of the latter approach, but is shown here to also use a type of implicit problem-difficulty metric by virtue of the mechanism used to do recombination. The metric does not ignore the question of the search algorithm because it is part *of* it. In this paper, a **landscape complexity** metric is extracted *a posteriori* by analyzing the structure of the evolving individuals and then plotted for various types of example problems. Preliminary results show that the CLGA appears to respond to problem difficulty naturally, as a consequence of its operation. The CLGA search strategy is reflected in the empirically derived landscape complexity metric, and is shown to adapt to the problem at hand. Adaptation occurs without any explicit operator control and without any assumptions regarding problem complexity.

The paper is organized as follows. Section 2 reviews the main features of the CLGA to provide the necessary background for understanding the landscape complexity metric. Section 3 discusses how the landscape complexity metric is extracted from the CLGA. Section 4 presents and discusses the results, and the paper is summarized and concluded in Section 5.

2 CLGA Description

A complete description of the CLGA is given in the sources cited previously. The following section briefly reviews the main features of the algorithm.

A CLGA consists of a population of adaptive learning agents called **SmartChromosomes**. Each SmartChromosome consists of two components: an instantiation of a collective learning automaton (CLA) and a chromosome string representing the best solution the SmartChromosome has found so far. The concept of a CLA is derived from collective learning systems theory [Bock, 1993]. The CLGA differs from other evolutionary algorithms in one very important way, and that is that recombination involves *not* an exchange of schemata, but an exchange of *information* which is then used by individual chromosomes to guide. Since that information is derived from observations of schemata within the population, recombination is necessarily driven by the quality and consistency of that information.

A CLA consists of a fixed number of **feature detectors**, each associated with a histogram that contains the accumulated knowledge of the fitness of schemata the SmartChromosome has "observed". Each feature detector monitors a unique set of chromosome sites. Its corresponding histogram contains one bin for every possible permutation of symbols for the chromosome sites, which the feature detector monitors. The number of feature detectors in each SmartChromosome (d) and the cardinality of the set of monitored sites (k) are both parameters of the algorithm. For example, for binary encodings, a feature detector monitoring k chromosome sites will have 2^k bins. Figure 1 shows a graphical depiction of two SmartChromosomes, taken from a binary encoded population with d=4, k=1 and string length N=12. The simplest case of k=1 is shown in the example to make it easier to understand the extracted landscape-complexity metric discussed in the next section.

Given a feature detector cardinality of k, the total number of possible combinations of monitored sites is N chromosome sites taken k at a time ${}_{N}C_{k}$. The number of SmartChromosomes (M) in the CLGA population is determined by the number of feature detectors per SmartChromosome and the **combination ratio** r_c, the fraction of combinations actually incorporated into the population. Hence,

$$M = \left\lceil \frac{r_c \, {}_{N}C_{k}}{d} \right\rceil \tag{1}$$

Incorporating all combinations insures problem-representation independence with respect to the recombination process. A population is created by randomly selecting the required number of combinations (without replacement) from the total and distributing feature detectors as randomly as possible among the SmartChromosomes. With large populations and small k, the combination ratio is usually much larger than one and corresponds to the number of copies of each feature detector in the population.

Initial chromosome strings are generated randomly. Each generation consists of five stages: mating, intelligent recombination, directed mutation, evaluation and individual selection, all of which are executed locally by the individual SmartChromosomes.

Mating. Each SmartChromosome mates with m other SmartChromosomes selected randomly from the population.

