Advanced Neighborhoods and Problem Difficulty Measures

Mark Hauschild
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Mathematics and Computer Science
University of Missouri in St. Louis
mwh308@umsl.edu

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Mathematics and Computer Science
University of Missouri in St. Louis
pelikan@cs.umsl.edu

ABSTRACT

While different measures of problem difficulty of fitness landscapes have been proposed, recent studies have shown that many of the common ones do not closely correspond to the actual difficulty of problems when solved by evolutionary algorithms. One of the reasons for this is that most problem difficulty measures are based on neighborhood structures that are quite different from those used in most evolutionary algorithms. This paper examines several ways to increase the accuracy of problem difficulty measures by including linkage information in the measure to more accurately take into account the advanced neighborhoods explored by some evolutionary algorithms. The effects of these modifications of problem difficulty are examined in the context of several simple and advanced evolutionary algorithms. The results are then discussed and promising areas for future research are proposed.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.6 [**Artificial Intelligence**]: Learning; G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Algorithms, Performance

Keywords

Genetic Algorithms, Hierarchical BOA, estimation of distribution algorithms, difficulty measures

1. INTRODUCTION

Understanding why certain problems are more difficult than others would aid researchers greatly in evolutionary computation. Towards this goal, many different measures have been proposed to assess problem difficulty for evolutionary algorithms and other metaheuristics. Some of the most common are the fitness distance correlation [10], the autocorrelation function [28], the signal-to-noise ratio [7], and scaling [27]. A number of studies have been done to measure the effectiveness of these measures on various types of optimization problems [28, 10, 12, 26]. However, a recent study [17] showed that in many cases these measures of problem difficulty did not correlate with the actual computational requirements of an advanced evolutionary algorithm, the hybrid hierarchical Bayesian optimization algorithm (hBOA) [18, 19, 16] and there are several studies that presented critical views on various measures of problem difficulty [23, 15].

One of the reasons for the lack of correlation between the measures of problem difficulty and the actual performance of advanced evolutionary algorithms is that the measures of difficulty and the evolutionary algorithms use different neighborhood structures. In most analyses to date, problem difficulty measures would exclusively use the single-bit flip neighborhood, in which candidate solutions are represented by binary strings and any two strings different in exactly one string position are considered neighbors. This means that the evolutionary algorithm is assumed to explore the search space by changing one bit or one variable at a time and the distance of two solutions is defined by the Hamming distance. On the other hand, many advanced evolutionary algorithms are capable of identifying interactions between problem variables and they use this information to change groups of bits or problem variables at a time, or they use other advanced operators capable of performing nontrivial modifications to solutions strings. It is possible that difficulty measures that use simple distance measures such as Hamming distance are simply not able to capture the difficulties inherent to certain types of problems when algorithms are using more complex neighborhoods to explore the search space.

The purpose of this paper is to explore whether it is possible to increase the accuracy of the problem difficulty measures in the context of advanced evolutionary algorithms by using the neighborhood structures that more closely correspond to the variation operators used. As the starting point, this paper considers two difficulty measures, the fitness distance correlation and the correlation length. These measures are then extended to explore nontrivial neighborhoods by exploiting linkage information. The resultant measures are then applied to a large number of random instances of additively separable problems. To see how closely the resulting difficulty measure matches up to advanced evolutionary algorithms, their accuracy in measuring the difficulty of ran-

GECCO'11, July 12–16, 2011, Dublin, Ireland.

domly separable problems is compared to the actual performance from an ideal extended compact genetic algorithm (ECGA) [8] given exact linkage information and the hierarchical Bayesian optimization algorithm (hBOA) [18, 19, 16]. To provide a point of reference, the results are also analyzed with respect to the performance of the simple genetic algorithm (GA) with uniform crossover.

The paper is organized as follows. Section 2 outlines the class of random additively separable problems. Section 3 describes the algorithms tested. Section 4 describes the problem difficulty measures considered in this paper and their modifications using nontrivial neighborhoods. Section 5 presents the experimental results. Lastly, section 6 summarizes and concludes the paper.

2. RANDOM ADDITIVELY SEPARABLE PROBLEMS

The fitness of an additively separable problem is defined by a sum of subfunctions of non-overlapping proper subsets of its variables [6]:

$$f_{sep}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i=0}^{m-1} f_i(X_{I_i}) \quad (1)$$

where each X_{I_i} denotes one of the subsets of $\{X_0, \ldots, X_{n-1}\}$, and each f_i denotes one subfunction.

In this paper we consider instances of random additively separable problems (rASP) where each subset is of the same size k. Each subfunction is defined as a lookup table which specifies a return value for each combination of bits of the corresponding subset. This lookup table covers all possible instantiations of bits in each subfunction and is generated randomly using the same distribution for each entry in the table (in our case, the values are generated using the uniform distribution over the interval $[0, 1)$. Each random instance is fully described by its total size n, the partition size k and the lookup tables for each partition.

