Experiments on Commonality in Sequencing Operators

Stephen Chen
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue,
Pittsburgh, PA 15213
chens@ri.cmu.edu
http://www.cs.cmu.edu/~chens

Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
sfs@ri.cmu.edu
http://www.cs.cmu.edu/~sfs

ABSTRACT

Traditionally, crossover operators are based on combination--an operator takes parts from two parents and combines them into an offspring. This paper presents a series of crossover operators based on commonality--an operator preserves the common parts from two parents and uses them as a base on which an offspring solution is built. Experiments on benchmark sequencing problems show that these new commonality-based operators perform better than previously developed combination-based operators. One new operator, Maximum Partial Order/Arbitrary Insertion, is capable of finding new best-known solutions for the Sequential Ordering Problem. Overall, the results support a new commonality-based framework for designing crossover operators.

1. Introduction

A genetic algorithm (GA) has three basic features: a population of solutions, fitness-based selection, and crossover. In isolation, fitness-based selection over a population of solutions causes various building blocks (or schemata) to increase or decrease over time in direct proportion to their observed fitness. These two features provide the basis for exploiting existing knowledge during the search process. The role of crossover is to use these building blocks to explore for better solutions.

The effect of crossover has classically been analyzed from the viewpoint of a single parent. Crossover distributes the parts of each parent over two offspring. This makes it unlikely that either offspring receives "long" schemata directly from a single parent. Thus, it is suggested that the critical building blocks manipulated by crossover are "short, low-order, above-average schemata" [Gol89]. The power of crossover then comes from its ability to preserve and recombine these building blocks when assembling new solutions.

Examining the effect of standard crossover operators from the viewpoint of both parents, it can be seen that schemata common to both parents are unconditionally propagated to offspring solutions. However, many important problems require non-standard representations (e.g. sequences). The crossover operators designed for these problems have not always preserved commonality--the focus of design has been on combination.

Combination and commonality receive equal treatment in Radcliffe's three design principles: respect, transmission, and assortment [Rad91]. Respect is equivalent to commonality--two parents that have a schema H in common should produce an offspring that also has schema H. Transmission is similar to "non-disruptive"--schemata in the offspring should come from one of the parents. Assortment is combination--it should be possible for two non-competing schemata, one from each parent, to both survive in the offspring.

This paper presents a series of crossover operators where the design focus is solely to preserve commonality. To isolate the effect of commonality versus combination, the new commonality-based operators are very similar to existing combination-based operators. On a series of benchmark sequencing problems, the new operators consistently perform better than the existing operators. One new operator, Maximum Partial Order/Arbitrary Insertion (MPO/AI), is capable of finding new best-known solutions for the Sequential Ordering Problem (SOP). Overall, the results support a new framework for designing crossover operators that emphasizes the singular importance of commonality.

The remainder of this paper is divided as follows. Commonality in standard crossover operators is examined in section 2. In section 3, existing non-standard crossover operators designed for the Travelling Salesman Problem are examined, and new commonality-based operators are developed. This examination will cover domain independent operators, heuristic operators, and hybrid operators. The effects of commonality and combination will be the focus of the analysis. In section 4, one of the new commonality-based operators, MPO/AI, is applied to the SOP. Results are discussed in section 5, and finally, in section 6, some conclusions are drawn.

2. Standard Crossover Operators

The power of crossover comes from striking a good balance between the exploitation and exploration of schemata. The traditional analysis has tended to focus on the survival/ disruption probabilities of various schemata contained in a single parent. However, if crossover is examined from the perspective of both parents, a "template" from which the offspring are generated is revealed. (See Figure 1.) This "template" is the maximal common schema of two parents. All standard crossover operators exploit this schema without risk of disruption, and it thus provides an identifiable base from which exploration occurs.

```
Parent 1:     1 0 1 1 0 1 1 0 0 0 1
Parent 2:     0 0 1 0 0 0 1 1 1 0 1

One-point:    1 0 1 1 0 0 1 1 1 0 1
Two-point:    1 0 1 0 0 0 1 0 0 0 1
Uniform:      0 0 1 0 0 1 1 1 0 0 1

Common:       * 0 1 * 0 * 1 * * 0 1
```

Figure 1: Example of standard crossover operators. The common 1's and 0's are guaranteed to be part of the offspring regardless of the crossover and the cut-points.

