Geometric Crossovers for Multiway Graph Partitioning

A. Moraglio, Y. H. Kim, Y. Yoon and B. R. Moon

November 1, 2006

Abstract

Geometric crossover is a representation-independent generalization of the traditional crossover defined using the distance of the solution space. By choosing a distance firmly rooted in the syntax of the solution representation as basis for geometric crossover, one can design new crossovers for any representation. Using a distance tailored to the problem at hand, the formal definition of geometric crossover allows us to design new problem-specific crossovers that embed problem-knowledge in the search.

The standard encoding for multiway graph partitioning is highly redundant: each solution has a number of representations, one for each way of labeling the represented partition. Traditional crossover does not perform well on redundant encodings. We propose a new geometric crossover for graph partitioning based on a labeling-independent distance that filters out the redundancy of the encoding. A correlation analysis of the fitness landscape based on this distance shows that it is well suited to graph partitioning.

A second difficulty with designing a crossover for multiway graph partitioning is that of feasibility: in general recombining feasible partitions does not lead to feasible offspring partitions. We design a new geometric crossover for permutations with repetitions that naturally suits partition problems and test it on the graph partitioning problem. We then combine it with the labeling-independent crossover and obtain a much superior geometric crossover inheriting both advantages. Extensive experimental results are reported.

1 Introduction

Geometric crossover [23] is a representation-independent operator defined over the distance of the search space. Informally, geometric crossover requires the offspring to lie between parents.

The formal definition of geometric crossover enables us to build a representation-independent theory of evolutionary algorithms: it can be shown that an evolutionary algorithm endowed with geometric crossover and selection does a form of convex search regardless of the underlying solution representation, fitness landscape, specific distance, probability distribution of the offspring and selection mechanism. Knowing how geometric crossover searches the search space is very important because it opens the way to identifying a general class of fitness landscapes for which all geometric crossovers work well.

Despite its simple geometric definition, geometric crossover captures the notion of "real-world" crossover: geometric crossover generalizes many pre-existing search operators for the major representations used in practice, such as binary strings and real vectors [23], permutations [25], syntactic trees [24] and sequences [26].

The formal definition of geometric crossover can also be used to guide the design of new specific crossover operators for non-standard representations using distances rooted on the specific representation (e.g., edit distances) as base for geometric crossover [25]. To be effective, specific geometric crossover operators need to be matched to the problem at hand. In order to embed problem knowledge in the crossover operator, this has to be based on a distance that is meaningful for the problem at hand. In previous work [24], we have suggested a rule of thumb: the distance chosen should make the resulting fitness landscape "smooth" in some statistical sense, or in other words, closer solutions should tend to have closer fitness.

Traditional crossovers for binary strings are geometric crossovers under Hamming distance: they all produce offspring on the shortest path between parent strings in the Hamming space. Differently from binary strings where a single, natural definition of distance is normally used, for permutations many notions of edit distance are equally natural (based on swaps, adjacent swaps, insertions, reversals, transpositions and other edit moves). This leads to a number of natural notions of geometric crossover for permutations. Geometric crossovers for permutations are intimately connected with the notion of sorting algorithm: offspring are on the shortest path between parents, hence they are on the minimal sorting trajectory between parents using a specific edit move. Interestingly, this allows to implement geometric crossovers for permutations by using traditional sorting algorithms such as bubble sort, selection sort and insertion sort. Many pre-existing recombination operators for permutations are geometric crossovers. For example, PMX (partially matched crossover) and cycle crossover are geometric under swap distance. Interestingly, cycle crossover is geometric also under Hamming distance restricted to permutations.

Permutations with repetitions are a natural generalization of simple permutations in which each element is allowed to occur more than once. In this paper we start studying the application of geometric crossover to permutations with repetitions. In particular, we propose a new geometric crossover for permutations with repetitions that is a natural generalization of cycle crossover.

Grouping problems are interesting and NP-hard [8]. In this paper, we focus on the multiway graph partitioning problem. When applying evolutionary algorithms to grouping problems, the standard solution encoding is highly redundant. This affects badly the performance of traditional crossover. In [4], a highly effective procedure to compensate for redundancy was introduced that requires a labeling-normalization phase before the actual exchange of genetic material between parents.

Distances are becoming of increasing importance both in the analysis and design of evolutionary algorithm. The Hamming distance is a naïve distance for grouping problems because it does not keep the inherent redundancy of the solution encoding into account. We introduce a natural distance for grouping problems, the *labeling-independent distance*, and we prove that it satisfies the metric axioms and that can be efficiently computed using the Hungarian method [20].

In this paper, we use the labeling-independent distance as basis of geometric crossover to design a new crossover for redundant encodings. Interestingly, this distance gives rise to a crossover that requires a labeling-normalization phase before exchanging genetic material between parents in the traditional way. The new crossover can be implemented exactly and efficiently using the Hungarian method that, unlike previous normalization heuristics, allows obtaining a perfect normalization in an efficient way. We compared the landscapes under Hamming distance and labeling-independent distance and found that the latter landscape presents characteristics that make it better matched with the corresponding geometric crossover; so we expected that the new geometric crossover would achieve better performance. We compared experimentally the traditional crossover (geometric under the Hamming distance), the new geometric crossover based on labeling-independent distance, and a third one employing a heuristic normalization that is known to be very good [17]. The new geometric crossover showed remarkable performance improvement.

A second difficulty with grouping problems is that traditional recombination does not preserve offspring feasibility: recombining parents with the same grouping structures does not lead in general to offspring with the same structure, requiring a repairing mechanism to be applied to the offspring. Per se, the repairing mechanism is not necessarily negative as it can be interpreted as a form of mutation. However, when the extent of change in the offspring is not small, the repairing mechanism degenerates into macro-mutation with deleterious effect on performance.

In general, a much preferred way to deal with this problem is to design a recombination operator that naturally transmits parent feasibility to offspring. In this paper we show that the new cycle crossover for permutation with repetitions allows to search only the space of feasible solutions without the need of any repairing mechanism. We then combine cycle crossover and labeling-independent crossover obtaining a new geometric crossover with both advantages that suits very well grouping problems with redundant encoding. We tested experimentally the new geometric crossovers on graph partitioning obtaining remarkable performance improvement.

The reminder of this paper is organized as follows. In Section 2, we introduce the multiway graph partitioning problem. In Section 3, we introduce the geometric framework and review previous work on permutations. In Section 4, we introduce a new geometric crossover based on labeling-independent distance. In Section 5, we present a correlation analysis of the fitness landscapes associated with the Hamming and labeling-independent distances. In Section 6, we introduce a new geometric crossover, extending the cycle crossover to permutations with repetitions, that preserves the sizes of repetition classes. In Section 7, we recast the graph partitioning problem in terms of permutations with repetitions and motivate the use of cycle crossover. We then combine cycle crossover and labeling-independent crossover into a new geometric crossover with both characteristics. In Section 8, we present experimental results. We draw conclusions in Section 9.

2 Multiway Graph Partitioning

Graph partitioning is an important problem that arises in various fields of computer science, such as sparse matrix factorization, VLSI circuit placement, network partitioning, and so on. Good partitioning of a system not only significantly reduces the complexity involved in the design process, but can also improve the timing performance as well as its reliability [10].

Let $G = (V, E)$ be an unweighted undirected graph, where V is the set of vertices and E is the set of edges. *K-way partition* is a partitioning of the vertex set V into K disjoint subsets $\{C_1, C_2, \ldots, C_K\}$. A K-way partition is said to be *balanced* if the difference of cardinalities between the largest and the smallest subsets is at most one. The *cut size* of a partition is defined to be the number of edges with endpoints in different subsets of the partition. The *K-way partitioning problem* is the problem of finding K-way balanced partition with minimum cut size.

Since the K-way partitioning problem is NP-hard [8], attempts to solve partitioning problems have focused on finding heuristics which yield approximate solutions in polynomial time. Among such methods, the Kernighan-Lin algorithm [16] and the Fiduccia-Mattheyses algorithm (FM) [7] are representative. They are local search heuristics for 2-way partitioning. There have been a number of algorithms for K-way partitioning [5, 6, 15, 30]. There have been also several methods using genetic algorithms [14, 17, 21].

3 Geometric framework

In this section, we report the essential concepts behind a theoretical framework of recent introduction that allows to analyze and design new crossover operators for any solution representation tailored to the problem at hand [23].

3.1 Geometric Preliminaries

The term *distance* or *metric* denotes any real valued function that conforms to the axioms of identity, symmetry and triangular inequality. A simple connected graph is naturally associated to a metric space via its *path metric*: the distance between two vertices in the graph is the length of a shortest path between the vertices. Given a set of editing operations (edit moves) well-defined over a set of syntactic objects, the *edit distance* between two syntactic objects is the minimum number of edit moves needed to transform one into the other. When the edit moves are reversible and every object can be transformed into any other using the edit moves available, the edit distance is a metric.

In a metric space (S, d), a *closed ball* is the set of the form $B(x; \delta) = \{y \in S \mid d(x, y) \leq \delta\}$ where $x \in S$ and δ is a positive real number called the radius of the ball. A *line segment* (or closed interval) is the set of the form $[x, y]_d = \{z \in S \mid d(x, z) + d(z, y) = d(x, y)\}$ where $x, y \in S$ are called extremes of the segment. Metric ball and metric segment generalize the familiar notions of ball and segment in the Euclidean space to any metric space through distance redefinition. These generalized objects look quite different under different metrics. Notice that a metric segment does not coincide to a shortest path connecting its extremes (*geodesic*) as in an Euclidean space. In general, there may be more than one geodesic connecting two extremes; the metric segment is the union of all geodesics.

