SAGA: Demonstrating the Benefits of Commonality-Based Crossover Operators in Simulated Annealing

Stephen Chen

School of Analytical Studies and Information Technology, York University
4700 Keele Street
Toronto, Ontario M3J 1P3
sychen@yorku.ca

Abstract

The crossover operator is traditionally viewed as the distinguishing feature and primary strength of a genetic algorithm. This multi-parent operator can recombine the useful features of two parent solutions into a single "super" offspring. However, a new analysis suggests that the primary benefit of crossover operators is the preservation of common components. In creating an offspring solution, crossover can focus its changes on the uncommon components of its two parents. This focus is a surprisingly important feature of genetic algorithms. To demonstrate the contribution of preserving common components to the search process of genetic algorithms, this feature has been isolated and transferred to simulated annealing. Results on the Travelling Salesman Problem indicate that the preservation of common components can lead to significant improvements in the performance of simulated annealing.

Introduction

The commonality hypothesis suggests that schemata common to above-average solutions are above average. Based on this hypothesis, the Commonality-Based Crossover Framework defines crossover as a two-step process: 1) preserve the maximal common schema of two parents, and 2) complete the solution with a construction heuristic. (Chen and Smith 1999) This framework specifies that common schemata should be preserved, but it does not necessarily give preference to the remaining schemata of the parents.

The preservation of common components is an important distinction between crossover and mutation. It causes the offspring produced by crossover to be a subset of those produced by mutation. As a result, crossover can be viewed as a restricted form of mutation – commonality-based mutation. A benefit from this restriction can occur if the common components are above average – changing an uncommon component will increase the probability that the change is beneficial (Chen and Smith 1999). (See figure 1.)

Focusing on their common components, genetic algorithms (GAs) and simulated annealing (SA) both employ "change operators" to create new candidate solutions before applying different selection and control strategies. The commonality hypothesis suggests that crossover-based change operators can benefit from the preservation of common components. To isolate this benefit from the remaining aspects of genetic algorithms, a commonality-based change operator is used with the selection and control strategies of simulated annealing.