To make the instances more challenging, string positions in each instance are *shuffled* by reording string positions according to a randomly generated permutation using the uniform distribution over all permutations. The algorithm used to solve the rASP instances in this paper is based on refs. [21, 22].

3. ALGORITHMS

3.1 Simple Genetic Algorithm

The genetic algorithm(GA) [9, 5] evolves a population of candidate solutions typically represented by binary strings of fixed length. The starting population is generated at random according to a uniform distribution over all binary strings. Each iteration starts by selecting promising solutions from the current population; in this work we use binary tournament selection without replacement. New solutions are created by applying uniform crossover and bit flip mutation. These new candidate solutions are then incorporated into the population using restricted tournament replacement (RTR) [16]. (RTR) is a niching method that helps to ensure diversity in a population by having new candidate solutions replace solutions that are similar to themselves in the population. The next iteration is executed unless some predefined termination criteria are met. For example, the run can be terminated when the maximum number of generations is reached or the entire population consists of copies of the same candidate solution.

3.2 hBOA

Some of the most powerful evolutionary algorithms are estimation of distribution algorithms (EDA) [2, 14, 11, 20]. EDAs work by building a probabilistic model of promising solutions and sampling new candidate solutions from the built model. The hierarchical Bayesian optimization algorithm (hBOA) [18, 19, 16] is an EDA that uses Bayesian networks to represent the probabilistic model. The initial population is generated at random according to the uniform distribution over the set of all potential solutions. Each iteration (generation) starts by selecting promising solutions from the current population using any standard selection method of genetic and evolutionary algorithms.

After selecting the promising solutions, hBOA uses these solutions to automatically learn both the structure (edges) as well as the parameters (conditional probabilities) of the Bayesian network. In this paper, a greedy algorithm is used to learn the structure of BNs with local structures [16]. To evaluate structures, the Bayesian-Dirichlet metric with likelihood equivalence for BNs with local structures [3] is used with an additional penalty for model complexity [4, 16].

The Bayesian network model is then sampled to generate new candidate solutions, which are incorporated into the population with RTR. The next iteration is executed unless the termination criteria are met. For more details about the basic hBOA procedure, see [18] or [16].

3.3 Ideal ECGA

The extended compact genetic algorithm (ECGA) [8] starts by generating a population at random according to a uniform distribution over all binary strings. Each iteration of the algorithm, ECGA builds a marginal product model (MPM) that divides the variables into multiple partitions, which are processed as independent groups. Once the model is complete, the algorithm then stores the probability of any particular instance of a partition. This model is then sampled to generate new candidate solutions, which are then incorporated into the population using RTR.

For example, consider a problem with variables $\{x_1, x_2, \ldots x_8\}$. After model building the ECGA might divide them up into the disjoint partitions $\{x_1, x_4\}$, $\{x_2, x_6\}$ and $\{x_3, x_5, x_7, x_8\}$. ECGA would then store the probability of any particular instance of that partition in the promising solutions and use that probability to generate new instantiations of the partitions when sampling to generate new candidate solutions. An example of a MPM dividing the variables in our example into partitions that are treated independently is given in Figure 1

While in the regular ECGA the partitions are divided up according to the MDL metric, in this paper we consider an idealized version of ECGA in which the model building phase is replaced by a perfect model built from knowledge of the problem instances being solved. By doing this, the results should not be biased by incorrect linkage groups learned during the model building phase. In this way we can compare the performance of an idealized operator with fixed linkage groups against the different neighborhood difficulty operators in section 4. For the separable problems examined in this paper, each partition will contain all the

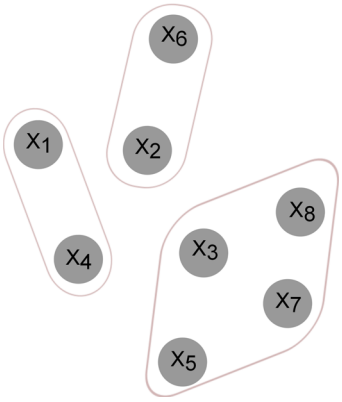

Figure 1: Graphical example of a MPM model built by ECGA. The variables are partitioned into independent groups.

variables in one of the separable subfunctions of size k. In the rest of this paper we will refer to this version of ECGA as ECGA$_{perfect}$.

3.4 Deterministic Hill Climber

For all GA, hBOA and ECGA$_{perfect}$ runs, a deterministic hill climber(DHC) was incorporated to improve performance. DHC takes a candidate solution represented by a n-bit binary string and performs one-bit changes on the solution that lead to the maximum improvement. This process is terminated when no possible single-bit flip improves solution quality.