We assign a structure to the solution *set* by endowing it with a notion of distance d. $M = (S, d)$ is therefore a solution *space* and $L = (M, g)$ is the corresponding fitness landscape, where g is the fitness function over S.

3.2 Definition of Geometric Crossover

The following definition is *representation-independent* whereby crossover is well-defined for any representation. The definition is only *function of the metric d* associated with the search space being based on the notion of metric segment.

Definition 1 (Geometric crossover).
A binary operator GX is a geometric crossover under the metric d if all offspring are in the segment between their parents, i.e., $\forall x, y : GX(x, y) \in [x, y]_d$.

A number of general properties for geometric crossover and mutation have been derived in [23].

Fact 1. *The traditional crossover for K-ary vectors with n crossover points is geometric under Hamming distance [23].*

3.3 Geometric Crossover for Permutations

In previous work we have studied various crossovers for permutations, revealing that PMX [9], a well-known crossover for permutations, is geometric under swap distance. Also, we found that cycle crossover [27], another traditional crossover for permutations, is geometric under swap distance and under Hamming distance (geometricity under Hamming distance for permutations implies geometricity under swap distance but not *vice versa*). Finally, we showed that geometric crossovers for permutations based on edit moves are naturally associated with sorting algorithms: picking offspring on a minimum path between two parents corresponds to picking partially sorted permutations on the minimal sorting trajectory between the parents.

3.4 Geometric Crossover Landscape

The notion of fitness landscape is useful if the search operators employed are connected or matched with the landscape: the greater the connection the more landscape properties mirror search properties. The conventional way to look at the landscape is to see it as a *function of the search operator employed* [12]. Whereas mutation is intuitively associated with the neighbourhood structure of the search space, crossover stretches the notion of landscape further leading to search spaces defined over complicated topological structures.

Geometric crossover and geometric mutation are based on the distance associated with the search space. This approach is the dual of Jones' approach: *we see the genetic operators as functions of the search space*. So, mutation and crossover share the same neighbourhood structure. This greatly simplifies the relationship between crossover and fitness landscape and allows to give a simple rule of thumb that tells what makes a fitness landscape well-searchable by crossover.

Geometric operators are defined as functions of the distance associated to the search space. However, the search space does not come with the problem itself. The problem consists only of a fitness function to optimize, that defines what a solution is and how to evaluate it, but it does not give any structure on the solution set. The act of putting a structure over the solution set is part of the search algorithm design and it is a designer's choice. A fitness landscape is the fitness function plus a structure over the solution space. So, for each problem, there is one fitness function but as many fitness landscapes as the number of possible different structures over the solution set. In principle, the designer could choose the structure to assign to the solution set completely independently from the problem at hand. However, because the search operators are defined over such a structure, doing so would make them decoupled from the problem at hand, hence turning the search into something very close to random search. To avoid this one needs to exploit problem knowledge in the search by choosing a distance that makes sense for the problem at hand.

What is a good distance for the problem at hand? In other words, under which conditions is a landscape well-searchable by geometric operators? As a rule of thumb, geometric mutation and geometric crossover work well on landscapes where the closer pairs of solutions, the more correlated their fitness values. Of course this is no surprise: the importance of landscape smoothness has been advocated in many different context and has been confirmed in uncountable empirical studies with many neighbourhood search meta-heuristics [29]. So, we can answer the previous questions as follows:
Rule-of-thumb 1: if we have a good distance for the problem at hand, then we have good geometric mutation and good geometric crossover.
Rule-of-thumb 2: a good distance for the problem at hand is a distance that makes the landscape "smooth."

4 Labeling-independent Crossover

4.1 Labeling-independent Distance

The standard representation of a solution for K-way graph partitioning is a vector x of size $|V|$ such as $x_i = j \Rightarrow v_i \in P_j$. Since the specific mapping of indices to partitions does not change how the graph is partitioned, each solution has $K!$ representations. For this encoding, the Hamming distance between two solutions is unnatural because it depends on the specific mapping between indices and partitions that is completely arbitrary. We propose a distance measure, the labeling-independent distance, that eliminates this dependency completely.

Formally, the term *distance* or *metric* denotes any real valued function that conforms to the axioms of identity, symmetry and triangular inequality.[1] A distance for which the axiom of identity is relaxed so that distance zero does not necessarily implies equality (but equality still implies distance zero) is called *pseudo-metric*.

Definition 2 (Labeling-independent distance).
Given two K-ary encodings $\mathfrak{a}, \mathfrak{b} \in U = \{1, 2, \ldots, K\}^{|V|}$ (fixed-length vectors on a K-ary alphabet) and a metric $\mathfrak{d}$ in U, we define the labeling-independent distance *LI associated to $\mathfrak{d}$ as follows:*

$$LI(\mathfrak{a}, \mathfrak{b}) := \min_{\sigma \in \Sigma_K} \mathfrak{d}(\mathfrak{a}, \mathfrak{b}_\sigma)$$

where Σ_K is the set of all permutations of length K and $\mathfrak{b}_\sigma$ is a permuted encoding of $\mathfrak{b}$ by a permutation σ, i.e., the i^{th} element b_i of $\mathfrak{b}$ is transformed into $\sigma(b_i)$.

Theorem 1. *Labeling-independent distance LI is a pseudo-metric in U.*

Proof. It is enough to show that LI satisfies the triangle inequality.

$$\begin{aligned} LI(\mathfrak{x}, \mathfrak{y}) + LI(\mathfrak{y}, \mathfrak{z}) &=_{(WLOG)} \mathfrak{d}(\mathfrak{x}, \mathfrak{y}_\sigma) + \mathfrak{d}(\mathfrak{y}, \mathfrak{z}_{\sigma'}) \\ &= \mathfrak{d}(\mathfrak{x}, \mathfrak{y}_\sigma) + \mathfrak{d}(\mathfrak{y}_\sigma, (\mathfrak{z}_{\sigma'})_\sigma) \\ &= \mathfrak{d}(\mathfrak{x}, \mathfrak{y}_\sigma) + \mathfrak{d}(\mathfrak{y}_\sigma, \mathfrak{z}_{\sigma \cdot \sigma'}) \\ &\geq \mathfrak{d}(\mathfrak{x}, \mathfrak{z}_{\sigma \cdot \sigma'}) \quad (\because \mathfrak{d} \text{ is a metric}) \\ &\geq LI(\mathfrak{x}, \mathfrak{z}) \quad (\because \sigma \cdot \sigma' \in \Sigma_K). \end{aligned}$$

□

Given an element $\mathfrak{a} \in U$, since $\mathfrak{d}$ is a metric, there are $K!$ elements such that the labeling-independent distance LI to $\mathfrak{a}$ is zero. If the labeling-independent distance LI between two elements is equal to zero, we define them to be *in relation* $\sim$. Then, the following proposition holds.

Proposition 1. *The relation $\sim$ is an equivalence relation.*

Proof. It is obvious that the relation $\sim$ is reflexive and symmetric. It is transitive as in the following.

$$\begin{aligned} \mathfrak{a} \sim \mathfrak{b}, \mathfrak{b} \sim \mathfrak{c} &\Rightarrow_{(WLOG)} \mathfrak{a} = \mathfrak{b}_\sigma, \mathfrak{b} = \mathfrak{c}_{\sigma'} \\ &\Rightarrow \mathfrak{a} = (\mathfrak{c}_{\sigma'})_\sigma = \mathfrak{c}_{\sigma \cdot \sigma'} \\ &\Rightarrow \mathfrak{a} \sim \mathfrak{c} \quad (\because \sigma \cdot \sigma' \in \Sigma_K). \end{aligned}$$

□

Theorem 2. *Suppose that Q is the quotient set of U by relation $\sim$ (i.e., $Q = U/\sim$). Then, (Q, LI) is a metric space, i.e., the labeling-independent distance LI is a metric in Q.*

Proof. By Proposition 1, Q is well defined. Since LI is a pseudo-metric by Theorem 1, it is clear that LI is a metric in Q. □

So, U is the set of all *labeled partitions* and $\mathfrak{d}$ is a metric on this set. Q is the set of all *unlabeled partitions* associated to U and LI is the corresponding metric on Q associated to $\mathfrak{d}$.

[1] For details, see Section A.1.

4.2 Efficient Normalization and Labeling-independent Distance

We say that $\mathfrak{b}_{\sigma^*}$ is normalized to $\mathfrak{a}$ when $LI(\mathfrak{a}, \mathfrak{b}_{\sigma^*}) = \mathfrak{d}(\mathfrak{a}, \mathfrak{b}_{\sigma^*})$ and we call σ^* a normalizing relabeling.

We show that, in the special case of $\mathfrak{d}$ being Hamming distance H, the problem of computing LI can be formulated as the optimal assignment problem[2] and it can be solved efficiently by the Hungarian method. The problem of finding a normalizing relabeling σ^* is equivalent to computing LI. Let $M = (m_{ij})$ be the $K \times K$ assignment weight matrix between two chromosomes X and Y. Each element m_{ij} means $\sum_{k=1}^{|V|} I(X_k = i, Y_k \neq j)$ or $\sum_{k=1}^{|V|} I(X_k \neq i, Y_k = j)$, where $I(\cdot)$ is the indicator function, i.e., $I(true) = 1$ and $I(false) = 0$. The problem of computing LI is exactly the problem of finding an assignment (permutation) with minimum summation.