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

Common:       1     1 0 1   1   1

Uncommon 1:     0 1       0   1
Uncommon 2:     1 0       1   0
```

figure 1: In the OneMax problem, the objective is to have all 1's. In the above example, a random mutation applied to a parent has a 30% chance of turning a 0 into a 1. However, restricting mutation to the uncommon components increases to 50% the chances of turning a 0

Background

Derived from natural genetics, the crossover operator in genetic algorithms (Holland 1975 and Goldberg 1989) is also synonymous with the concept of recombination – the process by which (different) parent components are recombined into a new offspring. Although the preservation of common components has been previously noted (e.g. Radcliffe 1991), most GA systems focus primarily on how to exploit the differences/variations among parent solutions (e.g. Radcliffe 1991, Mühlenbein 1991, Reeves 1994, and Eshelman, Mathias, and Schaffer 1996). The ability of crossover operators to identify and preserve common components has tended to be an under emphasized feature by the previous research.

If recombination is the key driving force of genetic algorithms, then it should also be a productive feature if it can be transferred to simulated annealing (Kirkpatrick,

Gelatt Jr., and Vecchi 1983). Several attempts to incorporate crossover into simulated annealing have been made (e.g. Wendt and König 1997 and Hiroyasu, Miki, and Ogura 2000). However, it has been remarked that for solutions far away from each other in the search space, "crossover can be compared to the very unpromising effort to recombine animals of different species" (Wendt and König 1997). The above observation actually begins to note that the common components preserved by crossover may actually be more relevant to the search process than the uncommon components that are recombined.

SAGA

Simulated annealing has the ability to probabilistically escape from local optima. Assuming a minimization objective, the simulated annealing process can be visualized as a ball rolling downhill through a landscape of peaks and valleys. Depending on how much "energy" is in the ball, it has the ability to "bounce out" of local minima. When the temperature/energy approaches zero, the ball will come to rest in a final minimum – ideally the global minimum if the cooling schedule has been slow enough.

The above model of simulated annealing does not specify how the "ball" will escape from local minima – it can climb any valley wall with equal probability. If local optima are randomly distributed throughout the search space, then this standard implementation of SA will be well suited. However, the local optima for many combinatorial optimization problems exhibit a "big valley" clustering – random local optima are more similar than random solutions, the similarities among local optima increases with their quality, and the global optima is in the "centre" of the cluster of local optima (Mühlenbein 1991 and Boese 1996).

The standard SA implementation is less well suited for problems with "big valley" fitness landscapes. When escaping from local minima, simulated annealing makes no attempt to determine if it is climbing an "interior" wall (towards better local optima and the global optimum) or an "exterior" wall (towards inferior local optima). (See figure 2.) Although the proof of convergence for simulated annealing does not require this insight, the speed of convergence can be affected.

In genetic algorithms, crossover operators tend to support convergence (through the exploitation of existing components) and mutation operators are used to create diversity. To increase convergence in simulated annealing, a commonality-based mutation operator can be used to replace the standard search operator. The preservation of common components should allow an SA application to concentrate its search efforts on the centre of the "big valley" and thus find better solutions in less time.

Combining the standard operation of SA with a commonality-based crossover/mutation operator from GAs leads to the formation of SAGA. SAGA consists of two standard SA implementations running in parallel. Before each annealing stage, the common components of the two current solutions are recorded. The change operators in SAGA will then use this information to focus their efforts on the uncommon components of the recorded solutions.

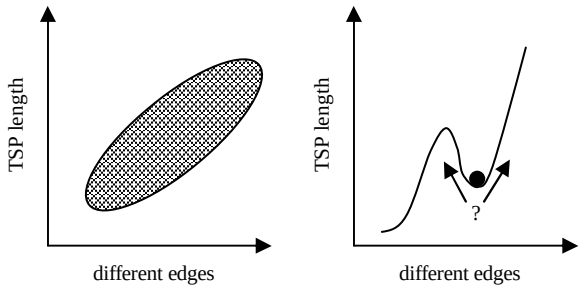

figure 2: In a typical distribution of local minima for the TSP, the best solutions have fewer differences. When escaping from a local minimum in simulated annealing, a change that produces more differences may be moving away from the best solutions.

The Travelling Salesman Problem

