Toward More Powerful Recombinations *

Andrew B. Kahng and Byung Ro Moon
UCLA Computer Science Department
Los Angeles, CA 90095-1596
abk@cs.ucla.edu, moon@nexus2.cs.ucla.edu

Abstract

This paper suggests a flexible framework for n-dimensional crossover, consisting of *cutting*, *classification*, and *copying* of genes. We prove that under this framework, any *cutting strategy* generates two equivalence classes of genes, making the framework appropriate as a crossover scheme. Three notable features of this framework are: (i) it enables more effective use of genes' geographical linkage, (ii) it enables more diverse ways of cutting than traditional multi-point crossover on linear strings or existing 2-dimensional crossover schemes, and (iii) it can be readily used within most existing genetic algorithm implementations, i.e., the underlying problem need not be inherently multi-dimensional. We provide guidelines for designing a crossover strategy under this framework, along with two example crossovers. Experimental results show that one can design new crossovers under this framework which outperform traditional crossover on linear strings, uniform crossover, and existing two-dimensional crossovers.

1 INTRODUCTION

The linear string is a symbolic feature of the genetic algorithm (GA) approach, and most GA implementations have been based on linear encodings [13]. At the same time, linear encodings have also been viewed as limiting GA performance [9, 2, 19]. For example, with graph problems it is inevitable that considerable adjacency is lost when mapping a graph (e.g., a multi-dimensional mesh) onto a linear string [7]. Cohoon and Paris [9] proposed a two-dimensional rectangle-style crossover for VLSI circuit placement; this scheme chooses a small rectangle and copies the genes in the rectangle from one parent into the offspring; the remaining genes are copied from the other parent. Anderson-Jones-Ryan [2] suggested block-uniform crossover on two-dimensional grid-type chromosomes; this approach tesselates the chromosome into $i \times j$ blocks; for each block genes in the block are copied as a group from a uniformly-selected parent. Bui and Moon [19, 7] suggested Z3, an n-dimensional generalization of traditional crossover which chooses k crossover points on the n-dimensional chromosome. Intuitively, the key merit of two- or higher-dimensional encodings is that they contain more geographical linkages of genes than the traditional linear encoding [7].

In the following discussion, we assume an l-ary n-cube structured chromosome. This structure is analogous to, e.g., the node structure of interconnection networks studied in [10, 1]. We use locus-based encodings with this chromosome structure. For example, linear locus-based encodings for graph problems mostly map each vertex to a fixed corresponding position (locus) on the linear string [11, 20, 18, 16, 21, 19].

The term "geographical linkage" is somewhat vague; here we describe the term as it is used below. If two genes are more likely to move together in a crossover (from a parent to the offspring) than are a random pair of genes, then we say that the two genes have stronger-than-average geographical linkage. The strength of geographical linkage between two genes can differ according to the encoding. Certainly, the dimension of the encoding is an important factor, as is the locus assignment of genes. Moreover, even given a fixed encoding, the geographical linkages of genes will vary depending on the crossover scheme. A given combination of encoding and crossover corresponds to a spectrum of geographical linkages over all pairs of genes.

Although multi-dimensional encodings can preserve more geographical linkages of genes, multi-dimensional crossovers to date have a potential weakness, namely, crossover diversity and its associated schema-

*This work was supported by NSF YI award MIP-9257982 and grant MIP-9223740. The authors are with the UCLA VLSI CAD Laboratory and the UCLA Commotion Laboratory (NSF CDA-9303148).

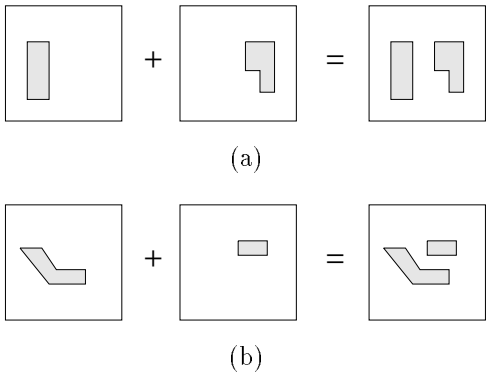

Figure 1: Two examples of 2-dimensional schema generation

generation power. Consider single-point crossover on a linear string of size N, where $N = l^2$ without loss of generality: there are $l^2 - 1$ crossover operators. On the other hand, if the chromosome is encoded on a two-dimensional $l \times l$ grid and cutting is done by a horizontal or vertical straight line, then there are $2(l-1)$ crossover operators. The diversity of the two-dimensional crossover is not comparable to that of traditional linear crossover. When only axis-parallel hyperplanes are used for cutting, there are fewer and fewer crossover operators as the encoding dimension increases. This phenomenon still holds as the number of crossover points is increased, and so one might expect that the two-dimensional crossover is inferior to the linear crossover in performance. However, evidence has been reported that such straight-line-based two-dimensional crossovers outperform the linear crossover [2, 7]. A key reason for this seems to be that the advantage of preserving more geographical linkage in the encoding compensates for any potential disadvantage in weaker crossover diversity.