4. PROBLEM DIFFICULTY

A fitness landscape consists of three main components: (1) A set S of admissible solutions, (2) a fitness function f that assigns a real value to each solution in S, and (3) a distance measure d that defines a distance between any two solutions in S. S and f define the problem being solved. Specifically, the task is to find $argmax_{x \in S} f(x)$. On the other hand, the distance measure depends on the operators used. Specifically, $d(x, y)$ defines the number of steps to get from x to y.

Defining a good distance measure that defines these steps is not always a trivial matter. For binary strings, Hamming distance is often used, which is equal to the number of string positions in which the two binary strings differ. This makes sense for evolutionary algorithms that use simple variation operators such as bit flip mutation, as solutions varying in a few bits should always be close to each other in steps. However, for more complex variation operators, it is possible that more complex neighborhoods should be considered. For example, the ECGA manipulates groups of bits at once when sampling a partition so it is possible that even though two solutions have many different bits between them, they should still be considered close. In this paper we will consider 3 different neighborhoods, which are described in section 4.3.

4.1 Correlation Length

Consider a random walk through the landscape which starts in a random solution and moves to a random neighbor of the current solution in each step (neighbors of a candidate solution are all solutions at distance 1 from it). To measure problem difficulty based on random walks, we can use the *random walk correlation function* (also called the fitness autocorrelation function) [28], which quantifies the strength of the relationship between the fitness values of a candidate solution x and the solutions that are obtained by taking a given number s of steps starting in x. In other words, the correlation function quantifies ruggedness of the landscape. For a random walk of $m-1$ steps passing through solutions of fitness values $\{f_t\}_{t=1...m}$, the random walk correlation function $\rho(s)$ for gap s is defined as [28]

$$\rho(s) = \frac{1}{\sigma_F^2 (m-s)} \sum_{t=1}^{m-s} (f_t - \bar{f})(f_{t+s} - \bar{f}), \qquad (2)$$

where s is the number of steps (gap), and $\bar{f}$ and σ_F denote the average fitness and the standard deviation of the fitness values, respectively. Typically, the larger the value of s, the weaker the correlations between fitness values; $\rho(s)$ can thus be expected to decrease with increasing s. Furthermore, the smaller the value of $\rho(s)$, the more rugged the landscape is. Therefore, the landscape should be relatively easier to explore for smaller $\rho(s)$ than for larger $\rho(s)$.

The correlation function can be used to compute the correlation length, which estimates the effective range of correlations between states in a random walk. The correlation length may be defined as [25]

$$l = -\frac{1}{ln(|\rho(1)|)}, \qquad (3)$$

The correlation function $\rho(s)$ can also be used to compute the autocorrelation coefficient $\delta = 1/(1-\rho(1))$ [1], which has approximately the same value as the correlation length [13]. The smaller the correlation length or autocorrelation coefficient, the harder the problem instance.

4.2 Fitness Distance Correlation

Consider a set of n candidate solutions with fitness values $F = \{f_1, f_2, \ldots, f_n\}$ and a corresponding set $D = \{d_1, d_2, ..., d_n\}$ of the distances of these solutions to the nearest global optimum. The fitness distance correlation (FDC) [10] quantifies the strength and nature of the relationship between the fitness value and the distance to the nearest global optimum as

$$r = \frac{c_{FD}}{\sigma F \sigma D} \qquad (4)$$

where σ_F and σ_D are standard deviations of F and D, respectively, and c_{FD} is the covariance of F and D. The covariance c_{FD} is defined as

$$c_{FD} = \frac{1}{n} \sum_{i=1}^{n} (f_i - \bar{f})(d_i - \bar{d}), \qquad (5)$$

where $\bar{f}$ and $\bar{d}$ are the means of F and D, respectively. Note that the computation of FDC necessitates knowledge of all global optima. FDC takes values from [-1, 1]. In general it should be easier to find the global optimum for smaller values of FDC than for larger ones as small FDC values means high fitness values are more likely to be consistently closer to the global optima than high FDC values.On the other hand, higher values of FDC indicate that the fitness may often mislead the search away from the global optimum. Thus, the smaller the values of r, the easier the maximization problem

should be. For example, for onemax, r = -1, whereas for the fully deceptive trap function of size 20, r $\approx$ +1 [10].

Since in this paper we are using DHC with all algorithms, we use a variant where only local optima are considered when calculating FDC. Each value of f_i is one local optima and d_i is the distance of that local optima from the closest global optima.