Theorem 3. *If the metric $\mathfrak{d}$ is Hamming distance H, then*

$$LI(X, Y) = \min_{\sigma \in \Sigma_K} \sum_{i=1}^{K} \sum_{k=1}^{|V|} I(X_k = i, Y_k \neq \sigma(i)).$$

Proof.

$$\begin{aligned}
LI(X, Y) &= \min_{\sigma \in \Sigma_K} H(X, Y_\sigma) \\
&= \min_{\sigma \in \Sigma_K} \left(\sum_{k=1}^{|V|} I(X_k \neq \sigma(Y_k)) \right) \\
&= \min_{\sigma \in \Sigma_K} \left(\sum_{k=1}^{|V|} \sum_{i=1}^{K} I(X_k = i, \sigma(Y_k) \neq i) \right) \\
&= \min_{\sigma \in \Sigma_K} \left(\sum_{k=1}^{|V|} \sum_{i=1}^{K} I(X_k = i, \sigma^{-1} \cdot \sigma(Y_k) \neq \sigma^{-1}(i)) \right) \\
&= \min_{\sigma \in \Sigma_K} \left(\sum_{k=1}^{|V|} \sum_{i=1}^{K} I(X_k = i, Y_k \neq \sigma^{-1}(i)) \right) \\
&= \min_{\sigma \in \Sigma_K} \left(\sum_{i=1}^{K} \sum_{k=1}^{|V|} I(X_k = i, Y_k \neq \sigma^{-1}(i)) \right) \\
&= \min_{\sigma \in \Sigma_K} \left(\sum_{i=1}^{K} \sum_{k=1}^{|V|} I(X_k = i, Y_k \neq \sigma(i)) \right) \quad (\because \sigma \in \Sigma_K \Leftrightarrow \sigma^{-1} \in \Sigma_K).
\end{aligned}$$

□

4.3 Geometric Crossover under Labeling-independent Metric

We design a new geometric crossover based on the labeling-independent metric. We call it n-point LI-GX.

Definition 3 (n-point LI-GX).
Normalize the second parent to the first under Hamming distance. Do the normal n-point crossover using the first parent and the normalized second parent.

Theorem 4. *The n-point LI-GX is geometric under the labeling-independent metric.*

Proof. Let p_1 and p_2 be parents and c be offspring after n-point LI-GX. It is enough to show that $LI(p_1, p_2) = LI(p_1, c) + LI(c, p_2)$. Since LI is a metric, by triangular inequality, it is trivial that $LI(p_1, p_2) \leq LI(p_1, c) + LI(c, p_2)$. Now, we will show that $LI(p_1, p_2) \geq LI(p_1, c) + LI(c, p_2)$. Let σ' be $argmin_{\sigma \in \Sigma_K} H(p_1, (p_2)_\sigma)$, where $(p_2)_\sigma$ is a permuted encoding of p_2 by σ, i.e., the i^{th} element p_{2i}

[2]For details, see Section A.2.

of p_2 is transformed into $\sigma(p_{2i})$. Then, we let p_2' be $(p_2)_{\sigma'}$.

$$\begin{aligned} LI(p_1, p_2) &= H(p_1, p_2') \\ &= H(p_1, c) + H(c, p_2') \quad (\because \text{Fact 1}) \\ &\geq LI(p_1, c) + LI(c, p_2') \\ &= LI(p_1, c) + LI(c, p_2) \end{aligned}$$

□

To implement n-point LI-GX, we use the Hungarian method to find a maximum matching of the labels of the second parent to the first parent and then we apply a traditional n-point crossover. The time complexity of n-point LI-GX is $O(|V| + K^3)$.

5 Landscape of Labeling-independent Crossover

5.1 Smoothness

5.1.1 Measuring Smoothness

To measure the smoothness of a fitness landscape, we plot its autocorrelation function (or *correlogram*): a graph that on the x-axis has distance between solutions and on the y-axis has the correlation between their fitness. The autocorrelation function for fitness landscapes was introduced by Weinberger [31].

To have a meaningful comparison between correlation functions of fitness landscapes based on different search space structures with different distance distributions, we normalize distances by dividing them by the average distance $E(d)$ between any two points of the search space. Let $r_d(\delta)$ be the normalized correlogram of a fitness landscape based on a metric d. Then $r_d(1)$ is the average distance correlation and is a measure of global smoothness of the landscape. If $r_d(1)$ is positive, we say that the landscape is globally smooth. If it is negative, the landscape is said to be globally discontinuous. If the normalized correlogram of a landscape is always above the normalized correlogram of a second landscape, then the former is smother than the latter.

5.1.2 Correlogram of Local-optimum Space

Since our evolutionary algorithm uses crossover together with hill-climbing, the space searched is the local-optimum space rather than the whole space. So we have to consider the correlogram of such a restricted space. However, it is not easy to plot it exactly. Given a distance value δ, it is not easy to find pairs of local optima such that their distances are equal to δ. If we get it by forming distance classes from a pool of randomly sampled local optima, since the local-optimum space is too big, the range of correlation becomes limited. However, an approximated correlogram can be obtained using algorithm 1.

First, we get M local optima by applying the hill-climber to M randomly generated solutions. In our experiments, we set M to be 3,000. For each perturbation rate p, we change p percent of the bits of each local optimum at random and apply the local optimizer to the perturbed solution to obtain a new local optimum. Then, we compute Hamming distance and labeling-independent distance between the new local optimum and the previous one. After computing the correlation coefficient between the fitness of M pairs, we plot the relation between the distances and the correlation coefficient.

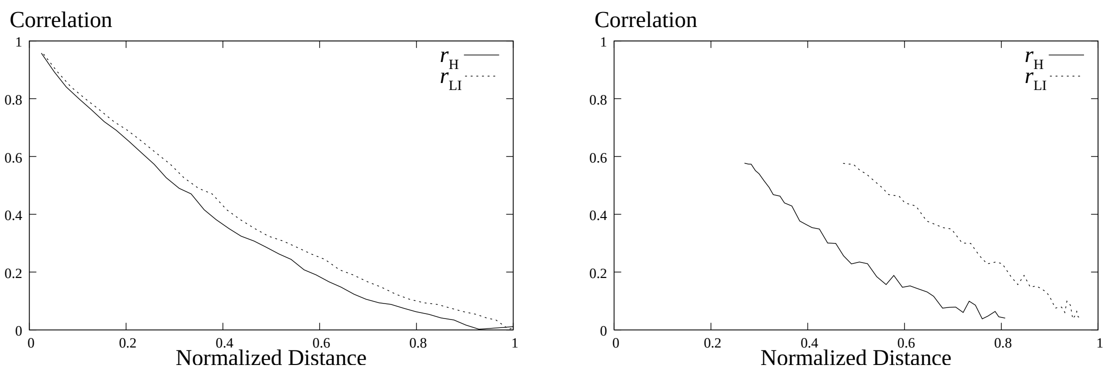
We have some remarks. The average of h_i's and the average of l_i's will increase as the perturbation rate p increases. c's are the same for two plots in Figure 1.

5.1.3 Distance Distributions

Let us consider the search space for 32-way partitions on a graph with 500 vertices (problem instance G500.2.5). We can compute the expected value $E(H)$ in all-partition space approximately. In all-partition space, the probability that the i^{th} positions of two solutions are the same is 1/32. Hence, $E(H)$ is approximately $500 \times (1 - 1/32) = 484.375$. This fact hints that "most pairs of points are nearly of a maximum distance." This is interesting and counterintuitive. The following table shows the average and the maximum distance for each space. It shows that the problem space becomes much narrower in the labeling-independent space (LI-space).

Algorithm 1 Approximated correlogram routine

```
Get randomly generated M local optima s_i's
for each p ∈ (0, 100] do
    Get corresponding M local optima t_i's after "random p% perturbation + local optimization"
    for each pair of M pairs (s_i, t_i)'s do
        Compute Hamming distance (h_i) and labeling-independent distance (l_i)
    end for
    Compute the correlation coefficient (c) between the fitness of M pairs
    Plot a point (average of h_i's/E(H), c)
    Plot a point (average of l_i's/E(LI), c)
end for
```



(a) All-partition space

(b) Local-optimum space

Figure 1: Normalized correlogram (G500.2.5)

Space	$E(d)^\dagger$	$\max(d)$
(all-partition, H)	484.364	500
(local-optimum, H)	484.369	$\leq$ max(whole, H)
(all-partition, LI)	429.010	$\leq 484^\ddagger$
(local-optimum, LI)	274.301	$\leq$ max(whole, LI)

† The values of $E(d)$ were empirically computed from a pool of 500 randomly sampled solutions.