Intelligent Recombination. Intelligent recombination comprises two processes: an acquisition of knowledge (**inspection**) and application of that knowledge to direct recombination (**interrogation**). A SmartChromosome *inspects* the strings of its mates by noting in each mate's string the particular permutation of symbols at the location of the sites monitored by the SmartChromosome's feature detectors. The bin weight associated with the observed permutation is replaced by the average of the current string fitness and all other string fitnesses that contributed to that bin in the past. A SmartChromosome *interrogates* its mates by superposing the best states of its own feature detectors with those of its mates' feature detectors in order to modify its string. To accomplish this, first all best states are sorted in descending order by weight magnitude. The corresponding permutations of symbols are superposed sequentially state by state beginning with the best state with the largest magnitude overwriting the relevant chromosome sites in the SmartChromosome's string. In order for a state to be **consistent**, none of the monitored sites of the corresponding feature detector may

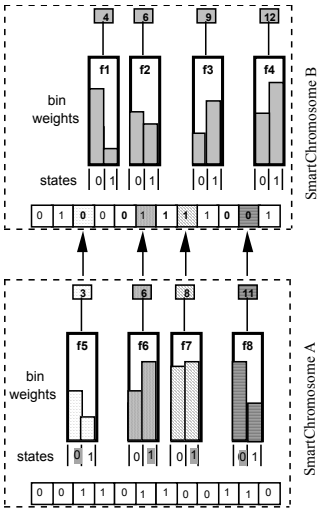

Fig. 1. During inspection, the feature detectors of SmartChromosome A "observe" the shaded symbols pointed to by arrows. For feature detector f5, this causes the bin weight for state [0] to be replaced by the average of the fitness of SmartChromsome B's string and all other string fitnesses that contributed to that bin in the past. The same is done for bin weight f6[1], f7[1] and f8[0]

overlap with those of any of the previously integrated states or, in case of overlap, the symbols in the overlapping sites must be identical. States, which are not consistent, are omitted. The fraction of a chromosome's sites affected by the best state superposition is referred to as the **superposition fraction** (s_f.).

Directed Mutation. Standard mutation is not applied in the CLGA, contrary to the experiments presented in [Riopka and Bock, 2000]. Many informal experiments have since shown that standard mutation is not needed and in fact, actually degrades CLGA performance. Consequently, a *directed mutation* operator is applied as follows. Each SmartChromosome maintains a FIFO* list of the last h unique chromosome strings it has evaluated, where h is some small number (to limit time and memory resources) referred to as **history length**. A SmartChromosome first searches its memory to see if the current string has already been evaluated. This will occur only if interrogation results in the creation of a string already in memory. If the string has already been evaluated, a single bit is randomly selected in the string and complemented. Only those bits that have *not* already been changed are eligible for mutation each time. Once all single bits are exhausted, combinations of two bits are complemented. Once those combinations are exhausted, combinations of three bits are complemented, etc. The need for directed mutation is related to the problem of feature detector convergence, a topic currently under research.

Evaluation. The SmartChromosome evaluates its string using the fitness function.

* FIFO here means that the chromosome that hasn't been *repeated* in the longest time is replaced first.

Individual Selection. If a modification results in a string whose fitness is greater than the previous string, the new string is retained, otherwise, the SmartChromosome reverts to the previous one. In effect, the offspring competes with its parent. The SmartChromosome inspects its new string regardless of what its relative fitness is, thus learning from both its successes and its failures.

3 A Landscape Complexity Metric

Significant evidence exists to strongly suggest that CLGA performance is independent of the size of the feature detector (k). For a detailed discussion of this result, the reader is referred to [Riopka, 2002]. Consequently, it is conceivable that the following analysis for k=1 may be extrapolated to higher values of k with similar results. Therefore, all of the experiments presented here use a value of k=1 for which the following analysis is valid.

Chromosome strings are modified during the interrogation process. If we consider the mating of two SmartChromosomes with k=1, feature detectors that they have in common will tend to compete. This case exists in figure 1 for feature detectors f2 and f6 at chromosome site 6. In this example however, the "opinion" of each feature detector as to the best bit value for chromosome site 6 differs. SmartChromosome A (feature detector f6) has a greater bin weight for bit 1 while SmartChromosome B (feature detector f2) has a greater bin weight for bit 0 at that chromosome site. In this particular competition, f6 will prevail because its best state f6[1] has a bin weight that is greater than the bin weight associated with the best state f2[0]. The bit at chromosome site 6 will therefore be set to 1.

