Cycle Crossover for Permutations with Repetitions Application to Graph Partitioning

Technical Report CSM-454

Alberto Moraglio[1], Yong-Hyuk Kim[2], Yourim Yoon[2],
Byung-Ro Moon[2], and Riccardo Poli[1]

[1] Comp. Sci. Dept., University of Essex, UK
{amoragn, rpoli}@essex.ac.uk
[2] Comp. Sci. Sch. & Eng., Seoul National University, Korea
{yhdfly,yryoon,moon}@soar.snu.ac.kr

Abstract. Geometric crossover is a representation-independent generalisation 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. In previous work, we have applied geometric crossover to simple permutations. In this paper, we design a new geometric crossover for permutations with repetitions that naturally suits partition problems and test it on the graph partitioning problem. Our new crossover outperforms all previous ones.
Keywords: Geometric crossover, cycle crossover, permutations with repetitions, graph partitioning.

1 Introduction

Geometric crossover [8] 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 can be used to guide the design of new specific crossover operators for non-standard representations using as base for geometric crossover distances rooted on the specific representation (e.g., edit distances) [10]. Geometric crossover generalises many pre-existing search operators for the major representations used in evolutionary algorithms, such as binary strings and real vectors [8], permutations [10], syntactic trees [9] and, sequences [11].

Traditional crossover for binary strings is geometric crossover under Hamming distance. 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 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 are geometric crossovers. For example, PMX 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 generalisation of simple permutations. 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 generalisation of Cycle crossover.

Grouping problems are interesting and NP-hard. When applying evolutionary algorithms to grouping problems, the standard solution encoding is highly redundant. This affects badly the performance of traditional crossover. In previous work [6], we developed a geometric crossover for the graph partitioning problem based on a labeling-independent distance that filters the inherent redundancy of the solution encoding that performed very well.

A second difficulty with grouping problems is that traditional recombination does not preserve feasibility of offspring: 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 naturally applies to grouping problems allowing 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 and report remarkable performance improvement.

The remainder of this paper is organised as follows. In Section 2, we introduce the geometric framework and review previous work on permutations. In Section 3, we introduce the multiway graph partitioning problem and a previous geometric crossover based on the labeling-independent distance. In Section 4, we introduce a new geometric crossover, extending the cycle crossover to permutations with repetitions, that preserves the sizes of repetition classes. In Section 5, 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 6, we present experimental setting and results, and we draw conclusions in Section 7.

2 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 [8].

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 with 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 generalise the familiar notions of ball and segment in the Euclidean space to any metric space through distance redefinition. These generalised objects look quite different under different metrics. Notice that a metric segment does not coincide with 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.

The following definition is *representation-independent* whereby crossover is well-defined for any representation. It 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 CX is a geometric crossover under the metric d if all offspring are in the segment between their parents, i.e., $\forall x, y : CX(x, y) \in [x, y]_d$.*

A number of general properties for geometric crossover and mutation have been derived in [8]. The traditional crossover for K-ary vectors with n crossover points is geometric under Hamming distance [8].

In previous work, we have studied various crossovers for permutations, revealing that PMX, a well-known crossover for permutations, is geometric under swap distance. Also, we found that Cycle crossover , another traditional crossover for permutations, is geometric under swap distance and under Hamming distance . Finally, we showed that geometric crossovers for permutations based on edit moves are naturally associated with sorting algorithms.

3 Graph Partitioning and Labeling-Independent Crossover

3.1 Multiway Graph Partitioning

Graph partitioning is an important problem that arises in various fields of computer science, such as sparse matrix factorisation, 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.

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, attempts to solve partitioning problems have focused on finding heuristics which yield approximate solutions in polynomial time.