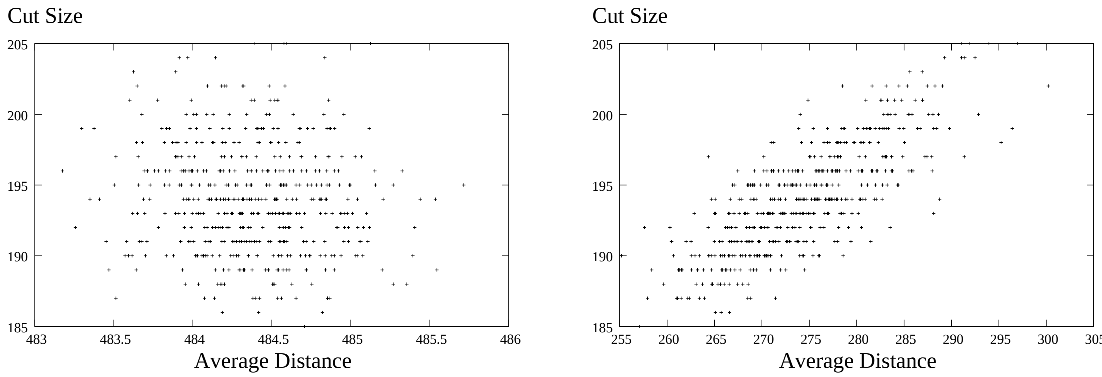
‡ This value is the simple upper bound. Let a and b be 32-way partitions of 500 vertices. The size of the largest subset for a or b is $\lceil 500 \times 1/32 \rceil = 16$. Let i and j be the indices of the largest subsets of a and b, respectively. Then, $LI(a, b) \leq H(a, b_\sigma)$, where σ is a permutation such that $\sigma(j) = i$. Since $H(a, b_\sigma) \leq 500 - 16 = 484$, $LI(a, b) \leq 484$.

5.1.4 Comparison

Figure 1 shows normalized correlograms for all-partition space and local-optimum space. The x-axis means normalized distance between solutions and the y-axis is the correlation coefficient between their fitness. First, we compare H-space and LI-space based on all partitions. The normalized correlograms look similar (see Figure 1a). The LI-correlogram is always above the H-correlogram, but not by much, so we can say that the LI-landscape is slightly smoother than the H-landscape. For both correlograms, points that are at a smaller than average distance have increasingly strong correlation; this makes both landscapes globally smooth and, so, suitable for geometric crossover.

We expect the geometric crossover based on LI-landscape to perform better than the one based on H-landscape. This is only partially due to the stronger correlation. Mostly it is due to the fact that the sizes of the spaces that are actually searched by geometric crossover are different: H-crossover searches the space of all labeled-partitions and LI-crossover searches the much smaller space of all unlabeled-partitions.

Next, we compare H-space and LI-space on the local-optimum space. It is interesting to note that the average distance of local-optimum LI-space is much smaller than the average distance of all-partition LI-space, whereas for the H-space there is almost no difference between the average



(a) Hamming distance

(b) Labeling-independent distance

Figure 2: The relationship between cost and distance (G500.2.5)

distances of the whole space and the local-optimum space. This happens because by local optimization, vertices of some subgraphs tend to belong to the same partition. In local-optimum space, the effect of normalization (associated to LI-space) is very large. The reason must be topological: all the peaks are somehow remapped and clustered by the normalization. Hence, the average distance of local-optimum LI-space is much smaller than that of all-partition LI-space.

Since the average distance between local optima for LI is so much smaller than that for H, the respective normalized correlograms are strongly affected (see Figure 1b): the LI-landscape is really much smoother than the H-landscape (LI-correlogram is much above H-correlogram). So, from the rule-of-thumb that smoother landscape has better geometric crossover performance, we expect that, on the local-optimum space, the geometric crossover based on labeling-independent distance (LI-GX) performs much better than the geometric crossover based on the Hamming distance (H-GX).

5.2 Global Convexity

Given a set of local optima, Boese *et al.* [2] plotted, for each local optimum, the relationship between the cost and the average distance from all the other local optima. They performed experiments for the graph bisection and the traveling salesman problem, and showed that both problems have strong positive correlations. This fact hints that the best local optimum is located near the center of the local-optimum space and, roughly speaking, the local-optimum space is globally convex. In this section, we repeat their experiments for multiway graph partitioning with different distances.

The solution space for the experiment is selected as follows. First, we choose five hundred random solutions and obtain the corresponding set of local optima by locally optimizing them[3]. Figure 2 shows the results for 32-way partitioning of an instance (G500.2.5). The result with the labeling-independent metric was consistent with Boese *et al.*'s results with strong cost-distance correlation (correlation coefficient was 0.79). On the other hand, the result with Hamming distance showed little correlation (correlation coefficient was -0.11).

In summary, there are three reasons we expect LI-GX to perform better than H-GX: (i) global convexity (by cost-distance correlation [2]), (ii) the smaller size of problem space (from average distance), and (iii) smoothness (by autocorrelation [31]). Properties (ii) and (iii) are more obvious in local-optimum space.

6 Cycle Crossover for Permutations with Repetitions

In Section 6.1 we introduce the notions of repetition class of a permutation with repetitions and class-preserving geometric crossover. In Section 6.2 we describe a generalization of cycle crossover for permutations with repetitions. This is a class-preserving geometric crossover under Hamming distance. In Section 6.3 we point out its properties. Cycle crossover for simple permutations is

[3]We used the local optimization algorithm described in Section 8.1.

geometric under swap distance. In Section 6.4 we show that the extension of cycle crossover to permutations with repetitions is not geometric under swap distance. This comes as a surprise because when the extended cycle crossover is applied to simple permutations it behaves exactly as the simple cycle crossover, hence it becomes geometric under swap distance. We elicit the origin of this counterintuitive result.

6.1 Geometric Crossover for Permutations with Repetitions

Permutations with Repetitions: In a permutation every elements occurs exactly once, e.g., (21453). In a permutation with repetitions the same value may occur more than once, e.g., (214154232), where 1 occurs twice, 2 occurs 3 times, 3 occurs once, 4 twice and 5 once. The numbers of repetitions of each element define the repetition class of the permutation. Two permutations with repetitions in which elements have the same number of repetitions belong to the same repetition class. All simple permutations (without repetitions) belong to the same repetition class.

Class-preserving Neighborhood Structure: Let us consider the set P of all the permutations with repetitions such as the element 1 is repeated n_1 times, element 2 is repeated n_2 times, and so on. So n_i is the size of group i. In P, any permutation has a fix number of repetitions for each group. So, all permutations with repetitions in P belong to the same repetition class.

We can define a neighborhood structure on P as follows: the neighbor is generated by swapping the positions of any two different element in the permutation. For example, swapping positions of the elements at positions 2 and 5 in 123231 gives 133221, so they are neighbors.

This operation transforms a permutation in P into another permutation in P: it does not change the sizes of the groups. In other words, the swap operation is a *class-preserving transformation.*

It is also easy to see that this neighborhood is symmetric (if p_1 is in the neighborhood of p_2, then p_2 is in the neighborhood of p_1) and connected (it exists a finite number of swaps that transform any permutation in P into any other permutation in P).

Notice that many other edit operations for simple permutations can be naturally extended to the case of permutations with repetitions and are indeed class-preserving transformations. For example, the block-reversal move that reverses a block of consecutive elements in the permutation does not change the class of permutation either. In this paper we focus on the case of the swap move.

Class-preserving Swap Distance: Passing from the neighborhood structure (symmetric and connected) to the corresponding notion of distance is straightforward: the distance between two solutions is simply the length of a shortest path connecting them in the neighborhood structure. This means that the distance between two permutations with repetitions (of the same repetition class) is the minimum number of swap operations required to transform one into the other. This is an edit distance hence it is a metric.

Class-preserving Geometric Crossover: By definition of geometric crossover, once we have a distance over a space we can define a crossover for that space. The geometric crossover has to pick offspring on a shortest path connecting two solutions. From previous work on geometric crossover for permutations, the extension to permutations with repetitions *seems* straightforward. It is immediate to implement geometric crossover for simple permutations by noticing that picking offspring on a shortest path means picking offspring on a minimal sorting trajectory to sort one permutation to the order of the other by using (in our case) the swap move. This can be easily implemented by using a classical sort algorithm (selection sort in this case), applying a number of sorting moves and interrupting the sorting at a random point (before the sort is complete) and return the permutation partially sorted as offspring. The sort algorithm can also be used to measure the distance between two permutations. Selection sort can be used as a base for geometric crossover because it provably sorts simple permutations within the minimum number of swaps.

However, when applied to permutations with repetitions, selection sort is not guaranteed to sort using the minimum number of swaps. This is very counterintuitive; we explain why this happens in Section 6.4. Therefore, partially sorted permutations are not necessarily on the minimal sorting trajectory (shortest path) between parent permutations and selection sort cannot be used to implement exactly a geometric crossover under swap distance. However, a recombination based on selection sort is class-preserving and it produces a very good approximation of geometric crossover under swap distance, hence it can be thought to be a geometric crossover plus a small mutation. It is possible to formalize the notion of approximated geometric crossover rigorously. The approximation is the largest ratio between the length of the shortest path connecting parents *and* passing through the offspring over the length of the shortest path connecting parents. In a (perfect) geometric crossover,

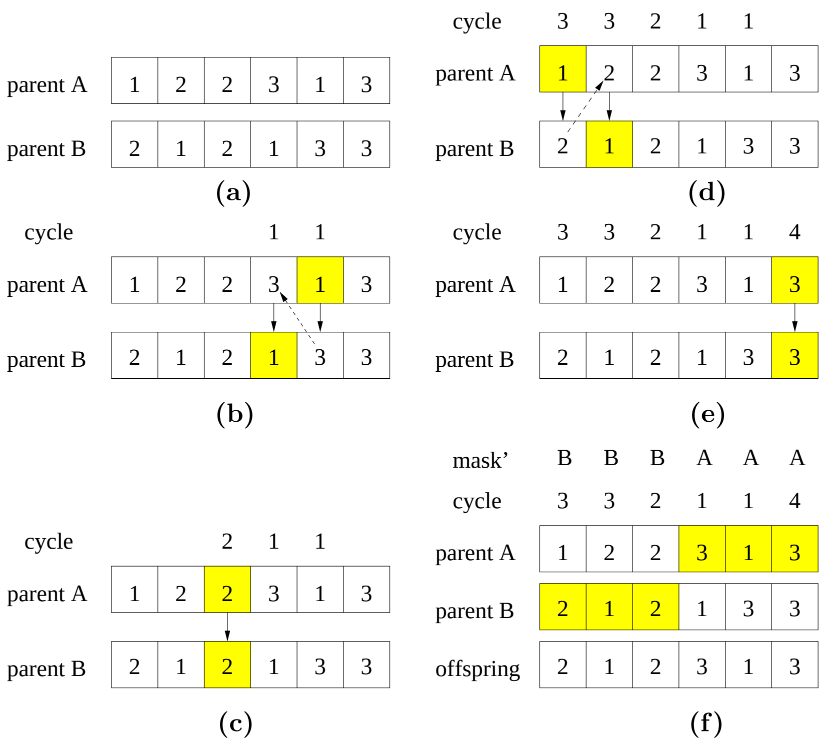


Figure 3: Cycle crossover step by step

this ratio is 1. In an approximated geometric crossover this ratio is greater than 1. The larger the ratio, the worse the approximation.