In a typical application, reasonable sized populations using k=1 give rise to a large number of copies of each feature detector. Consider a single chromosome site. For a given generation, that site will have some number of feature detectors in the population with one *opinion* as to the best bit state for that chromosome site, while the remainder will have the opposite *opinion*. If we take all those with a best bit state of 0 and compute the first two moments of the bin weight, and do the same for those with a best bit state of 1, we can test the null hypothesis that there is no difference between the means (assuming a t-distribution and using a two-tailed test).

Figure 2 shows the calculation of the t-statistic for an example chromosome of length N=12. The t-statistic is used to compute the confidence associated with the alternative hypothesis, that there is a difference between the two means. The intuition behind this is that a high confidence should indicate how *sure* the feature detectors have become regarding their *opinion* as to which bit state is correct, regardless of which bit state that is. The idea is that out of the feature detectors that agree with one another, in a competition they will simply set their bit to the agreed value. However, in instances where two feature detectors disagree, a high confidence implies that the difference will most likely be quite stark, and hence the decision as to which bit is set will be quite firm (either all tending to 0 or all tending to 1). On the other hand, a low confidence implies that there is little statistical difference between best states that disagree and hence bits will tend to flip between 0 and 1 more often, inducing a degree of randomness into the decisions being made.

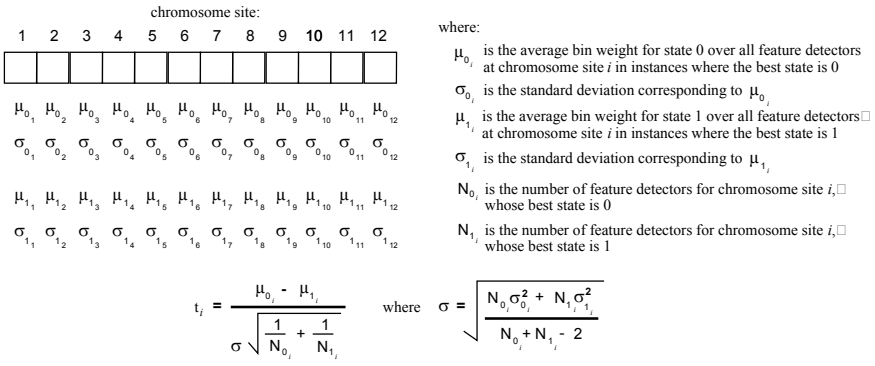

Fig. 2. The calculation of the t-statistic used to compute confidence values is shown in relation to an example chromosome of length N=12 and the corresponding feature detector moments

A single value was computed by averaging the confidence values over all of the chromosome sites, arriving at a single metric for the entire population for a given generation, referred to as the **average bit certainty**. In the following experiments, the t-score was used to compute confidence values due to the small combination ratio.

It is well known that as epistasis increases, the fitness landscape becomes more and more uncorrelated [De Jong, 1993]. It would therefore be expected that as epistasis increases the CLGA would find it more and more difficult to learn consistent relationships between bits due to greater variance in solution fitness. Consequently, this metric would be expected to be inversely proportional to the level of epistasis. Results from several experiments support this hypothesis.

4 Experiments

4.1 Fixed Parameters and Performance Metrics

Several problem generators have been developed to facilitate the design of more controlled experiments for testing evolutionary algorithms. Random problems generated by NK-Landscape, LSAT and Multi-modal problem generators were used in the following experiments. The reader is referred to [DeJong *et al.*, 1997] and [Heckendorn *et al,* 1998] for details.

The following fixed parameter values for the CLGA were used: m=1, h=4, k=1, s_f=15%, and a population size of M = 20. NK-Landscape problems of length N=30 were tested using d=23, while remaining problems of length N=100 were tested using d=75, resulting in a combination ratio of approximately r_c = 15. Although these parameters were not optimized, they were selected based on heuristics derived from extensive experiments detailed in [Riopka, 2002]. A relatively small population size was used for the following experiments, but very similar results were also obtained using larger population sizes as well. Average bit certainty was plotted versus function evaluations in all experiments.