Figure 1 shows two examples of schema generation by 2-dimensional crossovers. The shaded areas in each chromosome represent specific-symbol positions within schemata. A notable common feature of all preceding 2-dimensional crossovers [9, 8, 2, 7] is that they use only horizontal and vertical straight lines to cut the chromosomes. Using such "slicing" mechanisms, it is possible to generate new schemata as in (a), but not as in (b) since the two schemata shown overlap both horizontally and vertically. With this in mind, the central contribution of this paper is a new framework for multi-dimensional crossovers which (i) uses multi-dimensional encodings to better exploit useful geographical linkages, and (ii) provides more powerful cuttings to improve diversity in schema generation. In the framework, a crossover is performed by k cutting *(hyper)surfaces* each dividing all loci into two disjoint subsets in a relatively unrestricted manner.

The remainder of this paper is organized as follows. In Section 2, we suggest a formal framework for multi-dimensional crossovers. We show that all loci are classified into exactly two equivalence classes under the new framework, regardless of the chromosomal dimension or the number of cuts. No particular cutting strategy is fixed, but we do suggest guidelines for determining a useful cutting strategy. We also provide guidelines for modifying existing GA implementations to incorporate the new crossover framework with minimal change to existing code. Section 3 provides two exemplary cutting strategies for two- and three-dimensional encodings. Section 4 provides various experimental results on the graph bisection problem, and we conclude in Section 5 with ongoing and future research directions.

2 THE NEW FRAMEWORK

2.1 PRELIMINARIES

Consider the closed and continuous region $U^n \subset \Re^n$ with $U = [0, l-1]$.

Definition 1 *A* cutting surface *is a hypersurface that divides U^n into exactly two connected subregions.*

Assume that we have k cutting surfaces, and let C be the intersection of U^n with the union of the cutting surfaces. The space $U^n - C$ is divided into a number of subregions by the k cutting surfaces, such that any continuous path from a point in one subregion to a point in another subregion must cross at least one cutting surface. Define a relation R_e as follows:

Definition 2 *For two points $x, y \in U^n - C$, xR_ey if and only if there exists a continuous path in U^n from x to y which makes an even number of intersections, or* crosses, *with cutting surfaces.*

It is obvious that R_e is reflexive (xR_ex), commutative (xR_ey iff yR_ex), and transitive (if xR_ey and yR_ez, then xR_ez). Thus, we have

Fact 1 *R_e is an equivalence relation.*

To establish a role for cutting surfaces within the GA approach, we show that there are only two equivalence classes of points in $U^n - C$ induced by the equivalence relation R_e. The following fact says that if there exists a path in U^n from x to y with an even number of crosses, then any other path in U^n from x to y also has an even number of crosses.

Fact 2 *If xR_ey, any path in U^n from x to y has an even number of crosses.*

Proof: By the definition of R_e, there exists a path from x to y with an even number of crosses. Let its multiset of crossed cutting surfaces in the path be $A =$

$\{A_1, A_2, \ldots, A_{2i}\}$ for some i; note that any cutting surface is listed once for each time it is crossed. Assume toward a contradiction that there exists a path from x to y with an odd number of crosses, and let the multiset of crossed surfaces be $B = \{B_1, B_2, \ldots, B_{2j+1}\}$ for some j.

Observe that if a given cutting surface P appears an even number of times in A (or B), then x and y must be in the same halfspace with respect to P. If P appears an odd number of times, x and y must be on the opposite sides of P.