4.3 Neighborhoods

While using the simple bit-flip neighborhood can be sufficient for many problems, evolutionary algorithms often modify large numbers of related bits at the same time. In these cases, the simple bit-flip neighborhoods used could be misleading the difficulty measures, as solutions far apart in Hamming distance might be close together in the algorithms search space. For example, when you modify even just one of the k bits in one partition, it is the same as if you modified all of them, because they correspond to a single subfunction and the bits or variables in one partition may thus be strongly correlated or interdependent. To attempt to isolate the effect of different neighborhoods on the aforementioned difficulty measures, in this work we consider 3 different neighborhoods for correlation length:

Fixed partition This neighborhood is composed of all strings that are reachable by changing any combination of bits in a subfunction. For the rASP instances in this paper, this is all the strings that are reachable by changing any of the k bits in one of the subfunctions.

Random partition The neighborhood composed of all solutions that are reachable by changing at most x random bits. When used on the rASP instances in this paper, $x = k$, so that random neighborhoods of k can be compared to the strongly correlated neighborhoods of the fixed partition flip. This neighborhood is included to show that any change was not due to simply increasing the number of changed bits.

Single bit-flip This neighborhood is the simple bit-flip neighborhood commonly used.

Incorporating these neighborhoods into correlation length is straightforward. To implement the fixed partition neighborhood, the standard random bit walk is replaced by instead each step modifying randomly all of the variables that are in a single subfunction in the underlying instance. For the random partition neighborhood, a random set of size k is generated and all the variables in this set are modified randomly.

Incorporating these measures into FDC is more difficult. We must choose a distance metric such that if two solutions are next to each other in an advanced neighborhood, they will be of distance 1 in the metric. To do this, in this paper we consider 2 metrics when using FDC:

Single bit distance The simple hamming distance most commonly used with FDC, the sum total of the bit difference between two strings. This is used to compare the base fitness distance correlation against one that uses a more advanced neighborhood.

Partition distance This measures the distance by the amount of differing partitions in two strings. If the bits in a partition of one solution string are different from the corresponding bits in the same partition of a second solution string, then they are considered at least one apart. Their total distance from each other is the sum of how many different partitions they have from each other.

By comparing the two difficulty measures and the methods of incorporating different neighborhoods, it should be possible to see whether changing the type of neighborhood the difficulty measure uses improves the measure. For example, since the fixed partition flip neighborhood should strongly correspond to the type of neighborhoods in ECGA$_{perfect}$, it should be expected that the difficulty measures should more strongly correspond to the actual computational complexity of ECGA$_{perfect}$ when using the fixed partition neighborhood. On the other hand, it would be expected that the difficulty measures when used with the random neighborhoods (single bit-flip and random partition) would be less likely to correspond to the actual computational complexity of the instances was solved by ECGA$_{perfect}$.

5. EXPERIMENTS

5.1 Experimental Setup

For each problem instance, the correlation length was estimated by starting with 100 random walks of 1000 steps each, with all 3 neighborhood step operators used (bit-flip, random set flip and fixed partition flip). The correlation length and autocorrelation coefficient were estimated and if both these values were within 1% of their actual value with 99% probability (assuming Gaussian distribution of their means), the estimates were used. Otherwise, the random walks were repeated, this time extended by 1000 points each, with the maximum length of any walk restricted to one million steps. If the maximum length was exceeded, then the previous estimate is used.

The fitness distance correlation for each instance was calculated by starting with 100 samples of 1000 points each and then the fitness distance correlation r was computed, with this being done with all 3 different distance metrics. The local optima for these solutions was then found using DHC and the fitness distance correlation r_l was computed for the local optima. These two means were returned if they were within 1% of their true value with 99% probability. If not, an additional 1,000 points for each of the 100 samples was generated and the procedure repeated. As with correlation length, if the number of points exceeded one million, the procedure was terminated.

The GA, hBOA and ECGA$_{perfect}$ were applied to all problem instances. For all GA runs, bit-flip mutation was used with a probability of flipping each bit of $p_m = 1/n$, with a probability of crossover of 0.6. All algorithms used restricted tournament replacement (RTR) [16] as the replacement operator.

For all problem instances, bisection [24, 16] was used to determine the minimum population size to ensure convergence to the global optimum in 10 out of 10 independent runs, with the results averaged over the 10 runs. The number of generations was upper bounded according to preliminary experiments by $n * 4$, where n is the number of bits in the instance. Each run of GA, hBOA and ECGA$_{perfect}$ was terminated when the global optimum was found (suc-

cess) or when the upper bound on the number of generations had been reached without discovering the global optimum (failure).

In this paper instances of rASP of $n = 120$ are examined, with $k = 3$ and $k = 5$. 1000 random instances were considered for each of these types. The number of DHC flips required to solve an instance is used to rank instance difficulty, as using CPU time is not reliable when using a variety of hardware.