3.2 Geometric Crossover for Unlabeled Partitions

The standard representation of a solution for K-way graph partitioning is a vector r of size $|V|$ such as $r_i = j \Rightarrow v_i \in C_j$. Since the specific mapping of indices to partitions does not change how the graph is partitioned, each solution has $K!$ representations. Traditional crossover, that is geometric under Hamming distance, does not perform well on redundant encodings. In fact, for this encoding, Hamming distance between two solutions is unnatural because it depends on the specific mapping between indices and partitions that is completely arbitrary. We proposed a distance measure, the labeling-independent distance, that eliminates this dependency completely [5]. In [6], we proposed a new geometric crossover (LI-GX) for graph partitioning based on a labeling-independent distance associated with Hamming distance that filters the redundancy of the encoding. We showed that LI-GX can be implemented efficiently by using the Hungarian method [7] to normalise the labeling on the second parent to that of the first parent and then applying traditional crossover. LI-GX can be thought as restricting the search to the space of unlabeled-partitions only. It outperforms by far the traditional crossover (H-GX) that searches the whole space of labeled-partitions.

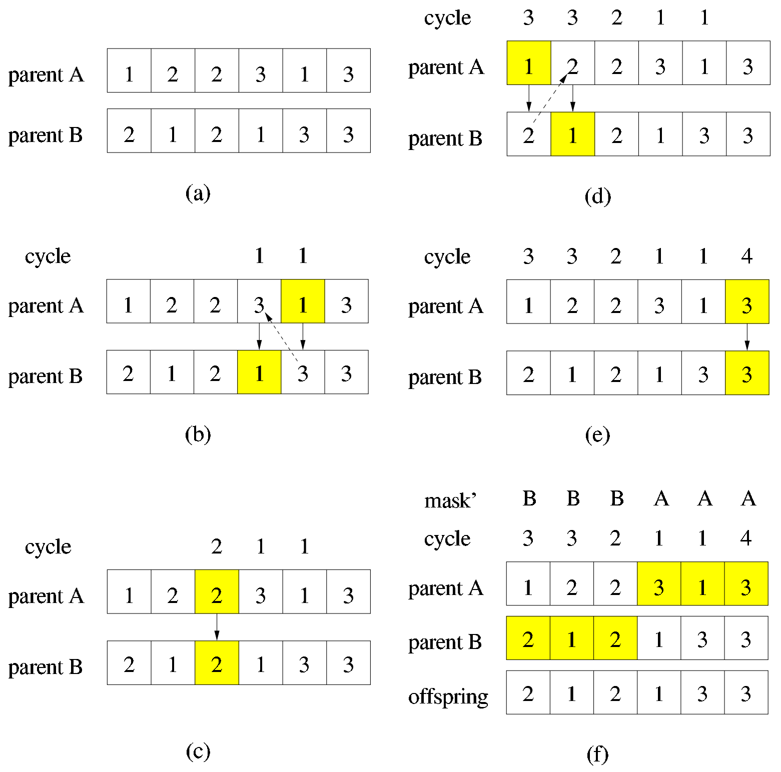

Fig. 1. Cycle Crossover step by step

4 Cycle Crossover for Permutations with Repetitions

Permutations with repetitions In a permutation, every element 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. Two permutations with repetitions in which elements have the same number of repetitions are said to belong to the same repetition class. All simple permutations (without repetitions) belong to the same repetition class.

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: $1 \rightarrow 3, 3 \rightarrow 1$. This is shown in fig. 1.b. 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: $2 \rightarrow 2$. ($3'$) We already found a cycle: $2 \rightarrow 2$ (fig. 1.c).

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: $1 \rightarrow 2$. Pick a 2 in parent A: the only one available is the one at position 2. Its corresponding element in parent B is: $2 \rightarrow 1$. ($3''$) We have found another cycle: $1 \rightarrow 2, 2 \rightarrow 1$ (fig. 1.d).

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: $3 \rightarrow 3$ (fig. 1.e).

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 (MIX 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: $c_i \rightarrow \mathtt{mask}(c_i)$ obtaining a new mask `mask'`.
(3) We perform standard mask-based crossover on the two parents using `mask'`, obtaining the offspring as shown in fig. 1.f.