Experimental results have shown that uniform crossover outperforms two-point crossover, which itself outperforms one-point crossover [Sys89]. If the maximal common schema has n wildcard slots (*), one-point crossover explores a set with $2n$ possible offspring, two-point crossover can explore n^2-n possible offspring, and uniform crossover can explore 2^n possible offspring. Since these are supersets, the best possible offspring for two parents under uniform crossover is at least as good as that under two-point crossover which is at least as good as that under one-point crossover. This dominance relationship may explain the relative performance among the operators.

```
Parent 1:   i b d | e f g | a c h j
Parent 2:   h g a | c b j | i e d f

                  | c b j | a c h j
            i b d | e f g |
                  |       |
Offspring:  e f g | c b j | a h i d
```

Figure 2: Example of OX. To Parent 1, c, b, j are relocated into a sub-tour. The maximal common sub-tour g-a-c is disrupted.

3. Sequencing Operators

The Travelling Salesman Problem (TSP) is perhaps the most popular of all combinatorial optimization problems. Stated simply, the objective of the TSP is to find the shortest Hamiltonian cycle through a set of n cities. The TSP presents an important challenge to GAs because standard crossover operators cannot be meaningfully applied to it. Thus, the TSP has been a major stimulus in the design of non-standard crossover operators.

Non-standard operators can be divided into three main classes: domain independent operators, heuristic operators, and hybrid operators. Domain independent operators do not use any problem-specific local evaluation information--heuristic operators do. Further, the resulting offspring solutions from these operators can be (locally) optimized before adding them to the population--the crossover-local optimization method pairing can be viewed as a hybrid operator.

3.1 Domain Independent Operators for the Travelling Salesman Problem

An early domain independent operator for the TSP is Order Crossover (OX) [Dav85]. It has the form of two-point crossover. Between the cut points, OX takes a sub-tour of elements from Parent 2. Then, starting from the second cut point, OX takes the elements of Parent 1. If the element has been supplied by Parent 2, it is skipped--these elements receive their order from Parent 1. (See Figure 2.) Order crossover works on the hypothesis that good solutions result from good sub-tours and/or good order. Thus, OX combines a sub-tour from parent 2 with order taken from parent 1.

```
Parent 1:   i | b d e f g a | c h j
Parent 2:   h | g a c b j i | d e f

                | g a c b j i | c h j
            i   | b d e f g a |
                |             |
Offspring:  f   | g a c b j i | h d e
```

Figure 3: Example of MST-OX. The maximal common sub-tour g-a-c is no longer disrupted.

Table 1: Performance of MST-OX and OX on 5 TSP instances taken from TSPLIB. Results are for a steady-state GA with a population size of 1000 run for 250 generations. Values are percent distance from known optimum for average of 5 runs.

TSP Instance	Size	Average Random Start Tour	Avg. Best MST-OX Tour	Avg. Best OX Tour
d198	198	+ 947 %	+ 113 %	+ 85 %
lin318	318	+ 1194 %	+ 301 %	+ 352 %
fl417	417	+ 3723 %	+ 760 %	+ 852 %
pcb442	442	+1328 %	+ 461 %	+ 485 %
u574	574	+ 1627 %	+ 630 %	+ 700 %
average		+ 1764 %	+ 453 %	+ 495 %

However, unlike standard crossover operators, OX does not necessarily preserve common sub-tours or order.

The (undirected) Maximal Sub-Tour (MST) that is common to both parents can be preserved by making a trivial modification to OX. Essentially, locate the MST in the second parent, choose the first cut-point to the immediate left of the MST, and choose the second cut-point a random distance to the right of the MST. (See Figure 3.) Using these cut-points, continue with the procedure from OX. The resulting operator is Maximal Sub-Tour Order Crossover (MST-OX). In OX, it is assumed that a good sub-tour (from Parent 2) can be used to build a better solution. In MST-OX, the hypothesis is added that the Maximal Sub-Tour is the best sub-tour to take (from Parent 2). Results indicate that MST-OX performs better than OX. (See Table 1.)