5.2 Correlation Length Results

The relationship between correlation length using the three different neighborhood types and their actual difficulty when solving instances with a GA with $k = 3$ is shown in Table 1a. The first column is the type of instances ranked by percentage difficulty (decided by number of local search steps). The second column shows the number of local search steps used for its set of instances. The remaining columns show the correlation length using the various neighborhood operators, with the ranking of the values shown in brackets. Noting that correlation length should be higher for the easier instances, the results show a strong relationship between correlation length using the bit flip neighborhood and the random partition neighborhood. In general as the problem difficulty increases, the correlation length decreases. However,correlation length using the set partition neighborhood seems to have no relation to the actual difficulty of the instances when solved with the GA.

The results for correlation length with respect to the instance difficulty when using hBOA to solve separable problems with $n = 120$ and $k = 3$ is shown in Table 1b. In the case of bit flip and random partition neighborhoods, there seems to be a very weak correlation between the difficulty of the instance and the corresponding correlation length. As with GA, the set partition neighborhood shows no relation, with the 50% most difficult instances having the lowest correlation length of any of the difficulty classes.

To examine the effect of the different neighborhoods on correlation length on an algorithm using ideal recombination, Table 1c shows the results on separable problems of $n = 120$ and $k = 3$ when solved with $ECGA_{perfect}$. Unlike the previous two algorithms, correlation length with bit flip neighborhood and random partition neighborhood does not show any relation to instance difficulty. The set partition neighborhood did not do any better either, failing to show any relation between instance difficulty and correlation length.

To examine the effect of increasing problem difficulty on the relationship between the different neighborhood types, algorithms and difficulty classes, instances of separable problems of $n = 120$ and $k = 5$ were examined. Table 2a shows relationship between the difficulty of instances for the GA and their corresponding correlation length using the three neighborhood types. Correlation length with bit-flip neighborhood shows little relation with instance difficulty, with the highest correlation length given to the 50% hardest instances. The set partition neighborhood also is not effective in ranking difficulty. However, the random partition does show a relationship.

In Table 2b the results for hBOA are shown on separable problems of $n = 120$ and $k = 5$. For the bit flip and random partition neighborhoods, there is a weak relationship between their actual difficulty and their correlation length. The set partition seems ineffective in this case, with the lowest correlation length going to the 25% easiest instances.

Lastly, the results for $\text{ECGA}_{perfect}$ are shown in Table 2c. The bit flip and random partition neighborhoods are unable to accurately rank the difficulty of the instances. The set partition neighborhood is even worse, showing an inverse relationship from what we would expect if the measure was working accurately, with higher correlation lengths corresponding to harder sets of instances.

As was suggested by the study of Pelikan [17], correlation length is not a good indicator for problem difficulty for decomposable problems of fixed size and order of subproblems (although in the study in ref [17], the target class of problems were NK landscapes with nearest-neighbor interactions). Except for a few isolated cases, the use of advanced neighborhoods did not seem to improve the situation for any algorithm.

5.3 Fitness Distance Correlation Results

The relationship between fitness distance correlation for local optima using the two distance measures and the actual difficulty when solving instances of separable problems with $k = 3$ using a GA is shown in Table 3a. In the case of FDC, as instance difficulty increases the measure should also increase. The results show a strong relationship when using bit distance, with it able to accurately rank the instances. The results using partition distance are also good, only showing a problem differentiating between the 25% and 10% easiest instances.

When solving the aforementioned instances with hBOA, FDC when using both distance measures is able to accurately rank instance difficulty, as shown in Table 3b. As the class of instances becomes more difficult, the fitness distance correlation increases in all cases. However, Table 3c shows that when solving the same instances with $\text{ECGA}_{perfect}$, the measures are unable to accurately differentiate between the harder classes of instances.

To explore the effects of increasing problem difficulty on FDC for local optima, instances of size $n = 120$ and $k = 5$ were examined. Table 4 shows that for these harder instances, FDC with both distance measures was able to accurately rank all the instances. The results for FDC for local optima show that FDC was able to accurately rank problem difficulty for all algorithms using both distance measures when $k = 5$. Only when using $\text{ECGA}_{perfect}$ on the smaller instances was FDC unable to accurately rank instance difficulty.

6. SUMMARY AND CONCLUSIONS

Most common problem difficulty measures assume standard bit flip neighborhoods, where the distance between solutions is measured with Hamming distance. However, many advanced evolutionary algorithms use more complex search operators than simple bit flips, which can result in more complex neighborhoods. By modifying some common measures of problem difficulty to take into account more complex neighborhoods, this paper attempted to increase their correspondence with the actual difficulty of problem instances when solved by evolutionary algorithms. These modified difficulty measures were then used on separable problems and compared to the actual computational requirements of the simple GA, hBOA and an ideal version of ECGA using a perfect model structure.

Table 1: Correlation length using different neighborhood types vs instance difficulty for randomly separable problems of $n = 120$ and $k = 3$. The ranking of the measures is shown in brackets.