GA and SA researchers often use the Travelling Salesman Problem (TSP) as the reference standard for combinatorial optimization problems. In addition to being well-defined (find the least-cost Hamiltonian cycle in a complete graph of N cities), many benchmark problems are readily available through TSPLIB (Reinelt 1994). The following experiments use two popular problems (LIN318 and PCB442) for parameter tuning, and for testing, all 22 symmetric TSPLIB instances with between 1000 and 4000 cities and the Euclidean metric. In general, instances below 1000 cities are no longer interesting (Johnson and McGeoch 2002), and instances above 4000 cities are too computationally expensive for the current implementation[1].

A Base SA Application for the TSP

The base SA (BaseSA) application for the TSP was developed with the following parameters. BaseSA has a geometric cooling schedule, starts from a random

Probability	Neighbourhood Size
1 %	N/2
2 %	N/4
7 %	N/8
15 %	N/16
35 %	N/32
40 %	N/64

figure 3: Probability for each neighbourhood size. After selecting the first edge for a two-opt swap, the second edge will be selected to have up to the neighbourhood size of cities in between.

[1] Source available from http://www.atkinson.yorku.ca/~sychen

μ	LIN318	PCB442
0.90	3.38	5.30
0.91	3.59	5.22
0.92	3.53	5.17
0.93	3.37	5.04
0.94	3.34	4.70
0.95	3.36	4.72
0.96	**3.13**	**4.53**
0.97	3.24	4.84
0.98	6.19	12.57
0.99	16.05	23.07

T_*	LIN318	PCB442
1.0	3.66	4.49
1.1	3.00	4.53
1.2	**2.88**	**4.22**
1.3	3.02	4.43
1.4	3.26	4.33
1.5	3.11	4.86
1.6	3.04	4.66
1.7	3.25	4.66
1.8	3.25	4.59
1.9	2.85	4.48

figure 4: For μ = 0.90 - 0.99, values represent average percentage above optimal for 100 BaseSA runs (10 each for 10 T_* values). For $T_0 = (T_* * 10^{-3} *$ start tour length) / ln(0.5), values represent average percentage above optimal for 10 BaseSA runs with μ = 0.96.

insertion solution with 10 random two-opt swaps applied (to ensure initial exploration), uses 100 annealing stages, applies two hundred thousand two-opt swaps per stage, returns to the best found solution before each new annealing stage, and uses the two-opt neighbourhoods described in figure 3. Re-annealing was also used after 50 stages – if no improvement occurs during an annealing stage and re-annealing has not been used in the past 10 stages, the temperature is increased by 20%.

The last two parameters of initial temperature (T_0) and cooling factor (μ) were determined by using a set of tuning experiments. Starting with a small number of single runs, the approximate ranges were estimated for μ (0.90-0.99) and T_0 (1.0-1.9 * 10^{-3} * length of start solution / ln(0.5)). Dividing each range into ten intervals, BaseSA was run ten times on LIN318 and PCB442 for each of the 100 parameter combinations. Measuring the percent difference above the known optimal for the final solution, it was determined first that μ = 0.96 led to the most robust performance, and that T_* = 1.2 led to the best performance for μ = 0.96 (see figure 4). These two parameters are used in all of the following experiments.

Probability	LIN318	PCB442
0.0	2.98	5.08
0.1	3.06	5.24
0.2	2.83	4.78
0.3	2.94	4.52
0.4	2.93	4.01
0.5	2.62	4.07
0.6	2.84	3.97
0.7	2.51	3.64
0.8	2.38	3.63
0.9	**2.39**	**3.42**
1.0	4.26	4.99

figure 5: Probability that first-edge is specifically selected to be an uncommon edge. Values represent average percentage above optimal for 10 SAGA runs.

Developing SAGA for the TSP

The SAGA implementation uses two parallel BaseSA applications. Each BaseSA run uses all of the same parameters as above except that only one hundred thousand two-opt swaps are now used during each annealing stage. This modification ensures that SAGA applies the same overall number of two-opt swaps as BaseSA.

A two-opt swap changes two edges in a solution. To focus changes on uncommon components, SAGA records the common edges of the two parallel BaseSA runs at the beginning of each annealing stage. Commonality-based two-opt specifically selects an uncommon edge as its first edge. (The second edge is selected as before.) As described in figure 1, changing an uncommon edge should increase the probability than an improving edge is found. Or, as described in figure 2, changing an uncommon edge during a "backward" move should help direct the search process towards the centre of the "big valley" of local minima.

The only additional parameter for SAGA specifies how often to apply commonality-based mutation versus the standard mutation (two-opt). This parameter was also determined through a set of tuning experiments. For each of the 11 probabilities shown in figure 5, SAGA was run ten times on LIN318 and PCB442. Measuring the percent difference above the known optimal for the final solution, it was determined that using commonality-based mutation 90% of the time led to the best results. This parameter is used in all of the remaining experiments.