3.2 Edge-Based Heuristic Operators for the Travelling Salesman Problem

To improve the effectiveness of crossover, it has been suggested that for each gene, the best allele from the parents should be selected. For the TSP, this results in Greedy Crossover (GX) [GGR85]. It creates an offspring by combining short edges taken from the parents. The most effective version of GX starts with a random element and builds the tour by picking the shortest edge from either parent that does not introduce a cycle. If all edges from the parents cause cycles, the Nearest Neighbor (NN) heuristic is used instead.

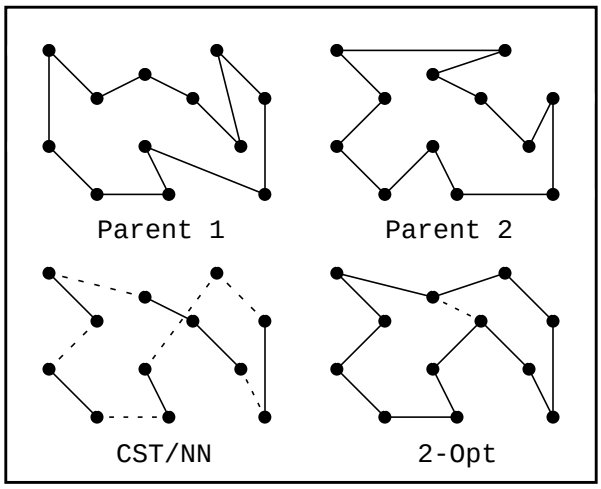

Figure 4: Example of CST/NN and CST/NN-2-Opt. One common edge is lost during 2-Opt.

Instead of combining edges, a new heuristic operator is designed to preserve only the Common Sub-Tours (CST), and thus all the common edges, from two parents. This operator starts with a random common sub-tour, and then connects the remaining elements and sub-tours by using NN. (See Figure 4.) The combined process results in the Common Sub-Tours/ Nearest Neighbor (CST/NN) operator.

Experimentally, solutions found by CST/NN are over 3% better than those found by GX. (See Table 2.) Although GX preserves most of the common edges, it also attempts to minimize disruption (i.e. follow the transmission design principle) by first using edges that are in the parents before it resorts to NN. Unfortunately, it is not known whether the edges taken from the parents are the good edges responsible for the high fitness of the parents, or if they are the poor edges which should be replaced. Being unable to differentiate

Table 2: Performance of CST/NN and GX on 5 TSP instances. Results are for a steady-state GA with a population size equal to problem size, run until 20 generations pass without an improvement. Values are percent distance from known optimum for average of 5 runs.

TSP Instance	Size	Average Best NN Start Tour	Average Best CST/NN Tour	Common Edges From Parents	Average Best GX Tour	Common Edges From Parents
d198	198	+ 12.42 %	+ 5.13 %	100.0 %	+ 10.72 %	99.8 %
lin318	318	+ 17.06 %	+ 9.16 %	100.0 %	+ 16.95 %	100.0 %
fl417	417	+ 16.92 %	+ 13.22 %	100.0 %	+ 10.66 %	100.0 %
pcb442	442	+15.17 %	+ 9.64 %	100.0 %	+ 12.11 %	99.0 %
u574	574	+ 19.92 %	+ 11.36 %	100.0 %	+ 15.98 %	99.9 %
average		+ 16.30 %	+ 9.70 %	100.0 %	+ 13.28 %	99.7 %

the quality of the components in the parents, poor components often "hitch hike" their way into the offspring by tagging along with the good components. Essentially, this weakness of the transmission design principle makes it a poor heuristic for domain dependent operators. Conversely, without being restricted by transmission, CST/NN (like uniform crossover) can explore a larger solution set.