Now consider the multisets A and B. If any cutting surface P is listed twice in A (or B), delete both instances of P from A (or B); repeat this process until no cutting surface is listed more than once in either A or B. This yields the reduced sets A' and B'. Since A' has an even number of elements and B' has an odd number of elements, there exists some cutting surface Q such that $Q \in A' - B'$ or $Q \in B' - A'$. If $Q \in A' - B'$, then membership in A' says that x and y are on opposite sides of Q, but membership in B' says that x and y are on the same side, a contradiction. An analogous contradiction occurs when $Q \in B' - A'$. ∎

As an immediate corollary, we have

Fact 3 *For any number k of cutting surfaces, the equivalence relation R_e induces exactly two equivalence classes of points in $U^n - C$.*

2.2 THE FRAMEWORK

Assume without loss of generality that the number of genes N is l^n. Define the chromosome domain space to be D^n where $D = \{0, 1, \ldots, l-1\}$. That is, assume an l-ary n-cube chromosomal structure. Each locus is represented by an n-vector $(a_1, a_2, \ldots a_n)$, with $a_i \in D$ for $i = 1, 2, \ldots, n$. The space D^n is a discrete analog of the region U^n defined in Section 2.1. Among all possible cutting surfaces, consider only the in $U^n - D^n$. That is, a cutting surface never touches the elements in D^n. It is clear that a cutting surface in $U^n - D^n$ divides the domain space D^n into two disjoint subspaces. We will use this type of cutting surface as the counterpart of a crossover point in traditional GAs.

We choose k such cutting surfaces. It is obvious that the three results of Section 2.1 still hold on points (loci) in D^n since $D^n \subset U^n - C$. Thus, based on Fact 3 we can classify all loci in D^n into two equivalence classes (say class 0 and class 1) associated with those cutting surfaces. Now we generate an offspring as follows. If a locus belongs to the class 0, copy the corresponding gene from the parent 0. If the locus belongs to the class 1, copy from the parent 1.

Note that the framework does not specify any cutting strategy, nor any chromosomal dimension. Details of the cutting strategies will typically be fixed by users based on attributes of the underlying application, as well as implementation constraints (e.g., the complexity of the new crossover algorithm depends mostly on that of its cutting strategy). In Section 3, we provide examples of cutting strategies.

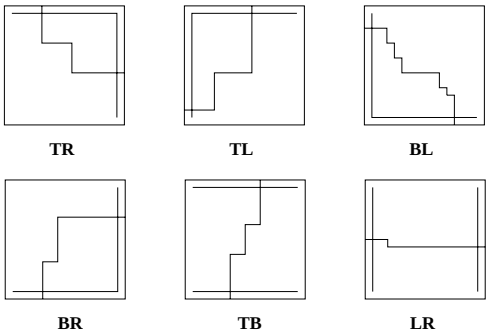

Figure 2: Six basic types of cutting patterns

2.3 REPLACING EXISTING CROSSOVERS

Our framework can be easily used with most existing GA implementations. One does not have to change the encoding scheme (which typically requires changes to data structures and consequently a wide range of modifications); rather, only the crossover part can be modified, leaving other code intact. Observe that a linear string is essentially a linear array. One must first generate a mapping from the linear array to the multidimensional encoding. The first stage of crossover then generates a masking array (l^n for an n-dimensional encoding) which specifies each gene's class assignment (0 or 1). This n-dimensional masking array is inverse-mapped to the linear array for the purpose of copying genes from parents to offspring. Thus, n-dimensional encoding is "imaginary": it exists temporarily just for crossover.

3 SAMPLE CUTTING STRATEGIES

In this section, we provide sample cutting strategies in two and three dimensions. Notice from Section 2.1 that Definition 1 is the only restriction on the cutting strategy, i.e., any cutting strategy satisfying Definition 1 will yield a well-defined crossover. However, since a complicated cutting strategy may make implementation more difficult than necessary, we initially seek relatively simple crossover schemes which may yet provide reasonably diverse crossover operators.

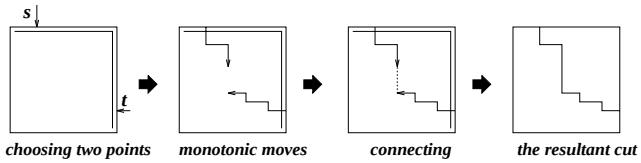

Figure 3: An example of TR-type cutting

3.1 A TWO-DIMENSIONAL EXAMPLE

With a chromosome size of $N = l^2$, a two-dimensional chromosome can be viewed as an $l \times l$ grid or matrix. Let us represent a locus on the chromosome by $(a_1, a_2) \in D \times D$ as in Section 2.2. After making the cuts as described below, we *classify* the loci into two equivalence classes by producing an $l \times l$ binary masking array, denoted *class*. The *copying* phase is then straightforward: for each locus (i, j) of the offspring, copy the gene (i, j) from parent 0 if $class[i, j]$ is 0, and from parent 1 otherwise. Note that classification and copying are common to any crossover under this framework, independent of the cutting strategy (assuming that a consistent data structure is used to represent the cut).

