Generalized Cycle Crossover for Graph Partitioning

Alberto Moraglio
Dept. of Comp. Sci.
University of Essex
Wivenhoe Park, Colchester
CO4 3SQ, UK
amoragn@essex.ac.uk

Yong-Hyuk Kim
Dept. of Math. Sci. and
Res. Inst. of Math.
Seoul National University
Seoul, 151-744 Korea
yhdfly@soar.snu.ac.kr

Yourim Yoon
Sch. of Comp. Sci. & Eng.
Seoul National University
Sillim-dong, Gwanak-gu
Seoul, 151-744 Korea
yryoon@soar.snu.ac.kr

Byung-Ro Moon
Sch. of Comp. Sci. & Eng.
Seoul National University
Sillim-dong, Gwanak-gu
Seoul, 151-744 Korea
moon@soar.snu.ac.kr

Riccardo Poli
Dept. of Comp. Sci.
University of Essex
Wivenhoe Park, Colchester
CO4 3SQ, UK
rpoli@essex.ac.uk

ABSTRACT

We propose a new crossover that generalizes cycle crossover to permutations with repetitions and naturally suits partition problems. We tested it on graph partitioning problems obtaining excellent results.

Categories and Subject Descriptors: G.2.3 [Mathematics of Computing]: DISCRETE MATHEMATICS – *Applications*

General Terms: Algorithms.

Keywords: Cycle crossover, permutations with repetitions, graph partitioning.

1. INTRODUCTION

Permutations with repetitions are a natural generalization of simple permutations. In this paper we propose a new crossover for permutations with repetitions that is a natural generalization of cycle crossover.

Grouping problems are interesting and NP-hard. A difficulty with groping problems is that traditional recombination does not preserve the feasibility of offspring. So, recombining parents with the same grouping structures does not lead to offspring with the same structure, requiring a repairing mechanism to be applied to the offspring. A better way to deal with this problem is to design a recombination operator that transmits parent feasibility to offspring. Our cycle crossover for permutation with repetitions naturally applies to grouping problems allowing to search only the space of feasible solutions. We tested the new crossover on graph partitioning obtaining remarkable performance.

2. GENERALIZED CYCLE CROSSOVER

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). Two permutations with repetitions in which elements have the same number of repetitions are said to belong to the same repetition class.

The extension of 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.

PHASE I (FINDING CYCLES):
In order to identify cycles, we proceed as follows:
(1) Pick a random position in parent A.
(2) Consider the corresponding element in parent B and pick at random any of its occurrences in parent A (among non-taken positions).
(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.

Notice that the cycle involves the same number of repetitions in both parents. Excluding the elements of cycle 1 from the two parents leaves elements with the same number of repetitions in the two parents. So the "leftover" permutations are still of the same repetition class.
(4) Repeat loop (1)–(3) to find more cycles until all position have been marked with a cycle tag.

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. The entries in the mask indicate from which parent each cycle is inherited.
(2) We convert this "cycle" mask into a standard recombination mask by relabeling all the entries c_i in `cycle` as follows: $c_i \rightarrow$ `mask`(c_i) obtaining a new mask `mask'`.
(3) We perform standard mask-based crossover on the two parents using `mask'`.

The following figure shows an example of the generalized cycle crossover.

mask'	B	B	B	A	A	A
cycle	3	3	2	1	1	4
parent A	1	2	2	3	1	3
parent B	2	1	2	1	3	3
offspring	2	1	2	3	1	3

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

The new cycle crossover has the following properties: (i) it preserves repetition class, (ii) it is a proper generalization of the cycle crossover (when applied to simple permutations, it behaves exactly like the cycle crossover), (iii) it searches only a fraction of the space searched by traditional crossover, and (iv) when applied to parent permutations with repetitions of different repetition class, offspring have intermediate repetition class.

3. CYCLE CROSSOVER FOR GRAPH PARTITIONING

Let $G = (V, E)$ be an unweighted undirected graph. *K-way partition* is a partitioning of the vertex set V into K disjoint subsets. A K-way partition is said to be *balanced* if the difference in cardinality 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. The problem is NP-hard.

This is a constrained optimization problem, where the constraint is the balancedness. Among all solutions (balanced or not), the feasible ones are only those that are balanced. A method that does not need to use any repairing mechanism is to have a 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.

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* 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: 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.

4. EXPERIMENTS

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

Initialization: We randomly create 50 chromosomes.

Selection: We use proportional selection with fitness scaling. 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, we repeatedly run swap mutation until the expected Hamming distance between chromosomes before and after mutation is approximately 1 percent of the problem size $|V|$.

Local optimization: As local optimization engine in our genetic algorithm, we use the *direct multiway partitioner* proposed in [1]. 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.

Table 1 shows the results of 32-way and 128-way partitioning. On G graphs, cycle crossover (CX) could not dominate traditional 5-point crossover (TX) on average, but it performed better on the best. On U graphs, CX performed better than TX both on average and in peak performance. In 128-way partitioning, the performance of CX was much better than in the case of 32-way partitioning. TX was always dominated by CX.

5. CONCLUSIONS

In this paper, we have designed a new crossover for permutations with repetitions that naturally suits partition problems. In extensive experimentation, we had demonstrated that this crossover outperforms traditional crossover.

Table 1: Multiway Partitioning Results

⟨ 32-way partitioning ⟩

Graph	5pt crossover (TX)			cycle crossover (CX)		
	Best	Ave	Gen	Best	Ave	Gen
G500.2.5	182	185.18	1091	177	185.59	1269
G500.05	626	637.25	1424	627	637.58	1660
G500.10	1576	1587.23	1984	1575	1587.38	1987
G500.20	4040	4049.44	2247	4039	4049.65	2198
U500.05	112	120.65	1327	113	120.41	1245
U500.10	534	542.75	1163	524	539.67	1263
U500.20	1837	1846.30	1123	1834	1843.80	1170
U500.40	5363	5389.93	1043	5372	5391.77	999

⟨ 128-way partitioning ⟩

Graph	5pt crossover (TX)			cycle crossover (CX)		
	Best	Ave	Gen	Best	Ave	Gen
G500.2.5	316	320.14	844	311	314.54	911
G500.05	850	853.09	869	839	844.13	977
G500.10	1904	1907.78	932	1896	1899.32	1033
G500.20	4568	4571.93	965	4564	4567.94	1004
U500.05	697	704.06	935	695	701.40	941
U500.10	1679	1684.13	913	1673	1682.74	923
U500.20	3836	3841.45	890	3838	3840.37	864
U500.40	8066	8068.54	853	8065	8067.74	845

† From 100 runs.

6. REFERENCES

[1] 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.