3.3 Order-Based Heuristic Operators for the Travelling Salesman Problem

The Matrix Intersection operator (MI) [FM91] represents a sequence with a Boolean matrix. The matrix is `n x n`, and element `(i,j)` is `1` if element `j` follows element `i` in the sequence, and it is `0` otherwise. A matrix represents a feasible solution if three constraints are satisfied. First, the order must be complete: there are `n(n-1)/2` ones. Second, transitive order is maintained: if `(i,j)=1` and `(j,k)=1`, then `(i,k)=1`. Third, there are no cycles: `(i,i)=0` for all `i`. If two feasible matrices are intersected, a matrix satisfying the last two constraints is created. This underconstrained matrix can be completed by using the Arbitrary Insertion[1] (AI) construction heuristic (modified for MI[2]). The resulting heuristic operator is Matrix Intersection/Arbitrary Insertion (MI/AI).

Similar in form to a commonality-based crossover operator, a deconstruction/reconstruction (Dec/Rec) procedure has been used by an A-team [TS92]. The deconstruction step creates a sub-tour by connecting the common edges of two parents in the order and orientation of Parent 1. (See Figure 5.) The reconstruction step uses AI to complete the sub-tour.

A new commonality-based operator is designed on the observation that any partial order that can be extended by insertion into both parents may be considered to be common to the two parents. The longest such partial order is the Maximum Partial Order (MPO). Using AI to complete the tour, the combined process results in the Maximum Partial Order/Arbitrary Insertion (MPO/AI) operator.

```
Parent 1:  a b c d e f g h i j k l
Parent 2:  a c e d k i j f h g l b

edges:     a-b   d-e   g-h   i-j

Sub-tour:  a b d e g h i j
```

Figure 5: Example of deconstruction. Edges a-b, d-e, and g-h have reverse orientation.

1. Starting with a sub-tour, pick an arbitrary element not in the sub-tour, and insert it in the least cost position. Repeat until all elements have been inserted.
2. The starting sub-tour is the Maximum Partial Order, and elements are inserted subject to additional matrix constraints.

```
- Represent parents by Boolean matrices
- Intersect matrices
- Sum columns to get each element's predecessors
- Build partial order graph
    - Find element with fewest predecessors
    - Attach element to the predecessor with the
          most ordered predecessors
- Find longest path in graph
```

Figure 6: Pseudo-code for MPO.

The pseudo-code for generating the Maximum Partial Order of two parents is given in Figure 6. It is assumed that all tours have the same first (dummy) element and the same orientation[3]. Boolean matrices are used to represent the parent sequences. Each matrix is `n x n`, and element `(i,j)` is `1` if element `j` follows (not necessarily immediately) element `i` in the sequence, and it is `0` otherwise. The intersection matrix has element `(i,j) = 1` if and only if element `j` follows element `i` in both of the parents.

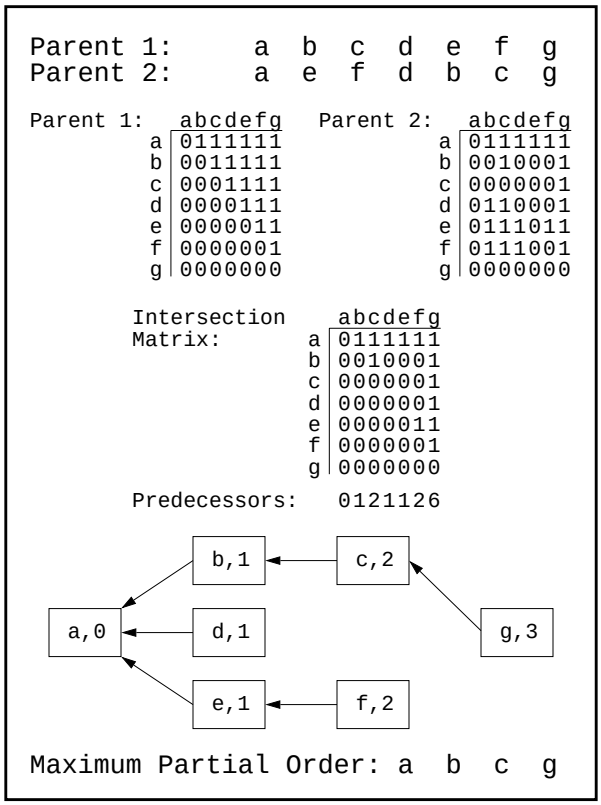

Figure 7: Example of MPO. Element g is after all elements, but elements c and f have the most ordered predecessors (2).