(a) Separable problems solved with GA of $n = 120$, $k = 3$

desc. of instances	DHC steps for GA	correlation length using bit flip	correlation length using random partition	correlation length using set partition
10% easiest	3130.5	27.722(1)	13.882(1)	29.415(5)
25% easiest	3577.6	27.668(3)	13.851(3)	29.424(1)
50% easiest	4165	27.674(2)	13.855(2)	29.420(3)
all instances	5655.3	27.603(4)	13.818(4)	29.417(4)
50% hardest	7145.6	27.531(5)	13.781(5)	29.413(6)
25% hardest	8545.6	27.519(6)	13.770(6)	29.407(7)
10% hardest	10556	27.479(7)	13.740(7)	29.422(2)

(b) Separable problems solved with hBOA of $n = 120$, $k = 3$

desc. of instances	DHC steps for hBOA	correlation length using bit flip	correlation length using random partition	correlation length using set partition
10% easiest	4371.6	27.817(1)	13.922(1)	29.417(4)
25% easiest	4645.4	27.719(2)	13.874(2)	29.419(2)
50% easiest	4890.6	27.685(3)	13.855(3)	29.414(6)
all instances	5540.4	27.603(4)	13.818(4)	29.417(5)
50% hardest	6190.3	27.520(6)	13.781(6)	29.419(1)
25% hardest	6798	27.504(7)	13.777(7)	29.418(3)
10% hardest	7828.5	27.556(5)	13.800(5)	29.412(7)

(c) Separable problems solved with ECGA$_{perfect}$ of $n = 120$, $k = 3$

desc. of instances	DHC steps for ECGA$_{perfect}$	correlation length using bit flip	correlation length using random partition	correlation length using set partition
10% easiest	1487.8	27.581(4)	13.804(5)	29.427(1)
25% easiest	1569.9	27.556(7)	13.794(7)	29.426(2)
50% easiest	1669.3	27.581(5)	13.805(4)	29.426(3)
all instances	1873	27.603(3)	13.818(3)	29.417(5)
50% hardest	2076.7	27.624(2)	13.831(2)	29.407(7)
25% hardest	2201.5	27.577(6)	13.803(6)	29.411(6)
10% hardest	2316.3	27.745(1)	13.875(1)	29.421(4)

For correlation length based on Hamming distance, the results show a weak relationship between the difficulty of instances solved by the GA. As problem difficulty increases, this correlation seems to get even weaker. This pattern is also repeated for hBOA. Unfortunately, using the more advanced neighborhoods based on linkage information about the problem did not seem to improve the results. The results are even worse with ECGA$_{perfect}$, with correlation length using set partition neighborhood actually ranking them in reverse of their actual difficulty. Fitness distance correlation had noisy results using both distance measures when ranking difficulties for the algorithms tested when $k = 3$, but as problem difficulty increased it was able to accurately rank all of the different classes of instance difficulty.

We expected that using more advanced neighborhoods in difficulty measures would help improve their accuracy in predicting the difficulty of solving these problems with advanced evolutionary algorithms. However, incorporating a more advanced neighborhood into correlation length did not seem to help at all. This could possibly be due to correlation length not measuring what is actually making some of the instances more difficult than others. In a similar fashion, the more advanced distance metric in FDC was only able to match the simple Hamming distance measure for accuracy.

Many critical studies exist that point out that some of the most common measures of problem difficulty for evolutionary algorithms and other metaheuristics have only little to do with the actual problem difficulty. Since it turns out that using advanced neighborhood structures that more closely correspond to the operators used in evolutionary algorithm does not seem to improve the results, the question of what measures to use to assess problem difficulty remains open. One way to tackle this question would be to learn from the theoretical studies of scalability of GAs [6, 27], which suggest signal-to-noise ratio, scaling, and fluctuating crosstalk as some of the major factors influencing problem difficulty. Nonetheless, these factors and other problem difficulty measures cannot be studied in isolation; otherwise, each measure will only have a limited scope and the potential for misleading results will remain great.

Acknowledgments

This project was sponsored by the National Science Foundation under CAREER grant ECS-0547013 and by the University of Missouri in St. Louis through the High Performance Computing Collaboratory sponsored by Information Technology Services, and the Research Award and Research Board programs. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily re-

Table 2: Correlation length using different neighborhood types vs instance difficulty for randomly separable problems of $n = 120$ and $k = 5$. The ranking of the measures is shown in brackets.