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.

The new crossover has the following properties:

1. It preserves repetition class.
2. It is a proper generalisation 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.

5 Cycle Crossovers for Graph Partitioning

5.1 Searching Balanced Partitions

In the multiway graph partitioning problem, one needs to keep the sizes of the partitions balanced. So this is a constrained optimisation problem, where the constraint is the balancedness. 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 [6] is searching using a crossover that searches the space of all solutions and then applying a repairing mechanism, that can be thought as a mutation, that repairs offspring and makes them feasible (balanced). There are other ways to deal with constraints and one nice property of geometric crossover is that it can be used to deal with constraints very naturally.

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).
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 swap mutation: you pick any two (different) elements in the permutation and swap their positions.

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).

This 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.

An important property of this mutation is that, if one measures the distance between solutions in the search space as the minimum number of swaps to transform one into the other, one gets a metric space. The cycle crossover suggested is (almost) geometric in this metric space as well. Hence, our swap mutation and cycle crossover are 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.

5.2 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 3.2) 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-normalisation phase before applying cycle crossover.*

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 being fixed-size class permutations with repetitions.

Equally balanced solutions: Relabeling a solution does not affect its balancedness but in general it is not a repetition-class-preserving transformation. If one applies cycle crossover using the Hungarian method to do the relabeling, one has to make sure that all the groups have the same size. If the groups have different sizes, a modified version of cycle crossover which can recombine permutations with repetitions of different repetition classes should be used.

Inexact balance and cycle crossover: Cycle crossover without normalisation is able to deal with different partition sizes. Cycle crossover plus normalisation requires all partitions to have *exactly* the same size because normalisation changes the given sizes of partitions. So, when we apply cycle crossover to balanced solutions with partitions of almost the same size, we do not always find a cycle. We solved this problem heuristically and considered some non-cycles (paths) as cycles.

6 Experiments

6.1 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. Under this framework, we will change only the crossover operator.

– *Encoding*: We use a K-ary string for each chromosome 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.
– *Initialisation*: We randomly create p chromosomes. Each chromosome 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 chromosome is chosen was set to four times higher than the probability that the worst chromosome is chosen.
– *Mutation*: After cycle crossover or normalised cycle crossover, we run the swap mutation. The mutation parameter `p_mut` is set to be 0.005. Then, the expected Hamming distance between chromosomes before and after mutation is approximately 1 percent of the problem size $|V|$.
– *Local optimisation*: Sanchis [12] 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 optimisation engine in our genetic algorithm, we use its variation proposed in [3]. 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.

6.2 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 [2] (four random graphs G*.* and four random geometric graphs U*.*). The two classes were used in a number of other graph-partitioning studies [1]. More detailed description of them is given in [4].

We conducted tests on 32-way and 128-way partitioning. 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 [6].

6.3 Results

We compare the geometric crossover based on Hamming distance (5pt H-GX), the geometric crossover based on the corresponding labeling-independent distance (5pt LI-GX), the geometric crossover based on Hamming distance restricted to permutation with repetitions (Cycle H-GX), and the geometric crossover based on the corresponding labeling-independent distance (Cycle LI-GX). 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 shows 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 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.

Table 2 shows 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.

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

For small number K, normalisation 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, normalisation 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.

7 Conclusions

In this paper, we have designed a new geometric crossover for permutations with repetitions that generalises cycle crossover and that naturally suits partition problems. We have then combined this crossover with another geometric crossover, developed in previous work, that deals well with redundant encoding obtaining a new, much superior geometric crossover that suits partition problems with redundant encodings. In extensive experimentation, we have 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.