3. Three evenly spaced convex hull elements (e.g. a,b,c) define the orientation of all tours (i.e., a...b...c...). These elements are our segment start elements.

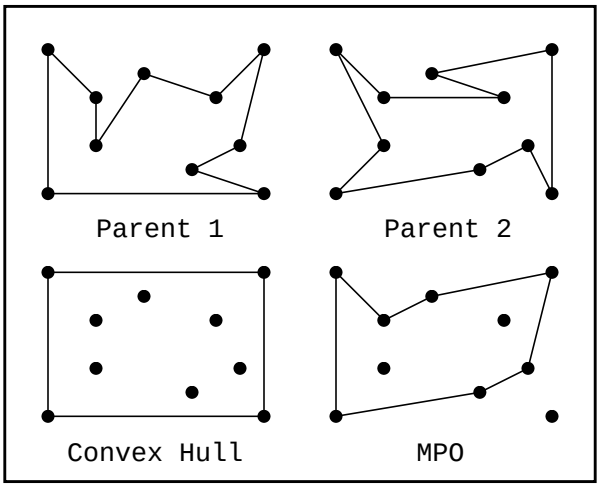

Figure 8: Example of "shape" from convex hull and MPO.

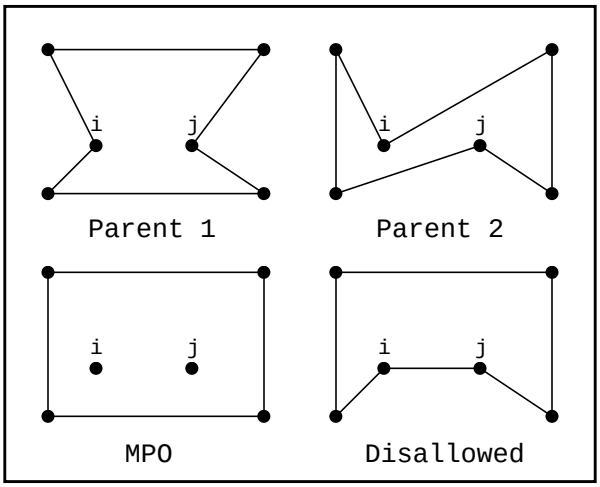

Figure 9: Element `i` precedes `j` in both parents, but AI might determine that `j` should precede `i`.

The column sums give the number of predecessors for each element that are common to both parents. The partial order graph starts with the segment start element, and at each node, it gives the longest partial order for that element. (All ties are broken randomly.) When all elements have been added, the longest path in the graph is the Maximum Partial Order. (See Figure 7.)

The performance of AI is improved when it is given a good tour outline. For example, AI started from the convex hull (CH) develops tours that are about 1% better than those developed when AI is started from three random elements [Rei94]. The MPO constitutes a notion of "shape". While good TSP tours appear "smooth", poor TSP tours appear "jagged". A partial order eliminates much of this "noise" and creates a good tour outline for AI. (See Figure 8.) Conversely, the ordering information in MI that is not part of the MPO is coincidental and it handicaps the performance of AI. (See Figure 9.) In Dec/Rec, AI does not necessarily preserve the edges of deconstruction, so information about good edges is not accumulated. Further, deconstruction can flip edge orientation (see Figure 5), so the tour outline it provides AI is of little benefit. Results indicate that MPO/AI outperforms both MI/AI and Dec/Rec. (See Table 3.)

3.4 Hybrid Operators for the Travelling Salesman Problem

Recognizing that GX solutions often have crossing edges that 2-Opt can correct, the GX-2-Opt hybrid operator was developed [JSG89]. Similarly, 2-Opt can be added to CST/NN to make the CST/NN-2-Opt hybrid operator. Further, for edge representations, 2-Opt is the Binomial Minimal Mutation (BMM) [RS94]. Applying BMM to Random, Respectful Recombination (RRR) [Rad91] (preserve common edges and assign the remaining edges randomly), we get RRR-BMM. (Note: RRR-BMM is equivalent to the operator used in CHC [Esh91].)