(a) Separable problems solved with GA of $n = 120$, $k = 5$

desc. of instances	DHC steps for GA	correlation length using bit flip	correlation length using random partition	correlation length using set partition
10% easiest	13328	19.139(2)	6.399(1)	19.435(7)
25% easiest	16877	19.134(4)	6.395(2)	19.448(1)
50% easiest	21211	19.128(5)	6.395(3)	19.446(5)
all instances	33457	19.137(3)	6.395(4)	19.447(4)
50% hardest	45703	19.145(1)	6.394(5)	19.448(2)
25% hardest	56967	19.124(6)	6.387(6)	19.447(3)
10% hardest	71993	19.073(7)	6.373(7)	19.438(6)

(b) Separable problems solved with hBOA of $n = 120$, $k = 5$

desc. of instances	DHC steps for hBOA	correlation length using bit flip	correlation length using random partition	correlation length using set partition
10% easiest	7482.9	19.207(1)	6.416(1)	19.444(5)
25% easiest	7945.3	19.203(2)	6.414(2)	19.440(7)
50% easiest	8399.8	19.179(3)	6.408(3)	19.447(1)
all instances	9311.6	19.137(4)	6.395(4)	19.447(2)
50% hardest	10223	19.095(6)	6.381(6)	19.447(3)
25% hardest	10900	19.066(7)	6.374(7)	19.444(4)
10% hardest	11806	19.119(5)	6.390(5)	19.441(6)

(c) Separable problems solved with ECGA$_{perfect}$ of $n = 120$, $k = 5$

desc. of instances	DHC steps for ECGA$_{perfect}$	correlation length using bit flip	correlation length using random partition	correlation length using set partition
10% easiest	2128.3	19.157(2)	6.396(3)	19.437(7)
25% easiest	2273.6	19.129(5)	6.390(5)	19.443(6)
50% easiest	2428.9	19.112(7)	6.386(6)	19.445(5)
all instances	2770.3	19.137(4)	6.395(4)	19.447(4)
50% hardest	3111.7	19.162(1)	6.403(1)	19.449(3)
25% hardest	3415.5	19.153(3)	6.399(2)	19.452(2)
10% hardest	3869.8	19.121(6)	6.383(7)	19.452(1)

flect the views of the National Science Foundation, the Air Force Office of Scientific Research, or the U.S. Government.

7. REFERENCES

[1] E. Angel and V. Zissimopoulos. Autocorrelation coefficient for the graph partitioning problem. *Theoretical Computer Science*, 191:229–243, 1998.

[2] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[3] D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. Technical Report MSR-TR-97-07, Microsoft Research, Redmond, WA, 1997.

[4] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In M. I. Jordan, editor, *Graphical models*, pages 421–459. MIT Press, 1999.

[5] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.

[6] D. E. Goldberg. *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer, 2002.

[7] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[8] G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.

[9] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[10] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. *International Conf. on Genetic Algorithms (ICGA-95)*, pages 184–192, 1995.

[11] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Boston, MA, 2002.

[12] P. Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12(3):303–325, 2004.

[13] P. Merz and B. Freisleben. Greedy and local search heuristics for unconstrained binary quadratic programming. *Journal of Heuristics*, 8(2):197–213, 2002.

[14] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature*, pages 178–187, 1996.

[15] B. Naudts and L. Kallel. Some facts about so called GA-hardness measures. Technical Report 379, Ecole Polytechnique, CMAP, France, 1998.

[16] M. Pelikan. *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*. Springer-Verlag, 2005.

[17] M. Pelikan. NK landscapes, problem difficulty, and hybrid evolutionary algorithms. In M. Pelikan and J. Branke,

Table 3: Fitness distance correlation with local optima using different neighborhood types vs instance difficulty for randomly separable problems of $n = 120$ and $k = 3$. The ranking of the measures is shown in brackets.

(a) Separable problems solved with GA of $n = 120$, $k = 3$

desc. of instances	DHC for GA	FDC bit distance	FDC partition
10% easiest	3130.5	-0.65915(7)	-0.67115(6)
25% easiest	3577.6	-0.65378(6)	-0.67190(7)
50% easiest	4165	-0.64500(5)	-0.66583(5)
all instances	5655.3	-0.62917(4)	-0.65771(4)
50% hardest	7145.6	-0.61334(3)	-0.64959(3)
25% hardest	8545.6	-0.60603(2)	-0.64778(2)
10% hardest	10556	-0.59750(1)	-0.64301(1)

(b) Separable problems solved with hBOA of $n = 120$, $k = 3$

desc. of instances	DHC for hBOA	FDC bit distance	FDC partition
10% easiest	4371.6	-0.65842(7)	-0.68031(7)
25% easiest	4645.4	-0.64346(6)	-0.67007(6)
50% easiest	4890.6	-0.63894(5)	-0.66447(5)
all instances	5540.4	-0.62917(4)	-0.65771(4)
50% hardest	6190.3	-0.61940(3)	-0.65095(3)
25% hardest	6798	-0.61141(2)	-0.64588(2)
10% hardest	7828.5	-0.60218(1)	-0.64098(1)