An alternative way to build geometric crossovers for permutations with repetitions under swap distance is to generalize PMX and cycle crossover that are geometric under swap distance for simple permutations. Despite they may seem unrelated with sorting algorithms, these operators are in fact sorting crossovers: they pick offspring on a minimal sorting trajectory between parents under swap move. The difference is that instead of doing this move by move (like sorting algorithms), they do it in one shot using a syntactic transformation equivalent to the sequential composition of a number of simple swaps. So, the above discussion about the non-geometricity of sorting crossover for permutations with repetitions applies to PMX and cycle crossover too. However cycle crossover is geometric also under Hamming distance. This allows us to generalize it to the case of permutations with repetitions and preserve geometricity under Hamming distance.

6.2 Generalization of Cycle Crossover

The extension of the cycle crossover we propose produces offspring of the same repetition class of the parents. This crossover has two phases: (i) finding cycles and (ii) mixing cycles. We explain these below by means of an example.

PHASE I (FINDING CYCLES):
Let us consider two parents of the same repetition class ($1 \times 2, 2 \times 2$ and 3×2) in Fig 1.a. In order to identify cycles, we proceed as follows.
(1) Pick a random position in parent A, e.g., position 5. In this position in parent A, we have the element 1 corresponding in parent B (at the same position) to the element 3: $1 \rightarrow 3$. Mark position 5 as taken.
(2) Consider the corresponding element in parent B and pick at random any of its occurrence in parent A (among non-taken positions). In our example, pick any 3 in parent A, let us pick the one at position 4. The corresponding element in parent B is 1: $3 \rightarrow 1$. Mark position 4 as taken.
(3) Continue this procedure until you get an element of parent B that is the first element you considered in parent A. When this happens, we have found a cycle. In our example, the last element

we got in parent B is 1 that is the same as the first element we considered in parent A. So we found the cycle: $1 \to 3, 3 \to 1$. This is shown in Figure 1b. Notice that the cycle involves the same number of repetitions in both parents. In our example, 1×1 and 3×1. Excluding the elements of cycle 1 from the two parents leave the remaining elements with the same number of repetitions in the two parents. So the "leftover" permutations are still of the same repetition class. In our example, the leftover repetition class is: 1×1, 2×2 and 3×1.
(4) Repeat loop (1)–(3) to find more cycles until all position have been marked with a cycle tag.

Continuing our example: ($1'$) Pick one free position in parent A at random. Say position 3. We have a 2. ($2'$) Corresponding element in parent B: $2 \to 2$. ($3'$) We already found a cycle: $2 \to 2$ (Figure 1c).

We then start searching for a third cycle: ($1''$) Pick one free position in parent A at random. Say position 1. We have a 1. ($2''$) Corresponding element in parent B: $1 \to 2$. Pick a 2 in parent A: the only one available is the one at position 2. Its corresponding element in parent B is: $2 \to 1$. ($3''$) We have found another cycle: $1 \to 2, 2 \to 1$ (Figure 1d).

We run the process one further time: ($1'''$) The only free position in parent A is position 6. We have 3. ($2'''$) The corresponding element in parent B is 3. ($3'''$) We have found a cycle: $3 \to 3$ (Figure 1e).

All the positions have been assigned to a cycle, so Phase I is over. Notice that the last iteration is always guaranteed to terminate with a cycle (and not with a simple sequence). The last position marked must be the end of the cycle.

PHASE II (MIXING CYCLES):
(1) Create a crossover mask with one entry for each cycle by randomly flipping a coin as many times as the number of cycles detected in the previous phase. In our example, we have 4 cycles and, say, the crossover mask we generate is `mask` = (ABBA). The entries in the mask indicate from which parent each cycle is inherited. In this example, the offspring will inherit the cycles 1 and 4 from parent A and the cycles 2 and 3 from parent B.
(2) We convert this "cycle" mask into a standard recombination mask by relabeling all the entries c_i in `cycle` as follows: $c_i \to \mathtt{mask}(c_i)$ obtaining a new mask `mask'`.
(3) We perform standard mask-based crossover on the two parents using `mask'`, obtaining the offspring as shown in Figure 1f.

Notice that by construction every offspring has the same number of repetitions of the parents. This is because exchanging any cycle between parents is repetition-preserving.

6.3 Properties of Cycle Crossover

The new crossover has the following properties:

1. It is repetition class preserving.
2. It is a proper generalization of cycle crossover: when applied to simple permutations it behaves exactly like cycle crossover.
3. It is geometric under Hamming distance because at any position the element in the offspring equals the element at the same position of one of the parents.
4. This geometric crossover is defined over the induced sub-metric space obtained by restricting the original vector space endowed with Hamming distance to the space of permutations with repetition of the same repetition class. The latter space is much smaller than the former and it is, hence, quicker to search.
5. Applying this crossover to permutations with repetitions of different repetition class (with minor modifications), one obtains offspring with intermediate repetition class with respect to the repetition classes of the parents.

6.4 Non-geometricity under Swap Distance

Theorem 5. *Cycle crossover is not geometric under swap distance.*

Proof. We prove it by giving a counter-example. Let us consider two parents $A = (122313)$ and $B = (213132)$. We can have (at least) two cycles decompositions after Phase I:

cycle	1	1	2	3	3	2
parent A	1	2	2	3	1	3
parent B	2	1	3	1	3	2

and

cycle	1	2	1	1	2	2
parent A	1	2	2	3	1	3
parent B	2	1	3	1	3	2

By combining cycles of the first decomposition, one obtains offspring that are always in the segment between parents under swap distance. However, by combining cycles of the second decomposition, one obtains an offspring that is not in the segment between parents under swap distance:

mask'	A	B	A	A	B	B
cycle	1	2	1	1	2	2
parent A	1	2	2	3	1	3
parent B	2	1	3	1	3	2
offspring	1	1	2	3	3	2

$d_{sw}(A, B) = 3$: This can be seen from the first decomposition in cycles (minimal): 3 cycles of length 2 is equivalent to 3 swaps away.
$d_{sw}(A, \textit{offspring}) = 2$: By construction, the minimal decomposition the offspring and parent A in cycles is the cycle of length 3 taken from parent B. This means they are 2 swaps away.
$d_{sw}(\textit{offspring}, B) = 2$: By construction, the minimal decomposition the offspring and parent B in cycles is the cycle of length 3 taken from parent A. This means they are 2 swaps away.

Since the swap distance between parents is smaller then the sum of the swap distances between parents and offspring, cycle crossover is non-geometric under swap distance. This ends the proof. □

It can be shown in general that the decomposition with the most cycles produces always offspring in the segment between parents under swap distance. All the other decompositions lead to at least one offspring not in the segment between parents. So, *the cycle crossover restricted to the decomposition with the most cycles is geometric crossover under swap distance.*

Cycle crossover for simple permutations produces always offspring within the segment between parents under swap distance. Simple permutations can be thought as permutations with one repetition for each element. Then, how can we connect the result of geometricity under swap distance for simple permutations and non-geometricity under swap distance for permutations with repetitions? We know from abstract algebra that there is a unique decomposition in cycles for simple permutations. A unique decomposition is also the one with the most cycles. Hence in the special case of simple permutations, all the offspring are within the segment between parents.

For simple permutations, cycle crossover is geometric under swap distance and Hamming distance. More generally it can be shown that any geometric crossover for permutations that is geometric under Hamming distance is necessarily geometric under swap distance, but not *vice versa*[4] This is because the segment under Hamming distance is a subset of the segment under swap distance for any choice of extremes. When considering the more general case of permutations with repetitions, cycle crossover is geometric under Hamming distance but it is not geometric under swap distance. As a consequence, the implication geometricity under Hamming distance implies geometricity under swap distance ceases to hold true when stretched to permutations with repetitions.

With similar arguments as for cycle crossover, it can be shown that PMX and all sorting crossovers under swap move are in general non-geometric crossovers.

[4]For example, PMX is geometric under swap distance but is not geometric under Hamming distance.

7 Cycle Crossovers for Graph Partitioning

7.1 Searching Balanced Partitions

In the multiway graph partitioning problem, one needs to keep the sizes of the partitions balanced. So this is a constrained optimization problem. Among all solutions (balanced or not), the feasible ones are only those that are balanced. The way we have dealt with it in previous work [19] is using a crossover that searches the space of *all* solutions and then applying a repairing mechanism that repairs offspring and makes them feasible (balanced). There are other ways to deal with constraints.