The solutions found by CST/NN-2-Opt are better than those found by GX-2-Opt and RRR-BMM. (See Table 4.) Returning the focus to commonality, some common edges can be disrupted by 2-Opt. (See Figure 4.) However, over 90% of the common (CST) edges from the parents are kept after 2-Opt during the first generation of new solutions generated by CST/NN-2-Opt. The number of disrupted

Table 3: Comparison of MPO/AI, MI/AI, and Deconstruction/Reconstruction on 5 TSP instances. Results are for a steady-state GA with a population of 400 run until 10 generations pass without an improvement. Values are percent distance from known optimum for average of 5 runs. Experiment runtimes are given for a SPARC 20.

TSP Instance	Average Best CH/AI Start	Average Best MPO/AI	Runtime (min)	Average Best MI/AI	Runtime (min)	Average Best Dec/Rec	Runtime (min)
d198	+ 3.05 %	+ 0.95 %	3	+ 1.14 %	4	+ 1.31 %	1
lin318	+ 6.04 %	+ 0.63 %	13	+ 1.42 %	15	+ 3.38 %	2
fl417	+ 1.91 %	+ 0.57 %	15	+ 0.61 %	18	+ 0.52 %	7
pcb442	+ 8.97 %	+ 1.84 %	37	+ 3.18 %	47	+ 4.20 %	9
u574	+ 8.45 %	+ 2.20 %	74	+ 2.63 %	73	+ 4.01 %	13
average	+ 5.68 %	+ 1.24 %		+ 1.80 %		+ 2.68 %	

Table 4: Comparison of CST/NN-2-Opt, GX-2-Opt, and RRR-BMM (CHC) on 5 TSP instances. Results are for a steady-state GA with a population of 400 run until 10 generations pass without an improvement. Values are percent distance from known optimum for average of 5 runs.

TSP Instance	Average Best 2-Opt Start	Average Best CST/NN-2-Opt	Kept By 2-Opt	Average Best GX-2-Opt	Kept By 2-Opt	Average Best RRR-BMM	Kept By 2-Opt
d198	+ 3.06 %	+ 0.87 %	89.9 %	+ 1.12 %	87.1 %	+ 1.99 %	80.7 %
lin318	+ 5.16 %	+ 0.31 %	93.5 %	+ 0.77 %	92.1 %	+ 3.04 %	86.8 %
fl417	+ 2.54 %	+ 1.16 %	86.6 %	+ 1.19 %	83.2 %	+1.66 %	76.2 %
pcb442	+ 7.29 %	+ 1.22 %	91.1 %	+ 2.28 %	87.9 %	+ 5.61 %	77.8 %
u574	+ 8.26 %	+ 2.68 %	89.7 %	+ 3.66 %	87.5 %	+ 6.75 %	80.0 %
average	+ 5.26 %	+ 1.25 %	90.2 %	+ 1.80 %	87.6 %	+ 3.81 %	80.3 %

common edges is a measure of how far the initial crossover solution is from a 2-Opt minima. Common edges are disrupted 2.6% more often in GX-2-Opt, so GX is less effective (and less time efficient) than CST/NN in guiding 2-Opt. A similar trend is seen with RRR-BMM.

4. Maximum Partial Order/Arbitrary Insertion on the Sequential Ordering Problem

The MPO/AI operator has also been tested on the Sequential Ordering Problem--a Hamiltonian path problem with additional precedence constraints. Specifically, $d(c_i, c_j) = -1$ if element j must precede (not necessarily immediately) element i. The objective in the SOP is to find the shortest Hamiltonian path subject to satisfying all precedence constraints. Formally, given n elements c_i with edge weights $d(c_i, c_j)$, find a sequence π of the n elements such that the sum is minimized and all precedence constraints are satisfied.