To make a cut, we choose two points at random anywhere on the four edges of the chromosome, but do not allow both points to be on the same edge. Let the four edges be T (Top), R (Right), B (Bottom), and L (Left). There are six types of cuts related to the two points: TR, TL, BR, BL, TB, and LR (Figure 2). We describe the cutting strategy for the TR type; other types are symmetric or similar. Two numbers s and t are selected from the set $\{0, 1, \ldots, l-2\}$. This means that the cut starts between the loci $(0, s)$ and $(0, s+1)$ on the T side, and between the loci $(t, l-1)$ and $(t+1, l-1)$ on the R side (we assume that the "origin" is at the top-left as is standard for matrix representations). The cutting corresponds to making random monotonic paths at the same speed from both points until the paths can be connected by a horizontal or vertical segment.[1] Figure 3 shows an example of TR-type cutting.

We randomly choose k such cuttings and call the resulting crossover a *k-cut crossover* or *k-cut geographic crossover*. Observe that the one-dimensional case reduces to the traditional k-point crossover on linear strings. Figure 4 shows various examples of k-cut geographic crossover operators in two dimensions. From Fact 3, we know that these cuts partition all loci into two equivalence classes; these are shown as shaded and unshaded regions. Figure 5 shows the two-dimensional classification algorithm. In the algorithm, *cuts* $[i, j,$ HOR$]$ (*cuts* $[i, j,$ VER$]$) contains the number of cutting lines passing between the loci (i, j) and $(i+1, j)$ (loci (i, j) and $(i, j+1)$).

[1] This is done by coin tosses. In TR-type cuttings, the cutting from the T side moves down or right according to the coin tosses; that from the R side moves up or left.

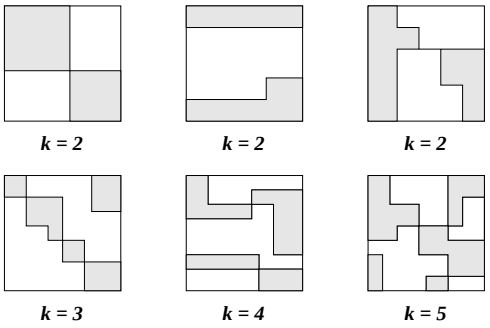

Figure 4: Examples of crossover operators with various values of k

Algorithm Classify
Input: $(l-1) \times (l-1) \times 2$ array *cuts*
Output: $l \times l$ array *class*
Constants: HOR = 0, VER = 1

```
class [0, 0] = 0;
for j = 1 to l − 1 {
    if (cuts [0, j − 1, VER] % 2 = 0)
        then class [0, j] = class [0, j − 1];
        else class [0, j] = ¬ class [0, j − 1];
}

for i = 1 to l − 1 {
    for j = 0 to l − 1 {
        if (cuts [i − 1, j, HOR] % 2 = 0)
            then class [i, j] = class [i − 1, j];
            else class [i, j] = ¬ class [i − 1, j];
    }
}
```

Figure 5: Classification algorithm

We believe that this crossover generates more diverse schemata than previous crossovers that are based on axis-parallel cuts; for example, note that the schema shown in Figure 1(b) can be generated. The lemma below indicates the number of possible cuttings achievable by this strategy.

Lemma 1 *On an $l \times l$ chromosome, the total number of possible cuttings by the above cutting strategy is $f(l) = 4\binom{2l-1}{l-1} - 2(l+1)$.*

Proof: Omitted. ∎

There are $N-1$ possible cuts of a linear encoding, and hence the number of k-point crossover operators on a linear encoding is $\binom{N-1}{k}$. The number of k-cut geographic crossover operators (in the two-dimensional case) is $\binom{f(l)}{k} - d$ where d is the number of duplica-

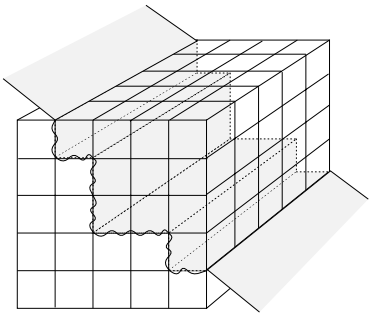

Figure 6: A cutting example in three dimensions, i.e., in a 5-ary 3-cube chromosome

tions (i.e., when the same classification is possible via different combinations of cuttings). We believe that $\binom{f(l)}{k} - d$ is considerably larger than $\binom{N-1}{k}$ for any fixed k, as the function $f(l)$ is exponential in l and consequently exponential in N. However, counting the number of duplications is difficult and we do not yet have an analysis.