4.2 NK-Landscape Experiments

Fifty random, 30-bit NK-Landscape problems using a neighborhood model were used in the first set of experiments. The CLGA was run for 50000 evaluations on three levels of epistasis, K=5, 10 and 15. Results are shown in figure 3.

Note that as epistasis increases, the average bit certainty as measured within the CLGA population decreases. This is consistent with the interpretation given previously, *i.e.*, that the weaker fitness correlations for higher epistasis problems cause the CLGA to behave more randomly. It is interesting to note that NK-Landscape problems with lower values of K seem to have a greater range of average bit certainty values. This is consistent with the intuition that with lower epistasis, there is always some non-zero probability of some problems being "hard" simply by accident.

4.3 L-SAT Experiments

Fifty random, 100-bit 3-SAT problems were used in the next set of experiments. The CLGA was run for 30000 evaluations on three levels of epistasis, with the number of clauses set to C=430, 1200, and 2400 for low, medium and high epistasis respectively. Results are shown in figure 4. To our surprise, there did not seem to be much difference between the 3-SAT problems for the three levels of epistasis. This perplexing result however, supports observations made by [DeJong, *et. al.*, 1997]. In that paper, the experimenters noted that a low mutation rate (less exploration) and low crossover rate (lower disruption) actually improved results. The results here suggest that the various levels of epistasis for 100-bit 3-SAT problems may contain more structure than previously supposed. The CLGA search strategy, as reflected by the average bit certainty, suggests the need for a less random search over all three levels of epistasis, implying greater problem structure and perhaps explaining why greater preservation of structure (less exploration) resulted in improved performance for DeJong *et al.*

Note the greater variance in the average bit certainty as epistasis decreases. This is consistent with the fact that the 100 variable 3-SAT is known to have a "phase transition" at approximately 430 clauses [Crawford and Auton,1996], resulting in problems that have an almost equal probability of being satisfiable and unsatisfiable and hence computationally most difficult [Gomes and Selman, 2002]. 3-SAT problems with fewer clauses are almost always satisfiable while those with more are almost always unsatisfiable. The larger variance may reflect the diversity of problems in this critical region. Experiments were also run for 3-SAT problems with fewer clauses, but, as expected, always obtained the optimum faster than an interpretable bit certainty could be extracted.

A second set of experiments was run using a fixed number of clauses (C = 1200) but varying the length of the clauses using values L = 2,3,4 and 5. Again, the CLGA was run on fifty random problems for 30000 evaluations. Results are shown in fig. 5.

In this case, a difference was observed, showing a decrease in bit certainty as the length of the clause was increased. This is not surprising if one considers the algorithm used to generate the L-SAT problems. As the clause length increases, the probability of

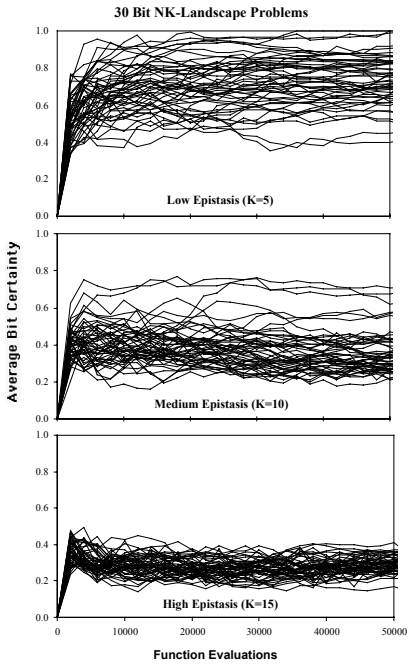

Fig. 3. Average bit certainty plotted for 50 random 30-bit NK-Landscape problems for low, medium, and high epistasis