An alternative method that does not need to use any repairing mechanism is to have a geometric crossover that searches only the space of balanced solutions. This is the approach we take here. This reduces the size of the search space considerably (the set of balanced solutions is a fraction of the whole search space).

Representation: The starting point for restricting the search to balanced-partitions only is to see the object representing the solution not as a vector of integer but as a permutation with repetitions. Every position in the permutation still represents a vertex of the graph and every integer still represents the label of the group the vertex at that position is assigned to.

A solution is balanced when all the partitions have approximatively the same number of vertices. This means that in the representation, there will be a similar number of repetitions of each element (integer). *Notice that two balanced solutions do not necessarily belong to the same repetition class.*

Equally Balanced Initial Population: In order to restrict the search only to the space of equally balanced partitions, we need to seed the initial population with solutions having *for the same partition exactly the same size for all solutions* (belonging to the same repetition class). Seeding the population with balanced solutions is not sufficient.

Balanced Crossover = Cycle Crossover: Cycle crossover preserves repetition class. Hence given two balanced parents belonging to the same repetition class, it returns offspring of the same repetition class, hence balanced. So there is no need for repairing mutations.

Balanced Mutation = Swap Mutation: We need to use a mutation that keeps a permutation with repetition within the same repetition class. So that, if a solution is balanced, the mutated solution is still balanced. A simple mutation with this characteristic is the simple swap mutation based on the swap move. Notice that the swap move is a good one for the graph partitioning problem because it produces a landscape with a smooth trend (solutions one swap away have very similar fitness). The swap move is also a good base for local search to search the space of balanced grouping only. This move can be used as a base of a more sophisticated mutation that decreases its probability exponentially with the distance from the parent solution.

In section 6.4, the cycle crossover was shown to be non-geometric under swap distance. However, in practice cycle crossover is a good approximation of a geometric crossover under swap distance, in the sense discussed in section 6.1. Hence, in practice our swap mutation and cycle crossover can be considered to be defined over and search the same metric space. This can also be extended to the local search. Having different operators searching the same space is interesting because it is then possible to interpret their interactions in a simple geometric way.

7.2 Crossover Landscape of Cycle Crossover

One important benefit of knowing that a certain search operator is geometric under a specific distance is that it is possible to tell a priori whether the search operator is likely to be a good choice for a specific problem. From previous work, we know that, as a rule-of-thumb, search operators associated to a fitness landscape with a smooth trend, in which closer solutions have stronger fitness correlation, are likely to perform well. Although this is quite intuitive for mutation operators, the geometric framework allows the extension to the case of crossover operators. In the following, we will show that the fitness landscapes associated with the swap move is smooth, making the search operators proposed a good choice for the graph partitioning problem.

The fitness function (to minimize) is the cut size: the number of edges that connect nodes in different groups. The best possible fitness is zero that corresponds to the case in which all nodes are connected only with nodes of the same group. Now, let us consider the worst possible fitness. If d is the maximum degree of a node in the graph (the maximum number of connection with other nodes) the worst case happens when for each node, no matter the group it belongs to, has d connections with nodes not in its group. So, if we have n nodes in the graph, the worst possible fitness is $n \cdot d/2$. The division by two arises from the fact that we count the edges between the nodes, and for symmetry we can swap the two nodes identifying an edge and obtain the same edge. In the worst case, the maximum global fitness variation is therefore $\Delta F = n \cdot d/2$, that is the difference between the worst possible fitness and the best possible fitness.

Let us now consider the maximum possible variation in fitness of two solutions one swap away. The worst case happens when two nodes belonging to different groups that have connections only with nodes of their own groups are swapped. So the worst case of the total change in fitness due to a single swap is $\Delta f_{swap} = 2d$ because, in the transition to a different group, each node increments the cut size of its degree.

A simple measure of smoothness of the landscape is maximum local variation over maximum global variation. The smaller this measure, the smoother the landscape. $S = \Delta f_{swap}/\Delta F = 2d/(n \cdot d/2) = 4/n$. This does not depends on the maximum degree d of the node in the graph but only on the number n of its nodes. For $n = 500$, as typical instance size of the problem, gives a value of landscape smoothness $S = 0.008$. *The fitness landscape under swap distance is very smooth* when compared with the two extreme of smoothness 0 (no change between neighbor solutions) and 1 (maximum and minimum are neighbors).

It is also interesting to notice that the smoothness S of the landscape is inversely proportional to the problem size n. *So, for increasing size of the problem, the fitness landscape becomes smoother.* So this may counteract the increasing difficulty of the search due to a larger space making the performance of the evolutionary algorithm scaling well.

So, the cycle crossover has a number of advantages. It searches only feasible solutions without any need of repairing mechanism. Doing so, it also restrict the search to a much smaller set of candidate solutions than the original one. Finally, it is associated with an extremely smooth landscape that makes it an excellent choice for the graph partitioning problem.

7.3 Combining Labeling-Independent Crossover and Cycle Crossover

Combination of Relabeling and Balanced Solutions: Cycle crossover (Cycle H-GX) searches the space of balanced partitions. LI-GX (see Section 4) searches the space of labeling-independent partitions. We combine these two geometric crossovers obtaining a new geometric crossover with both advantages: it operates fully within the space of labeling-independent balanced partition space, which is a fraction of the original space and could produce highly competitive performance. *The new crossover (Cycle LI-GX) consists of a labeling-normalization phase before applying cycle crossover.* The normalization is done using the Hungarian method as for LI-GX.
Geometricity of Compound Crossover: Cycle LI-GX is still geometric on the phenotypic space restricted to balanced phenotypes. It is in fact the traditional mask-based crossover restricted to the subspace of vectors that take the form of fixed-size class permutations with repetitions.
Equally Balanced Solutions: Relabeling a solution does not affect its balancedness but it may change its repetition class. Cycle crossover without normalization is able to deal with different partition sizes. Cycle crossover plus normalization requires all partitions to have exactly the same size. In this special case, the relabeling does not change the repetition class of the solution and the cycle crossover is therefore applied to solutions of the same repetition class.
Inexact Balance and Cycle Crossover: However we would like to apply the cycle crossover with normalization not only to the restricted class of problems with partitions of exactly the same size but to all balanced partitions. We therefore have to further extend cycle crossover to the recombination of solutions belonging to different repetition classes. When attempting to use the cycle crossover on permutations of different repetition classes, we may not obtain a proper decomposition in cycles. In this case we consider some non-cycles (paths) as cycles and proceed with the recombination. This modification produces offspring with intermediate repetition class with respect to the repetition classes of the parents.

8 Experiments

8.1 Genetic Framework

We used the general structure of hybrid steady-state genetic algorithms. In the following, we describe the framework of genetic algorithm used in our experiments. In this framework, we will change only the crossover operator.

- *Encoding*: We use a K-ary string for each individual to represent a K-way partition. For example, if vertex v_i belongs to partition C_j, the value of the i^{th} gene is j.
- *Initialization*: We randomly create p individuals. Each individual satisfies a balance criterion. We set the population size p to be 50.
- *Selection*: We use the roulette-wheel-based proportional selection scheme. The probability that the best individual is chosen was set to four times higher than the probability that the worst individual is chosen.
- *Mutation*: After traditional 5-point crossover with/without normalization (5pt H-GX, GEFM[17] and 5pt LI-GX), individuals are usually not balanced. We run the following balance adjustment. We start at a random position on the individual and adjust the gene values to the right until the balance is satisfied. This makes some mutation effect, so we do not add any specific mutation.

  After cycle crossover (Cycle H-GX) or normalized cycle crossover (Cycle LI-GX), we run the following swap mutation.

  ```
  function: offspring mutate(offspring) {
      for i = 1 to N {
          if(p_mut > rnd_real_range(0,1))
              offspring = do_rnd_swap(offspring, i);
      }
      return offspring;
  }
  ```

  where $N = |V|$, `rnd_real_range(0,1)` returns random real numbers in the range [0,1] and `do_rnd_swap(offspring, i)` chooses a random point j ($j \neq i$) and swaps positions i and j.

  The mutation parameter `p_mut` is set to be 0.005. Then, the expected Hamming distance between individuals before and after mutation is approximately 1 percent of the problem size $|V|$.
- *Local optimization*: Sanchis [30] extended the FM algorithm for K-way partitioning. The algorithm considers all possible moves of each vertex from its home set to any of the others. He showed that this direct multiway partitioning approach obtained better solutions compared to the recursive approach for random networks. As local optimization engine in our genetic algorithm, we use its variation proposed in [17]. Its time complexity is $O(K|E|)$.
- *Replacement*: If it is superior to the closer parent, the offspring replaces the closer parent, and if not, the other parent is replaced if the offspring is better. Otherwise the worst in the population is replaced.
- *Stopping criterion*: For stopping, we use the number of consecutive fails to replace one of the parents. We set the number to be 50.

8.2 Test Environment

Before showing the experimental results, we first introduce the benchmarks used in this experiment and test environment. We tested on a total of eight graphs which consist of two groups of graphs. They are composed of eight graphs with 500 vertices from [11] (four random graphs G*.* and four random geometric graphs U*.*). The two classes were used in a number of other graph-partitioning studies [1, 3, 18] [22]. More detailed description of them is given in [18].