3.2 A THREE-DIMENSIONAL EXAMPLE

With a chromosome size of $N = l^3$, a three-dimensional chromosome can be viewed as an $l \times l \times l$ grid or matrix, and an analogous three-dimensional cutting strategy can be devised. Here a cut corresponds to a surface instead of a line. For practical reasons, our implementation restricts a cutting surface to be parallel to one of the three coordinate axes (thus, there are three symmetric types of cutting surfaces, each parallel to some coordinate axis).

Let the three dimensions be X, Y, and Z. A cutting surface parallel to, e.g., Z, will cut all XY planes of the three-dimensional chromosome (there are l such planes) in the same way. That is, the cutting strategy used for two-dimensional chromosomes is simply replicated l times to yield a three-dimensional cutting. There are a total of $3 \times 6 = 18$ cutting patterns for this three-dimensional crossover. Figure 6 shows an example of a cutting surface on a three-dimensional chromosome. The dimension of a cutting surface will increase with the encoding dimension, and similar constructions based on lower-dimensional cuts can be easily implemented.

For classification, the array *cuts* now has four dimensions: three to represent the loci, and one for the three boundaries of each locus. For space reasons, we omit a formal template of the cutting algorithm. Initialization of data structures requires $\Theta(N)$ time, and the actual cutting takes $\Theta(kN^{\frac{1}{2}})$ time and $\Theta(kN^{\frac{2}{3}})$ time for the two-dimensional and three-dimensional schemes, respectively.

4 EXPERIMENTAL RESULTS

We have tested the utility of geographic crossovers versus other crossovers on low-dimensional chromosomal representations using the *graph bisection* problem. Given a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, a *bisection* of G is a partitioning of the vertex set V into two disjoint subsets with the difference between cardinalities of the two subsets at most 1. The *cut size* of a bisection is the number of edges whose endpoints are in different subsets in the bisection. The graph bisection problem seeks a bisection with minimum cut size and is well-known to be NP-hard [12].

We tested our GA implementations on five types of benchmark graphs: random graphs, random geometric graphs, random regular graphs with known small bisection widths, caterpillar graphs, and grid graphs. These have been extensively studied in the discrete algorithms and optimization literatures, e.g., [14, 15, 6, 19].[2] All implementations in this paper used steady-state GAs with five cuts for crossover unless otherwise noted; we use the code of [6] except for the crossover part.[3] The GAs stop when 80% of the population is occupied by solutions of the same quality. All results are averages (bisection cut sizes) over 50 runs on each graph.

4.1 PURE GEOGRAPHIC CROSSOVERS

In this subsection, we compare the 2-dimensional and 3-dimensional k-cut geographic crossovers described in Sections 3.1 and 3.2 with the traditional k-point

[2] Briefly, a *random graph* is a graph in which an edge is randomly and independently placed between every two vertices with probability p. G$n.d$ represents a random graph on n vertices where the probability p is chosen so that the expected vertex degree $p(n-1)$ is equal to d. A *random geometric graph* is a graph whose vertices are uniformly random in the unit square, with an edge between two vertices if their Euclidean distance is $\leq t$. U$n.d$ represents a random geometric graph on n vertices with expected vertex degree equal to d. The *caterpillar graph* is a graph with sequentially connected articulation points each having the same number of legs. cat.n represents a caterpillar graph of n vertices. A *regular graph* is a random graph in which each vertex has the same degree d (we use $d = 3$ in all our test cases), and whose optimal cut size b is both significantly smaller than for similar-sized random graphs and also known with high probability (such constructions were first proposed by Bui et al. [3]). breg$n.b$ represents a regular graph of n vertices for which the optimal bisection cut size is b with probability $1 - o(1)$. grid$n.b$ represents a two-dimensional grid graph on n vertices whose optimal cut size is known to be b.

[3] Note that [6] studied hybrid-type GAs, but that we remove the local optimization part to isolate the effect of crossovers. Thus the quality of results for all GA variations here is actually worse than those reported in [6] for hybrid-type GAs.