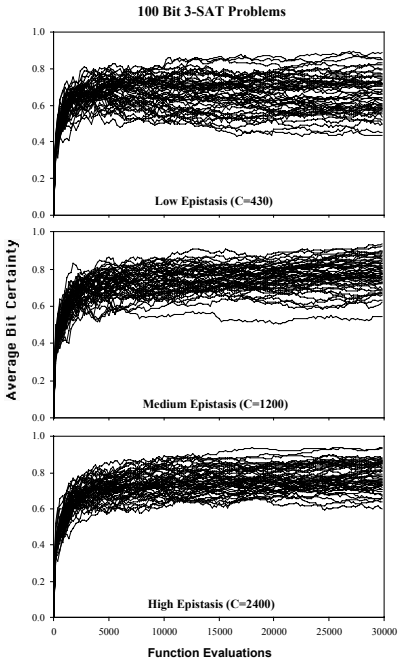

Fig. 4. Average bit certainty plotted for for 50 random 100-bit 3-SAT problems for low, medium and high epistasis

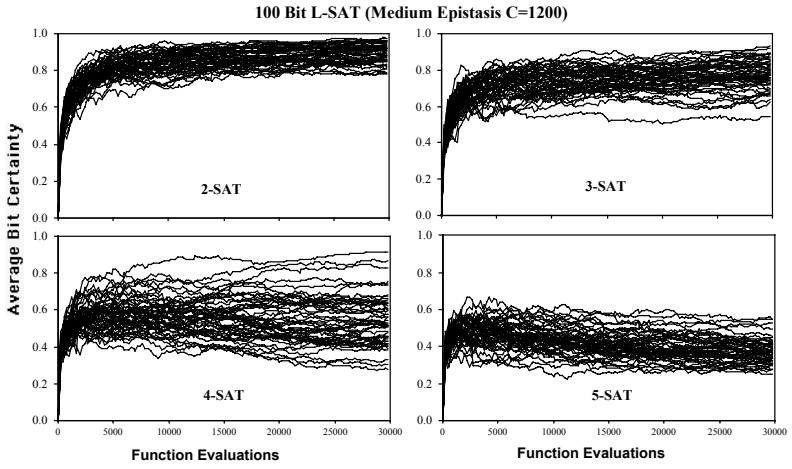

Fig. 5. Average bit certainty plotted for the CLGA for fifty random 100-bit 2-SAT, 3-SAT, 4-SAT and 5-SAT problems using 1200 clauses

satisfying the clause increases (since only a single true value is required to make the entire clause true). A more random strategy is therefore reasonable, since the likelihood of satisfying clauses is high. On the other hand, with a short clause length, a more methodical approach may be more efficient (in the long run).

4.5 Multi-Modal Experiments

A final set of experiments was run using 50 random, 100-bit Multi-modal problems. The CLGA was run for 30000 evaluations on two levels of epistasis with the number of peaks equal to 1 for low epistasis and 500 for high epistasis. Results are shown in figure 6. Only 5000 evaluations are shown since all optima were found in that time. The results of the last set of experiments were entirely consistent with our expectations. The single peak problems resulted in a less random approach due to the highly correlated fitness landscape. The higher epistasis problems on the other hand, resulted in significantly more random behavior, given the less correlated landscape. Optima, however, were found in both cases in approximately the same time.

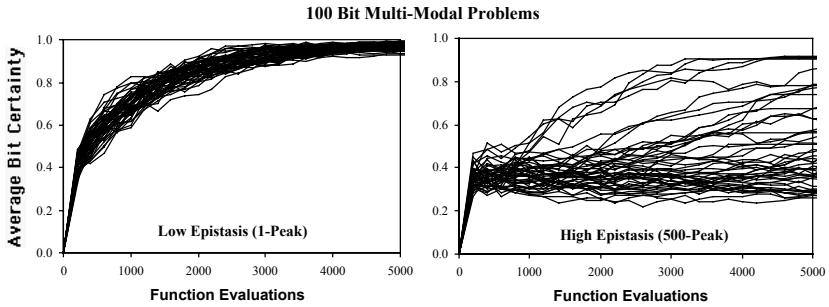

Fig. 6. Average bit certainty plotted for the CLGA for fifty random 100-bit Multi-modal problems for low, and high epistasis