We conducted tests on 32-way and 128-way partitioning. A C language program was used on a Pentium III 1GHz computer with Linux operating system. It was compiled using *gcc* compiler.

Although the instances of the test-bed are from standard library, most research have focused on 2-way partitioning (bi-partitioning). Moreover, many research about multiway partitioning dealt with circuit partitioning (hyper-graph partitioning). However we know the lower bounds for 32-way partitioning instances from previous literature [14, 17, 19].

8.3 Results

We compare the geometric crossover based on the Hamming distance (5pt H-GX), the geometric crossover based on the corresponding labeling-independent distance (5pt LI-GX), the geometric crossover based on the Hamming distance restricted to permutation with repetitions (Cycle H-GX), the geometric crossover based on the corresponding labeling-independent distance (Cycle LI-GX), and a crossover with previous normalization technique that is known to be very good [17] (GEFM). Notice that these crossovers search different search spaces:

crossover	search space
5pt H-GX	the space of all labeled partitions
5pt LI-GX	the space of all unlabeled partitions
Cycle H-GX	the space of all labeled well-balanced partitions
Cycle LI-GX	the space of all unlabeled well-balanced partitions

Table 1: The Results of 32-way Partitioning

Graph	5pt H-GX			GEFM[17]			5pt LI-GX		
	Best	Ave†	Gen(CPU‡)	Best	Ave†	Gen(CPU‡)	Best	Ave†	Gen(CPU‡)
G500.2.5	182	185.18	1091(173.33)	178	181.83	1503(159.39)	178	181.77	1529(180.30)
G500.05	626	637.25	1424(330.64)	624	631.27	1974(359.25)	624	630.07	2367(334.80)
G500.10	1576	1587.23	1984(713.07)	1575	1582.42	2250(672.64)	1573	1581.40	2422(571.50)
G500.20	4040	4049.44	2247(1502.26)	4038	4045.41	2425(1402.75)	4034	4044.89	2522(1245.96)
U500.05	112	120.65	1327(319.04)	113	117.06	1592(314.26)	112	116.75	1599(331.95)
U500.10	534	542.75	1163(449.76)	529	537.81	1468(464.70)	531	537.04	1494(483.50)
U500.20	1837	1846.30	1123(814.94)	1829	1841.68	1327(715.80)	1832	1841.02	1353(747.04)
U500.40	5363	5389.93	1043(1399.31)	5355	5380.19	1353(1410.09)	5353	5380.30	1374(1398.19)

† Average over 100 runs.

‡ CPU seconds on Pentium III 1GHz.

Table 2: The Results of 32-way Partitioning

Graph	Best Known	Cycle H-GX			Cycle LI-GX		
		Best	Ave†	Gen(CPU‡)	Best	Ave†	Gen(CPU‡)
G500.2.5	178	177	185.59	1269(154.60)	177	**180.50**	1582(204.50)
G500.05	624	627	637.58	1660(282.69)	623	**628.63**	2232(378.46)
G500.10	1574	1575	1587.38	1987(537.10)	1573	**1580.53**	2381(641.58)
G500.20	4037	4039	4049.65	2198(1270.58)	4035	**4043.29**	2538(1354.42)
U500.05	113	113	120.41	1245(300.42)	109	**112.60**	2056(371.45)
U500.10	529	524	539.67	1263(472.37)	523	**528.50**	2086(583.02)
U500.20	1825	1834	1843.80	1170(790.02)	1825	**1831.55**	1646(844.36)
U500.40	5328	5372	5391.77	999(1314.44)	5348	**5365.00**	1691(1515.15)

† Average over 100 runs.

‡ CPU seconds on Pentium III 1GHz.

Table 3: The Results of 128-way Partitioning

Graph	5pt H-GX			GEFM[17]			5pt LI-GX		
	Best	Ave†	Gen(CPU‡)	Best	Ave†	Gen(CPU‡)	Best	Ave†	Gen(CPU‡)
G500.2.5	316	320.14	844(535.13)	318	320.08	853(513.96)	310	314.08	950(759.72)
G500.05	850	853.09	869(688.03)	847	852.38	936(684.05)	839	843.61	1020(947.68)
G500.10	1904	1907.78	932(1300.94)	1901	1906.71	996(1224.89)	1894	1898.03	1168(1316.21)
G500.20	4568	4571.93	965(2333.56)	4565	4570.48	1028(2187.80)	4560	4566.40	1116(2261.33)
U500.05	697	704.06	935(910.83)	698	703.30	971(830.69)	695	702.90	978(1227.62)
U500.10	1679	1684.13	913(1302.62)	1676	1683.92	925(1223.74)	1676	1683.47	921(1618.17)
U500.20	3836	3841.45	890(1437.07)	3836	3841.57	895(1378.42)	3836	3841.44	874(1790.69)
U500.40	8066	8068.54	853(1971.41)	8066	8068.71	850(1847.12)	8065	8068.77	831(2250.17)

† Average over 100 runs.

‡ CPU seconds on Pentium III 1GHz.

Table 1 and Table 2 show the results of 32-way partitioning. On random graphs, Cycle H-GX could not dominate 5pt H-GX on averages, but it performed better on the best. On random geometric

Table 4: The Results of 128-way Partitioning

Graph	Cycle H-GX			Cycle LI-GX		
	Best	Ave†	Gen(CPU‡)	Best	Ave†	Gen(CPU‡)
G500.2.5	311	314.54	911(319.23)	310	**313.05**	964(694.41)
G500.05	839	844.13	977(463.58)	840	**843.19**	1058(876.97)
G500.10	1896	1899.32	1033(766.90)	1893	**1896.71**	1191(1200.03)
G500.20	4564	4567.94	1004(1774.22)	4560	**4565.06**	1164(2149.44)
U500.05	695	701.40	941(684.25)	692	**698.96**	1219(1480.44)
U500.10	1673	1682.74	923(1095.62)	1675	**1681.55**	988(1656.25)
U500.20	3838	**3840.37**	864(1186.80)	3835	3841.15	887(1799.27)
U500.40	8065	**8067.74**	845(1493.73)	8065	8068.42	832(2115.29)

† Average over 100 runs.

‡ CPU seconds on Pentium III 1GHz.

graphs, Cycle H-GX performed better than 5pt H-GX both on averages and the best. Cycle LI-GX and 5pt LI-GX always outperformed Cycle H-GX and 5pt H-GX. Cycle LI-GX showed more improved performance compared with 5pt LI-GX. Except on U500.40, it found lower bounds better than or equal to the best known. LI-GX was better than GEFM on the average and also wins on the best. Similar numbers of generations were reported.

Table 3 and Table 4 show the results of 128-way partitioning. Cycle LI-GX also performed best. Except on G500.05 and U500.10, it found the best solution among them. The performance of Cycle H-GX was better than in the case of 32-way partitioning. On random geometric graphs, it outperformed 5pt LI-GX. But on random graphs, 5pt LI-GX performed better. 5pt H-GX was always dominated by others. LI-GX was better than 5pt H-GX and GEFM both on the average and on the best.

In summary, we got clear improvements for all the tested instances. In particular, for 32-way partitioning on random geometric graphs, there was a large improvement.

Let us now turn to the computing time. For small number K, normalization by the Hungarian method affects computational time little. In 32-way partitioning, Cycle LI-GX was about 1.2 times slower than Cycle H-GX. But, normalization time increases as K increases. In fact, Cycle LI-GX was about 1.7 times slower than Cycle H-GX in 128-way partitioning. Cycle crossovers were faster than 5-point crossovers. In results, Cycle H-GX and Cycle LI-GX were faster than 5pt H-GX and 5pt LI-GX, respectively. Consequently, Cycle H-GX was fastest among them.

8.4 Geometricity

Let LI be the labeling-independent metric and let p_1, p_2 be parents, c be offspring after crossover, and o be offspring after mutation (balance adjustment). Each value in Table 5 and Table 6 means the average value of $(LI(p_1,c) + LI(c,p_2) - LI(p_1,p_2))/LI(p_1,p_2)$ and $(LI(p_1,o) + LI(o,p_2) - LI(p_1,p_2))/LI(p_1,p_2)$. To measure the degree of distortion from the shortest path, we make this experiments. By Theorem 4, it is trivial that the values of "After xover" in 5pt LI-GX are zero. The values of "After mutation" in 5pt LI-GX were also the smallest.

From a geometrical viewpoint, it is not surprising that 5pt LI-GX and GEFM performed similarly, even if 5pt LI-GX was better, because after all GEFM is almost geometric under the distance 5pt LI-GX is based upon. Since, from the landscape analysis we know that this is a very meaningful distance for the problem at hand, this just corroborates the rule-of-thumb that given a good distance, one get a good geometric crossover, other details being of secondary importance (exact probability distribution of the search operator). The argument that geometricity is what really counts is reinforced by noting (see Table 6) that GEFM was a less good approximation of geometric crossover for 128-way partitioning (0.03 non-geometric) and this was mirrored in Table 4 with a bigger difference in absolute values between the averages of GEFM and 5pt LI-GX. So, when GEFM was less geometric, this was immediately reflected in a loss of performance.