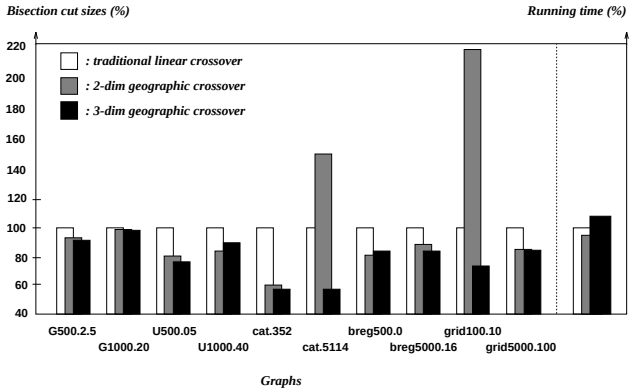

Figure 7: Comparison of crossovers on different dimensions

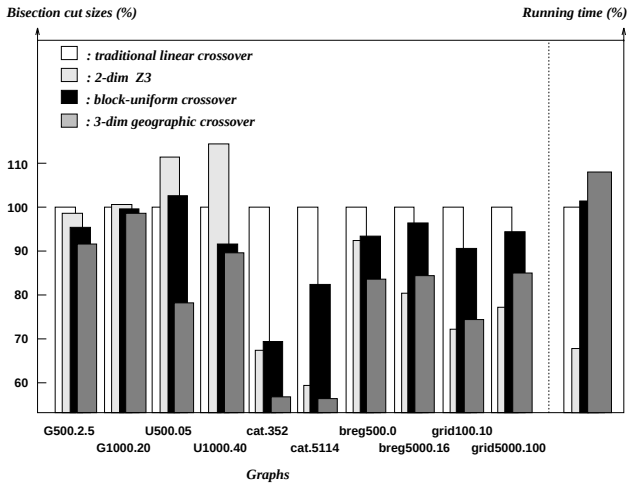

Figure 8: Performances of Z3, block-uniform crossover, and 3-dimensional geographic crossover relative to traditional linear crossover

crossover. Figure 7 shows the relative results (average bisection cut sizes) of GAs using two-dimensional and three-dimensional geographic crossover to the GA using traditional linear crossover (with $k = 5$). The bisection cut sizes are normalized to those obtained with traditional k-point crossover. The data clearly show that the two-dimensional geographic crossover generally outperforms the traditional linear one, and that the three-dimensional geographic crossover outperforms the two-dimensional one. It is notable that the two-dimensional geographic crossover showed significantly worse results for some graphs; we do not have a reasonable explanation for this. Runtimes remained essentially constant over all dimensions of models. We believe that the main reason for this improvement lies in the power of the new crossover operators to exploit geographical linkages of genes and create new schemata.

4.2 COMPARISON WITH 2-DIMENSIONAL CROSSOVERS

In this subsection, we compare the performance of 3-dimensional geographic crossover with existing 2-dimensional crossovers, namely, rectangle-style crossover [9], block-uniform crossover [2], and Z3 [19, 7]. For the graph bisection problem, the crossover of [9] yielded much worse results than the other crossovers and thus we do not include the results. Figure 8 shows the relative performances of Z3, block-uniform crossover, and 3-dimensional geographic crossover versus traditional linear crossover. Overall, three-dimensional geographic crossover performed best.

4.3 HEURISTIC EMBEDDING

For the preceding comparisons, we embedded (encoded) the genes onto multi-dimensional chromosomes in a naive way: row-major order. Bui and Moon showed that the embedding can significantly affect GA performance when a linear encoding is used [4, 5]. We believe that this phenomenon will persist with high-dimensional encodings. Recently, Linial-London-Rabinovich [17] proved that we need a fairly high-dimensional encoding to embed a graph with low distortion. Since a too high-dimensional encoding is hard to implement, a practical alternative is to embed the graph into a low-dimensional mesh, say of dimension two or three, and minimize the distortion in this dimension. Our first method sought to embed the vertices of input graphs onto the $l \times l$ grid so as to minimize the sum of the embedded lengths of edges. If the two endpoints of an edge are located on the loci (x_1, y_1) and (x_2, y_2) in an embedding, the edge length is given by the Euclidean distance $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. We developed a fast $\Theta(N\sqrt{N})$ heuristic for this problem. Profiling using the tool *gprof* on the input G500.2.5 shows that the embedding occupies 12.0% of the total running time. This objective seemed a reasonable candidate for minimizing the distortion caused by embedding, and resulted in improvements as expected.