5 Conclusions

How is the computed average bit uncertainty different in principle from any of the metrics cited earlier? It too is based on information ultimately obtained from sampling of solutions. However, the difference is that the sampling is done from both successes and failures of the particular *operator* being used (in this case, intelligent recombination). In other words, the information accumulated in the feature detector histograms is based on not only good schemata (gleaned from inspections of a SmartChromosome's mates), but on schemata in the neighborhoods of the fitness landscape directly induced by the intelligent recombination. Recall that after modifying its schemata, a SmartChromosome inspects the result, whether it is better or worse than its original string. If the fitness landscape is an artifact of the search algorithm as Jones and Forest suggest [Jones and

Forest, 1995], then the CLGA is responding to the fitness landscape it is navigating through. If the landscape indicates that its generated solutions are uncorrelated, the CLGA responds by generating trial solutions more randomly. On the other hand, if the landscape indicates its generated solutions are correlated, the correlations are strengthened and the CLGA responds by generating more correlated responses.

The empirical landscape complexity metric computed in this paper reflects the strategy chosen by the CLGA to solve the given problem. One cannot make an argument that it is in any way an objective measure of *problem* difficulty. However, if problem difficulty can be considered a relative term with respect to a search algorithm then taken from that perspective the metric is ideal (for the CLGA). The real question is, what kinds of problems evoke which type of behavior? More importantly, is the behavior evoked by a given problem suitable for solving it (efficiently)? The NFL theorem [Wolpert and Macready, 1997] strongly suggests that the answer to this question is no in general, but it is interesting to consider what types of problems respond to the CLGA approach and the scope of its applicability. Future research will investigate information theoretic constraints on the efficacy of the CLGA and hopefully provide more insight into a fascinating new mechanism for evolutionary computation.

References

1. Back, T. (1997). Self Adaptation. In *The Handbook of Evolutionary Computation*, pp. 1-23. IOP Publishing and Oxford University Press.
2. Bock, P. (1993). *The Emergence of Artificial Cognition*. World Scientific Pub. Co.
3. Crawford, J.A., Auton, L.D. (1996). Experimental Results on the Crossover Point in Random 3SAT. Art. Int. Vol. 81, No. 31.
4. Davidor, Y. (1991). Epistasis Variance. In *Foundations of GAs* Morgan Kaufmann.
5. DeJong, K.A., Potter, M.A., Spears, W.M. (1997). Using Problem Generators to Explore the Effects of Epistasis. In *Proc. 7th Int. Conf. on GAs*. Morgan Kaufmann.
6. Gomes, C.P., Selman, B. (2002). Satisfied with Physics. *Science*. Vol. 297 No. 5582.
7. Heckendorn, R.B., Whitley, D. (1999). Walsh Functions and Predicting Problem Complexity. *Evol. Comp.,* Vol. 7, No. 1, pp. 69-101. MIT Press.
8. Hinterding, R., Michalewicz, Z., Eiben, A.E. (1997). Adaptation in Evolutionary Computation: A Survey. In *Proc. 4th Int. Conf. on Evol. Comp.* pp. 65-69. IEEE Press.
9. Jones, T., Forrest, S. (1995). Fitness Distance Correlation As a Measure of Problem Difficulty for GAs. In *Proc. 6th Int. Conf. on GAs*, pp. 184-192. Morgan Kaufmann.
10. Reeves, C. R. (1999). Predictive Measures for Problem Difficulty. In *Proc. 1996 Congress on Evol. Comp.,* pp. 736-743. IEEE Press.
11. Riopka, T. P. (2002). Intelligent Recombination Using Genotypic Learning in a Collective Learning Genetic Algorithm, Doctoral dissertation, GWU, Washington, DC.
12. Riopka, T.P., Bock, P. (2000). Intelligent Recombination Using Individual Learning in a CLGA, In *Proc. Genetic and Evol. Comp. Conf.,* pp. 104-111. Morgan Kaufmann.
13. Rose, H., Ebeling,W., Asselmeyer, T. (1996). The Density of States - a Measure of the Difficulty of Optimization Problems. In *PPSN- IV,* Springer-Verlag.
14. Wolpert, D.H., Macready, W.G. (1997). No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comp.,* vol. 1, no. 1, pp. 67-82.