9 Conclusions

Geometric crossover is a representation-independent generalization of the traditional crossover defined using the distance of the solution space. By choosing a distance firmly rooted in the syntax of the

Table 5: The Degree of Distortion on 32-way Partitioning

Graph	5pt H-GX		GEFM[17]		5pt LI-GX		Cycle LI-GX	
	After xover[1]	After mutation[2,†]	After xover[1]	After mutation[2,†]	After xover[1]	After mutation[2,†]	After xover[2]	After mutation[2,‡]
G500.2.5	0.56	0.69	0.01	0.14	0.00	0.12	0.01	0.03
G500.05	0.42	0.53	0.01	0.14	0.00	0.13	0.01	0.04
G500.10	0.42	0.53	0.01	0.13	0.00	0.11	0.01	0.04
G500.20	0.39	0.49	0.01	0.12	0.00	0.12	0.01	0.03
U500.05	1.07	1.25	0.01	0.15	0.00	0.14	0.01	0.06
U500.10	1.49	1.72	0.01	0.15	0.00	0.15	0.01	0.12
U500.20	1.90	2.18	0.01	0.15	0.00	0.14	0.01	0.09
U500.40	0.96	1.12	0.01	0.15	0.00	0.14	0.01	0.06

1 Not balanced (not feasible) solution. 2 Balanced (feasible) solution.

† Balance adjustment. ‡ Swap mutation (1%).

Average over 100 runs.

Table 6: The Degree of Distortion on 128-way Partitioning

Graph	5pt H-GX		GEFM[17]		5pt LI-GX		Cycle LI-GX	
	After xover[1]	After mutation[2,†]	After xover[1]	After mutation[2,†]	After xover[1]	After mutation[2,†]	After xover[2]	After mutation[2,‡]
G500.2.5	0.75	0.99	0.03	0.25	0.00	0.21	0.02	0.05
G500.05	0.69	0.91	0.03	0.24	0.00	0.22	0.02	0.05
G500.10	0.62	0.83	0.03	0.24	0.00	0.21	0.02	0.05
G500.20	0.55	0.75	0.03	0.23	0.00	0.21	0.02	0.04
U500.05	1.14	1.45	0.03	0.25	0.00	0.22	0.02	0.07
U500.10	0.83	1.07	0.03	0.25	0.00	0.22	0.02	0.05
U500.20	0.55	0.74	0.03	0.23	0.00	0.21	0.02	0.04
U500.40	0.42	0.60	0.03	0.22	0.00	0.19	0.01	0.03

1 Not balanced (not feasible) solution. 2 Balanced (feasible) solution.

† Balance adjustment. ‡ Swap mutation (1%).

Average over 100 runs.

solution representation and tailored to the problem at hand as basis for geometric crossover, one can design good new crossovers for any representation and any problem.

The multiway graph partitioning problem presents two challenges for crossover: feasibility and redundancy. In this paper we have addressed these issues and made the following contributions:

- We have designed a new crossover for permutations with repetitions that naturally suits partition problems and addresses the feasibility problem.
- Also, we have shown that the important notion of normalization before recombination, that is very effective on problems with redundant encodings, can be naturally cast in geometric terms using a distance that filters the redundancy of the encoding together with the formal definition of geometric crossover.
- This geometric point of view on normalization has allowed us to study its effect on the fitness landscape and explain, within the geometric framework, why normalization before recombination for redundant encodings is a good idea.
- The landscape analysis also provided evidence for the fact that the labeling-independent distance is more suitable for the solution space of multiway graph partitioning problem than the Hamming distance.
- We have designed another geometric crossover based on the labeling-independent distance that addresses the redundancy problem.
- We have then combined these two crossovers obtaining a new, much superior geometric crossover that suits partition problems with redundant encodings. In extensive experimentation, we had demonstrated that this crossover outperforms previously known methods, either providing new lower bounds or equalling known best lower bounds in a variety of graph partitioning benchmark problems.

This paper shows that the theory of geometric crossover is not just theory for its own sake. Indeed it can be put at work in practice and produce excellent results for hard combinatorial optimization problems such as the multi-way graph partitioning problem. Redundancy and feasibility are important issues arising in practice in many problems. Geometric crossover can address them in a natural way within its general framework.

10 Acknowledgments

The authors would like to thank Riccardo Poli for his helpful comments and suggestions that improved the quality of this paper.

A Appendix

A.1 Metric

A metric space is a set of points with an associate *metric* on the set [13]. Given a space X, the properties of its metric $\mathfrak{d} : X \times X \to \mathbb{R}$ are in the following.

1. $0 \leq \mathfrak{d}(x, y) < \infty$ for all x and y in X.
2. $\mathfrak{d}(x, y) = 0$ if and only if $x = y$.
3. $\mathfrak{d}(x, y) = \mathfrak{d}(y, x)$ for all x and y in X.
4. (Triangle Inequality)
   $\mathfrak{d}(x, z) \leq \mathfrak{d}(x, y) + \mathfrak{d}(y, z)$ for all x, y, and z in X.

If property 2 is violated (i.e., for some x and y such that $x \neq y$, $\mathfrak{d}(x, y) = 0$), then $\mathfrak{d}$ is called a *pseudo-metric*. If property 4 does not hold, then $\mathfrak{d}$ is called a *semi-metric*.

A.2 Optimal Assignment Problem

Consider a weighted complete bipartite graph with bipartition (X, Y), where $X = \{x_1, x_2, \ldots, x_K\}$, $Y = \{y_1, y_2, \ldots, y_K\}$, and each edge $(x_i, y_j) \in X \times Y$ has its weight w_{ij}. The *optimal assignment problem* is the problem of finding a maximum-weight (or minimum-weight) perfect matching in this weighted graph as follows:

$$\max_{\sigma \in \Sigma_K} \left. \sum_{i=1}^{K} w_{i\sigma(i)} \right) \quad \text{or} \quad \min_{\sigma \in \Sigma_K} \left. \sum_{i=1}^{K} w_{i\sigma(i)} \right)$$

where σ is a permutation. It is also known as the bipartite weighted matching problem.

To solve the optimal assignment problem, it is possible to enumerate all $K!$ permutations in Σ_K and find an optimal one among them. However, for a large K, such a procedure is intractable. Fortunately, Kuhn [20] proposed an efficient way to solve the problem. It is called the *Hungarian method*. Roughly speaking, starting from the initial unweighted bipartite graph with no edge, the method iteratively modifies edge weights, adds new edges into the bipartite graph, and applies the *maximum matching* (or *minimum covering*) in the resulting bipartite graph. It continues the above process until a perfect matching is found. The Hungarian method gives an optimum assignment and it can be implemented in $O(K^3)$ time [28].

References

[1] R. Battiti and A. Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Transactions on Computers*, 48(4):361–385, 1999.

[2] K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 15:101–113, 1994.

[3] T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 45(7):841–855, 1996.

[4] S. S. Choi and B. R. Moon. Normalization in genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 862–873, 2003.

[5] J. Cong and S. K. Lim. Multiway partitioning with pairwise movement. In *Proceedings of the International Conference on Computer-Aided Design*, pages 512–516, 1998.

[6] J. Cong, S. K. Lim, and C. Wu. Performance driven multi-level and multiway partitioning with retiming. In *Proceedings of the ACM/IEEE-CAS/EDAC Design Automation Conference*, pages 274–279, 2000.

[7] C. Fiduccia and R. Mattheyses. A linear time heuristics for improving network partitions. In *Proceedings of the 19th ACM/IEEE Design Automation Conference*, pages 175–181, 1982.

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

[9] D. E. Goldberg and R. Lingle. Alleles, loci, and the travelling salesman problem. In *Proceedings of the First International Reference on Genetic Algorithms and their Applications*, pages 154–159, 1985.

[10] T. C. Hu and E. S. Kuh. *VLSI Curcuit Layout Theory and Design*. IEEE Press, New York, 1985.

[11] D. S. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, Part 1, graph partitioning. *Operations Research*, 37:865–892, 1989.

[12] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, 1995.

[13] D. W. Kahn. *Topology: An introduction to the point-set and algebraic areas*. Dover Publications, Inc., New York, 1995.

[14] S. J. Kang and B. R. Moon. A hybrid genetic algorithm for multiway graph partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 159–166, 2000.

[15] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[16] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, Feb. 1970.

[17] J. P. Kim and B. R. Moon. A hybrid genetic search for multi-way graph partitioning based on direct partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 408–415, 2001.

[18] Y. H. Kim and B. R. Moon. Lock-gain based graph partitioning. *Journal of Heuristics*, 10(1):37–57, January 2004.

[19] Y. H. Kim, Y. Yoon, A. Moraglio, and B. R. Moon. Geometric crossover for multiway graph partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1217–1224, 2006.

[20] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.*, 2:83–97, 1955.

[21] G. Laszewski. Intelligent structural operators for the k-way graph partitioning problem. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 45–52, July 1991.

[22] P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. *Evolutionary Computation*, 8(1):61–91, 2000.

[23] A. Moraglio and R. Poli. Topological interpretation of crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1377–1388, 2004.

[24] A. Moraglio and R. Poli. Geometric crossover for the permutation representation. *Technical Report CSM-429, , University of Essex*, 2005.

[25] A. Moraglio and R. Poli. Topological crossover for the permutation representation. In *GECCO 2005 Workshop on Theory of Representations*, 2005.

[26] A. Moraglio, R. Poli, and R. Seehuus. Geometric crossover for biological sequences. In *Proceedings of European Conference on Genetic Programming*, pages 121–132, 2006.

[27] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the travelling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230, 1987.

[28] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, N.J., 1982.

[29] P. M. Pardalos and M. G. C. Resende, editors. *Handbook of Applied Optimization*. Oxford University Press, 2002.

[30] L. A. Sanchis. Multiple-way network partitioning. *IEEE Transactions on Computers*, 38(1):62–81, 1989.

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