On the SOP, MPO/AI has been able to find new best-known solutions many problem instances. The overall runtime may seem high, but in general, the genetic algorithm

Table 5: Performance of MPO/AI on 16 SOP instances taken from TSPLIB. Results are for 5 runs of a steady-state GA with a population of 500 run until 20 generations pass without an improvement. Overall best MPO/AI solutions in bold improve previous best-known bound. Times are given for execution on a SPARC Ultra 1 (167 MHz).

SOP Instance	Size	Constraints	Average Best AI Start	Average Best MPO/AI	Overall Best MPO/AI	Previous Bounds	Time to Bound (s)	Runtime (s)
ft70.1	71	17	41809	39615	39545	**39313**	NA	76
ft70.2	71	35	43485	40435	**40422**	[39739, 41778]	5.4	73
ft70.3	71	68	46731	42558	**42535**	[41305, 44732]	2.8	48
ft70.4	71	86	55982	53583	**53562**	[52269, 53882]	2.4	47
kro124p.1	101	25	45758	40996	**40186**	[37722, 42845]	5.4	136
kro124p.2	101	49	49056	42576	**41677**	[38534, 45848]	5.2	94
kro124p.3	101	97	63768	51085	**50876**	[40967, 55649]	3.8	103
kro124p.4	101	131	87975	76103	**76103**	[64858, 80753]	2.0	76
rbg323a	325	2412	3466	3161	**3157**	[3136, 3221]	207.6	1566
rbg341a	343	2542	3184	2603	**2597**	[2543, 2854]	70.6	2205
rbg358a	360	3239	3165	2636	**2599**	[2518, 2758]	533.8	4491
rbg378a	380	3069	3420	2843	**2833**	[2761, 3142]	67.6	5354
ry48p.1	49	11	16602	15813	**15805**	[15220, 15935]	2.4	22
ry48p.2	49	23	18071	16676	**16666**	[15524, 17071]	1.6	32
ry48p.3	49	42	22074	19905	**19894**	[18156, 20051]	7.6	29
ry48p.4	49	58	32591	31446	31446	[29967, 31446]	5.0	19

first finds better solutions than a branch and cut algorithm[1] [Asc96] (which provided the previous bounds) in roughly 2% of the CPU time. (See Table 5.)

The Maximum Partial Order of two parent solutions represents a common "structure". MPO is based on order, so the ordering constraints provide structure in the SOP. In contrast, methods based on edge representations (like many integer programming formulations) have their search space convoluted by ordering constraints. On the TSP, branch and cut algorithms can often find the optimum solution.

5. Discussion

Genetic algorithms depend on being able to both exploit existing schemata and explore for new schemata. Combination (assortment) has been viewed as the means to do both. Thus, views on crossover include Syswerda, "... the next step is to combine [good schemata] together in one genome. This is, after all, the overt purpose of crossover." [Sys89], Radcliffe, "Given instances of two compatible [schemata], it should be possible to cross them to produce a child which is an instance of both [schemata]." [Rad91], and Davis, "... the benefits of crossover ... is that crossover acts to combine building blocks of good solutions from diverse chromosomes." [Dav91].

However, these comments do not address the quality of the manipulated schemata. In the Nearest Neighbor construction heuristic, many good edges are selected first, but after myopically "painting itself into a corner", a poor edge must be selected. Compared with the selection of good edges, the selection of poor edges is more dependent on the start condition. Thus, two NN tours are likely to have the same good edges, but different poor edges. This observation leads to the "commonality hypothesis"--schemata common to above-average solutions are above average (good), but uncommon schemata may be below average (poor). Experimentally, GX uses NN and maintains most of the common edges, but it performs over 3% worse than CST/NN. To minimize disruption (maximize transmission) and promote combination (assortment), the GX operator forces itself to take (poor) uncommon edges, thus handicapping NN.

The "commonality hypothesis" creates a new focus for the design of crossover operators. If the common components are to be kept, the task becomes one of extending this partial solution into a complete solution. This can be done randomly, with the left-over parts of the parents (i.e. transmission and assortment), or with a problem-specific construction heuristic. To generalize, a commonality-based crossover operator is designed by following this general framework: 1) identify the maximal common schema of two parents, and 2) complete the solution with a construction heuristic.