We also tried another very simple embedding. From a given linear ordering of vertices, first reorder them according to a depth first search (DFS) starting at a random vertex. Then, from this ordering embed the vertices into a 2- or 3-dimensional mesh (the encoding scheme) by row-major order. This embedding obviously takes only linear time; it occupies just 0.1% of the total running time for the graph G500.2.5. Figure 9 shows the performance improvement from the pure 2-dimensional geographic crossover when Euclidean embedding or DFS-row-major embedding is applied. The DFS-row-major embedding clearly dominates the Euclidean embedding, which is strikingly

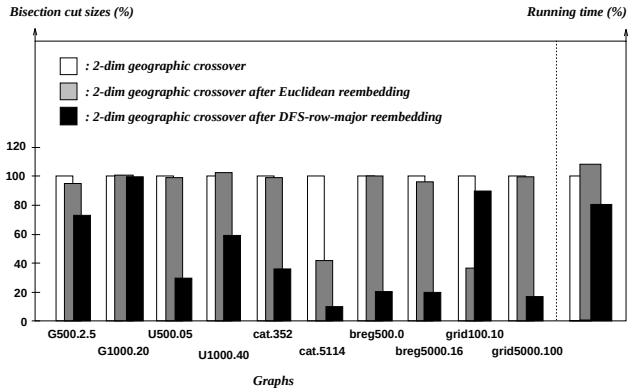

Figure 9: The effect of heuristic 2-dimensional embeddings

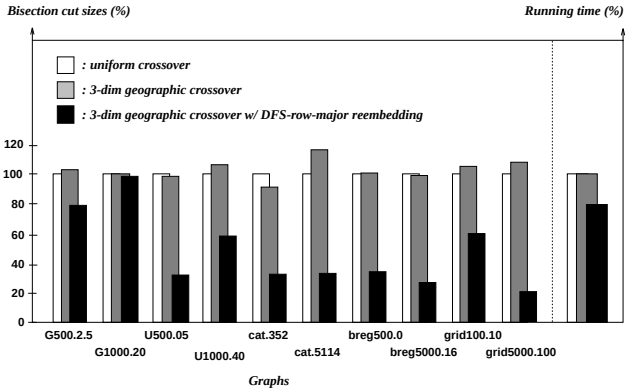

Figure 10: Performances relative to uniform crossover

opposite to what we expected. We strongly suspect that the DFS-row-major embedding is still not the best embedding possible, mainly because of its naive row-major embedding phase.

4.4 COMPARISON WITH UNIFORM CROSSOVER

Uniform crossover is noted for being independent of the loci of genes, i.e., it is independent of the genes' order in the encoding. Theoretically, uniform crossover can preserve any schema and can also generate any schema possible from the two parents. We also implemented and tested uniform crossover and found that it performed slightly better on the average than the pure 3-dimensional geographic crossover. However, when reembedding is applied, further improvement is possible for 3-dimensional geographic crossover which is not possible for uniform crossover.

Recall that the results of Sections 4.1 and 4.2 used given orderings of vertices in the input graphs. Section 4.3 showed that geographic crossovers can benefit from simple reembeddings. Uniform crossover, of course, does not benefit from reembedding or reordering genes because of its independence of the gene locations. Figure 10 shows the performances of pure 3-dimensional geographic crossover and DFS-row-major embedded 3-dimensional geographic crossover, relative to uniform crossover. The 3-dimensional geographic crossover with reembedding performed noticeably better than either the pure three-dimensional geographic crossover or uniform crossover.

5 DISCUSSION AND FUTURE WORK

In this paper, we have proposed a flexible framework for multi-dimensional crossovers and provided some example crossovers under the framework. Experiments on a wide range of graph bisection instances generally showed visible improvement over traditional k-point crossover on linear strings, previous 2-dimensional crossovers, and uniform crossover. We observed that performance improves as the dimension of crossover increases (up to three dimensions).

We believe that the suggested framework is applicable to most GAs that use traditional linear encodings and crossovers. Given a genetic algorithm encoded on a linear string, one may simply relocate the genes onto a two- or higher-dimensional encoding and apply the proposed crossover. As noted in Section 2.3, the multi-dimensional encoding does not have to be "real". In our implementation, we use exactly the same code as for a GA with the traditional linear scheme, except for the crossover routine (i.e., the multi-dimensional encodings are "imaginary"). For crossover, we need only provide mapping information between loci on the linear encoding and loci on the multi-dimensional encoding; the copying is done on the linear string. This helps simplify the algorithm as there is no need for special data structures for graphs with a high-dimensional indexing scheme. We believe that this replacement of the crossover scheme can improve the performance of traditional GAs in many cases; we thus hope to see more experimental results on various problem classes in the future.

Although we noted that geographic crossovers can create diverse schemata, we do not want to be able to create all schemata possible from the two parents (If we did, we would definitely choose uniform crossover). The experimental results of Section 4.4 hint that navigating a promising region of the solution space is more important than obtaining theoretical diversity (power) in the search operator. Our current work seeks to more precisely determine the mechanisms by which the geographic crossover operators, in conjunction with multi-dimensional chromosome structure, improve GA performance. Among the research questions we hope to resolve are the following.