References

1. Battiti, R., and Bertossi, A. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Transactions on Computers 48*, 4 (1999), 361–385.
2. Johnson, D. S., Aragon, C., McGeoch, L., and Schevon, C. Optimization by simulated annealing: An experimental evaluation, Part 1, graph partitioning. *Operations Research 37* (1989), 865–892.
3. Kim, J. P., and Moon, B. R. A hybrid genetic search for multi-way graph partitioning based on direct partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2001), pp. 408–415.

Table 1. The Results of 32-way Partitioning

Graph	Best	5pt H-GX			5pt LI-GX			Cycle H-GX			Cycle LI-GX		
	Known	Best	Ave†	Gen	Best	Ave†	Gen	Best	Ave†	Gen	Best	Ave†	Gen
G500.2.5	178	182	185.18	1091	178	181.77	1529	177	185.59	1269	177	180.50	1582
G500.05	624	626	637.25	1424	624	630.07	2367	627	637.58	1660	623	628.63	2232
G500.10	1574	1576	1587.23	1984	1573	1581.40	2422	1575	1587.38	1987	1573	1580.53	2381
G500.20	4037	4040	4049.44	2247	4034	4044.89	2522	4039	4049.65	2198	4035	4043.29	2538
U500.05	113	112	120.65	1327	112	116.75	1599	113	120.41	1245	109	112.60	2056
U500.10	529	534	542.75	1163	531	537.04	1494	524	539.67	1263	523	528.50	2086
U500.20	1825	1837	1846.30	1123	1832	1841.02	1353	1834	1843.80	1170	1825	1831.55	1646
U500.40	5328	5363	5389.93	1043	5353	5380.30	1374	5372	5391.77	999	5348	5365.00	1691

† Average over 100 runs.

Table 2. The Results of 128-way Partitioning

Graph	5pt H-GX			5pt LI-GX			Cycle H-GX			Cycle LI-GX		
	Best	Ave†	Gen	Best	Ave†	Gen	Best	Ave†	Gen	Best	Ave†	Gen
G500.2.5	316	320.14	844	310	314.08	950	311	314.54	911	310	313.05	964
G500.05	850	853.09	869	839	843.61	1020	839	844.13	977	840	843.19	1058
G500.10	1904	1907.78	932	1894	1898.03	1168	1896	1899.32	1033	1893	1896.71	1191
G500.20	4568	4571.93	965	4560	4566.40	1116	4564	4567.94	1004	4560	4565.06	1164
U500.05	697	704.06	935	695	702.90	978	695	701.40	941	692	698.96	1219
U500.10	1679	1684.13	913	1676	1683.47	921	1673	1682.74	923	1675	1681.55	988
U500.20	3836	3841.45	890	3836	3841.44	874	3838	3840.37	864	3835	3841.15	887
U500.40	8066	8068.54	853	8065	8068.77	831	8065	8067.74	845	8065	8068.42	832

† Average over 100 runs.

4. Kim, Y. H., and Moon, B. R. Lock-gain based graph partitioning. *Journal of Heuristics 10*, 1 (January 2004), 37–57.
5. Kim, Y. H., and Moon, B. R. New topologies for genetic search space. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2005), pp. 1393–1399.
6. Kim, Y. H., Yoon, Y., Moraglio, A., Moon, B. R., and Poli, R. Geometric crossover for multiway graph partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference (to appear)* (2006).
7. Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart. 2* (1955), 83–97.
8. Moraglio, A., and Poli, R. Topological interpretation of crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2004), pp. 1377–1388.
9. Moraglio, A., and Poli, R. Geometric crossover for the permutation representation. *Technical Report CSM-429* (2005).
10. Moraglio, A., and Poli, R. Topological crossover for the permutation representation. In *GECCO 2005 Workshop on Theory of Representations* (2005).
11. Moraglio, A., Poli, R., and Seehuus, R. Geometric crossover for biological sequences. In *Proceedings of EuroGP 2006 (to appear)* (2006).
12. Sanchis, L. A. Multiple-way network partitioning. *IEEE Transactions on Computers 38*, 1 (1989), 62–81.