When using this framework, the results for MPO/AI, MI/AI, and Dec/Rec demonstrate the importance of a proper match between the basis of commonality and the construction heuristic. The construction heuristic should be aided by, but not restricted by nor destructive to, the common elements.

For hybrid operators, Binomial Minimal Mutation has been suggested as a generic local optimization method [RS94]. It has also been suggested that the local optimizer "... should simply be the best one available ..." [UPL91]. However, a hybrid operator built on a crossover operator that randomly combines subparts of two parents is not significantly different from random multi-start of the local optimizer. Alternatively, using a crossover operator that is designed to preserve commonality for the basis of the local optimizer can improve the performance of the hybrid operator.

6. Conclusion

The original analysis of crossover operators was conducted on standard (positionally dependent) representations. In this case, preserving commonality was a fortunate coincidental effect of combination-based operators. However, for non-standard representations (e.g. sequences), this is often not the case. For sequencing problems, commonality presents an alternative design focus. In this paper, a series of commonality-based crossover operators has been developed. Experimentally, these operators consistently outperform similar previously developed combination-based operators. This implies that commonality may be more important than combination. The following is proposed as a general framework for crossover: 1) identify the maximal common schema of two parents, and 2) complete the solution with a construction heuristic.

Acknowledgments

The work described in this paper was sponsored in part by the Advanced Research Projects Agency and Rome Laboratory, Air Force Material Command, USAF, under grant numbers F30602-95-1-0018 and F30602-97-C-0227, and the CMU Robotics Institute. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Advanced Research Projects Agency and Rome Laboratory or the U.S. Government.

1. In this study, the program was given 300 minutes on a SPARC 2; 1200 minutes for the "rbg" instances.

References

[Asc96] N. Ascheuer. (1996) *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems.* ZIB Technical Report TR 96-3.

[Dav85] L. Davis. (1985) "Applying Adaptive Algorithms to Epistatic Domains." In *Proc. Ninth International Joint Conference on Artificial Intelligence*.

[Dav91] L. Davis. (1991) *Handbook of Genetic Algorithms*. Van Nostrand Reinhold.

[Esh91] E.J. Eshelman. (1991) "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination." In *Foundations of Genetic Algorithms*, G. Rawlins, ed. Morgan Kaufmann.

[FM91] B. Fox and M.B. McMahon. (1991) "An Analysis of Reordering Operators for Genetic Algorithms." In *Foundations of Genetic Algorithms*, G. Rawlins, ed. Morgan Kaufmann.

[GGR85] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht. (1985) "Genetic Algorithms for the Traveling Salesman Problem." In *Proc. of an International Conference on Genetic Algorithms and their Applications*.

[Gol89] D. Goldberg. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[JSG89] P. Jog, J.Y. Suh, and D. Van Gucht. (1989) "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem." In *Proc. Third International Conference on Genetic Algorithms*.

[Rad91] N.J. Radcliffe. (1991) "Forma Analysis and Random Respectful Recombination." In *Proc. Fourth International Conference on Genetic Algorithms*.

[Rei94] G. Reinelt. (1994) *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag.

[RS94] N.J. Radcliffe and P.D. Surry. (1994) "Formal Memetic Algorithms." In *Evolutionary Computing. AISB Workshop*. T.C. Fogarty, ed. Springer-Verlag.

[Sys89] G. Syswerda. (1989) "Uniform Crossover in Genetic Algorithms." In *Proc. Third International Conference on Genetic Algorithms*.

[TS92] S. Talukdar and P. de Souza. (1992) "Scale Efficient Organizations." In *Proc. 1992 IEEE International Conference on Systems, Man and Cybernetics*.

[UPL91] N.L.J. Ulder, E. Pesch, P.J.M. van Laarhoven, H.-J. Bandelt, E.H.L. Aarts. (1991) "Improving TSP Exchange Heuristics by Population Genetics." In *Parallel Problem Solving In Nature*., R. Männer and H.-P. Schwefel, eds. Springer-Verlag.