(c) Separable problems solved with $\text{ECGA}_{perfect}$ of $n = 120$, $k = 3$

desc. of instances	DHC for ECGA_p	FDC bit distance	FDC partition
10% easiest	1487.8	-0.64460(7)	-0.67060(7)
25% easiest	1569.9	-0.63868(6)	-0.66548(6)
50% easiest	1669.3	-0.63145(5)	-0.66049(5)
all instances	1873	-0.62917(3)	-0.65771(3)
50% hardest	2076.7	-0.62689(2)	-0.65493(2)
25% hardest	2201.5	-0.62623(1)	-0.65466(1)
10% hardest	2316.3	-0.63102(4)	-0.65841(4)

Table 4: Fitness distance correlation with local optima using different neighborhood types vs instance difficulty for randomly separable problems of $n = 120$ and $k = 5$. The ranking of the measures is shown in brackets.

(a) Separable problems solved with GA of $n = 120$, $k = 5$

desc. of instances	DHC for GA	FDC bit dist.	FDC partition
10% easiest	13328	-0.40403(7)	-0.44834(7)
25% easiest	16877	-0.39321(6)	-0.44414(6)
50% easiest	21211	-0.38170(5)	-0.44055(5)
all instances	33457	-0.36075(4)	-0.43371(4)
50% hardest	45703	-0.33981(3)	-0.42687(3)
25% hardest	56967	-0.32773(2)	-0.42079(2)
10% hardest	71993	-0.30975(1)	-0.41416(1)

(b) Separable problems solved with hBOA of $n = 120$, $k = 5$

desc. of instances	DHC for hBOA	FDC bit distance	FDC partition
10% easiest	7482.9	-0.37490(7)	-0.45194(7)
25% easiest	7945.3	-0.37354(6)	-0.44697(6)
50% easiest	8399.8	-0.36649(5)	-0.43962(5)
all instances	9311.6	-0.36075(4)	-0.43371(4)
50% hardest	10223	-0.35501(3)	-0.42780(3)
25% hardest	10900	-0.35022(2)	-0.42226(2)
10% hardest	11806	-0.34699(1)	-0.41737(1)

(c) Separable problems solved with $\text{ECGA}_{perfect}$ of $n = 120$, $k = 5$

desc. of instances	DHC for ECGA_p	FDC bit distance	FDC partition
10% easiest	2128.3	-0.36917(7)	-0.443970(7)
25% easiest	2273.6	-0.36651(6)	-0.438740(6)
50% easiest	2428.9	-0.36419(5)	-0.435860(5)
all instances	2770.3	-0.36075(4)	-0.433710(4)
50% hardest	3111.7	-0.35731(3)	-0.431560(3)
25% hardest	3415.5	-0.35088(2)	-0.426320(2)
10% hardest	3869.8	-0.34526(1)	-0.421470(1)

editors, *Genetic and Evolutionary Computation Conf. (GECCO-2010)*, pages 665–672. ACM, 2010.

[18] M. Pelikan and D. E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. *Genetic and Evolutionary Computation Conf. (GECCO-2001)*, pages 511–518, 2001.

[19] M. Pelikan and D. E. Goldberg. A hierarchy machine: Learning to optimize from nature and humans. *Complexity*, 8(5):36–45, 2003.

[20] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[21] M. Pelikan, K. Sastry, M. V. Butz, and D. E. Goldberg. Performance of evolutionary algorithms on random decomposable problems. In *PPSN*, pages 788–797, 2006.

[22] M. Pelikan, K. Sastry, D. E. Goldberg, M. V. Butz, and M. Hauschild. Performance of evolutionary algorithms on nk landscapes with nearest neighbor interactions and tunable overlap. MEDAL Report No. 2009002, Missouri Estimation of Distribution Algorithms Laboratory, University of Missour–St. Louis, St. Louis, MO, 2009.

[23] S. Rochet, G. Venturini, M. Slimane, and E. M. E. Kharoubi. A critical and empirical study of epistasis measures for predicting ga performances: A summary. In *Selected Papers from the Third European Conference on Artificial Evolution*, AE '97, pages 275–286, London, UK, 1998. Springer-Verlag.

[24] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, Department of General Engineering, Urbana, IL, 2001.

[25] P. F. Stadler. Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20:1–45, 1996.

[26] A. M. Sutton, L. D. Whitley, and A. E. Howe. A polynomial time computation of the exact correlation structure of k-satisfiability landscapes. *Genetic and Evolutionary Computation Conf. (GECCO-2009)*, pages 365–372, 2009.

[27] D. Thierens, D. E. Goldberg, and A. G. Pereira. Domino convergence, drift, and the temporal-salience structure of problems. *International Conf. on Evolutionary Computation (ICEC-98)*, pages 535–540, 1998.

[28] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990.