- As mentioned in the introduction, an encoding/crossover pair makes a spectrum of geographical linkages. A formal analysis as to what types of geographical-linkage spectrum are desirable for high-performance GAs remains an open direction for research.
- It is open as to which chromosomal dimension performs best. Although higher-dimensional encodings (whether real or imaginary) can preserve more geographical gene linkages, we suspect that too high a dimension would not perform desirably. We are studying the question of which dimension of encoding is best for a given instance. It is likely that the optimal dimension is somehow dependent on the chromosome size and the input graph topology; interactions with the flexibility of crossover are yet unknown. The interaction of these considerations with the number of cuts used in the crossover is also an open issue.
- In relocating genes onto a multi-dimensional chromosome, the simplest way is via a sequential assignment such as row-major order. Section 4 showed that performance improves when a DFS-row-major reembedding is used for two- and three-dimensional encodings. We suspect that this phenomenon will be consistent for higher-dimensional cases, and hope to perform more detailed investigations in the future. Although DFS reordering proved to be helpful for both linear encodings [19] and multi-dimensional encodings, we do not believe DFS-row-major reembedding is a good approach for the multi-dimensional cases since the row-major embedding is so simplistic. We are considering alternative 2-dimensional and 3-dimensional reembeddings which will hopefully provide further improvement.
- Finally, the l-ary n-cube chromosomal structure may be further generalized to an n-cuboid structure in which each dimension can have different length. This generalization will be helpful in more closely capturing the structure of practical problem instances.

References

[1] A. Agarwal. Limits on interconnection network performance. *IEEE Trans. on Parallel and Distributed Systems*, 2(4):398–412, 1991.

[2] C. A. Anderson, K. F. Jones, and J. Ryan. A two-dimensional genetic algorithm for the ising problem. *Complex Systems*, 5:327–333, 1991.

[3] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[4] T. N. Bui and B. R. Moon. Hyperplane synthesis for genetic algorithms. In *Fifth International Conference on Genetic Algorithms*, pages 102–109, July 1993.

[5] T. N. Bui and B. R. Moon. Analyzing hyperplane synthesis in genetic algorithms using clustered schemata. In *International Conference on Evolutionary Computation*, Oct. 1994. *Lecture Notes in Computer Science*, 866:108-118, Springer-Verlag.

[6] T. N. Bui and B. R. Moon. Graph partitioning and genetic algorithms. 1994. Submitted to *IEEE Trans. on Computers* (in revision).

[7] T. N. Bui and B. R. Moon. On multi-dimensional encoding/crossover, 1995. To appear in *Sixth International Conference on genetic Algorithms*.

[8] P. Chan, P. Mazumder, and K. Shahookar. Macrocell and module placement by genetic adaptive search with bitmap-represented chromosome. *Integration*, 12(1):49–77, 1991.

[9] J. P. Cohoon and W. Paris. Genetic placement. In *IEEE International Conference on Computer-Aided Design*, pages 422–425, 1986.

[10] W. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. on Computers*, 39(6):775–785, 1990.

[11] K. DeJong and W. Spears. Using genetic algorithms to solve NP-complete problems. In *Third International Conference on Genetic Algorithms*, pages 124–132, 1989.

[12] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[13] J. Holland. *Adaptation in Natural and Artificial Systems, 2nd ed.* MIT Press, 1992.

[14] D. S. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part 1, graph partitioning. *Operations Research*, pages 865–892, 1989.

[15] C. Jones. *Vertex and Edge Partitions of Graphs*. PhD thesis, Penn. State Univ., University Park, PA, 1992.

[16] D. Levine. A genetic algorithm for the set partitioning problem. In *Fifth International Conference on Genetic Algorithms*, pages 481–487, July 1993.

[17] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Foundations of Computer Science*, pages 577–591, 1994.

[18] N. Mansour and G. Fox. A hybrid genetic algorithm for task allocation in multicomputers. In *Fourth International Conference on Genetic Algorithms*, pages 466–473, July 1991.

[19] B. R. Moon. *Hybrid Genetic Algorithms with Hyperplane Synthesis: A Theoretical and Empirical Study*. PhD thesis, Pennsylvania State University, University Park, PA, 1994.

[20] A. Murthy and G. Parthasarathy. Clique finding– a genetic approach. In *IEEE Conference on Evolutionary Computation*, pages 18–21, June 1994.

[21] C. Palmer and A. Kershenbaum. Representing trees in genetic algorithms. In *IEEE Conference on Evolutionary Computation*, pages 379–384, June 1994.