Experiments

Using the above parameters, BaseSA and SAGA were run 30 times each on the 22 test TSP instances. In the first set of experiments, all parameters are held constant from the tuning experiments. In the second set of experiments, the number of two-opt swaps performed during each annealing stage is increased by a factor of 10 (to two million for BaseSA and one million for SAGA). This alteration helps accommodate the increase in problem size, and it also helps analyze the differences in robustness between BaseSA and SAGA. One-tailed t-tests are performed on each instance to confirm that the improvements observed with commonality-based mutation are statistically significant.

Experiments 1

The results for the first set of experiments are shown in figure 6. The results for SAGA are significantly better than BaseSA for 20 of the 22 instances. Although the performance of both methods degrades noticeably with increasing problem size, the relative performance of SAGA versus BaseSA also appears to increase with problem size. This observation supports the hypothesis

Instance	BaseSA		SAGA		t-test
	avg.	std. dev.	avg.	std. dev.	
dsj1000	4.82	0.62	4.25	0.47	< 1 %
pr1002	5.51	0.75	4.67	0.66	< 1 %
u1060	5.44	0.77	4.93	0.66	< 1 %
vm1084	6.71	0.81	5.79	0.56	< 1 %
pcb1173	7.62	0.60	6.84	0.66	< 1 %
d1291	9.82	1.46	7.26	1.46	< 1 %
rl1304	10.44	1.31	8.10	1.04	< 1 %
rl1323	9.59	1.54	6.91	1.05	< 1 %
nrw1379	5.67	0.55	5.61	0.47	32.2 %
fl1400	7.40	4.54	4.47	2.24	< 1 %
u1432	5.70	0.68	5.54	0.52	15.0 %
fl1577	6.77	1.98	5.13	1.23	< 1 %
d1655	8.66	1.11	7.05	0.90	< 1 %
vm1748	8.63	1.19	6.82	0.75	< 1 %
u1817	10.34	1.00	8.42	0.97	< 1 %
rl1889	11.84	1.31	9.08	0.87	< 1 %
d2103	16.94	1.69	12.75	1.11	< 1 %
u2152	11.53	0.93	9.78	0.85	< 1 %
u2319	5.95	1.68	4.31	0.33	< 1 %
pr2392	11.68	1.78	8.80	0.84	< 1 %
pcb3038	19.63	2.00	12.16	1.38	< 1 %
fl3795	27.24	5.13	21.10	4.20	< 1 %

figure 6: Average percentage above optimal for 30 runs of BaseSA and SAGA with twenty million total two-opt swaps performed.

Instance	BaseSA		SAGA		t-test
	avg.	std. dev.	avg.	std. dev.	
dsj1000	2.44	0.36	2.21	0.39	1.5 %
pr1002	2.77	0.50	2.50	0.43	2.2 %
u1060	3.06	0.59	2.54	0.38	< 1 %
vm1084	4.48	0.82	3.53	0.69	< 1 %
pcb1173	4.04	0.59	3.18	0.49	< 1 %
d1291	6.38	1.00	3.77	0.97	< 1 %
rl1304	6.17	1.32	3.92	1.14	< 1 %
rl1323	5.40	0.93	2.94	0.89	< 1 %
nrw1379	3.18	0.41	2.98	0.36	2.5 %
fl1400	5.42	2.67	2.89	0.82	< 1 %
u1432	2.91	0.42	2.60	0.41	< 1 %
fl1577	3.72	0.36	1.70	1.06	< 1 %
d1655	4.98	0.82	4.02	0.51	< 1 %
vm1748	5.47	0.77	4.06	0.50	< 1 %
u1817	6.14	0.61	4.89	0.59	< 1 %
rl1889	6.80	1.12	5.09	0.67	< 1 %
d2103	8.86	1.33	5.10	0.82	< 1 %
u2152	6.66	0.72	5.24	0.53	< 1 %
u2319	3.52	0.97	2.06	0.36	< 1 %
pr2392	6.98	0.73	5.05	0.50	< 1 %
pcb3038	13.21	3.79	6.34	0.93	< 1 %
fl3795	26.37	5.19	15.42	4.53	< 1 %

figure 7: Average percentage above optimal for 30 runs of BaseSA and SAGA with two hundred million total two-opt swaps performed.

that commonality-based mutation operators can help achieve a higher rate of convergence.

Experiments 2

BaseSA and SAGA have had insufficient time to converge in the first set of experiments. A second set of experiments increases the search effort for both by a factor of ten. The results indicate that SAGA maintains its performance advantage over BaseSA (see figure 7). The consistency of the results (all 22 instances are statistically significant to a one-in-twenty level) supports the conclusion that commonality-based mutation can improve the performance of simulated annealing.

Discussion

The common feature of all local search techniques (e.g. hill climbing, simulated annealing, tabu search (Glover 1989), genetic algorithms, evolutionary strategies (Fogel, Owens, and Walsh 1966 and Schwefel 1981), etc) is the change operator. The change operator produces a new candidate solution from an existing candidate solution(s) by keeping some parts and changing some parts. Implicitly, the "good" parts of a solution should be kept, and the "bad" parts of a solution should be changed. However, many change operators only focus on what is changed. For example, two-opt can improve a solution by replacing two longer edges with two shorter edges (that connect the same four cities), but it pays no attention to the remainder of the solution which has been left unchanged.

The commonality hypothesis provides some insight into which parts of a solution a change operator should keep unchanged. If the common components are above average (and the uncommon components are below average), then a change operator should keep the common components and change the uncommon components. To identify common components, a multi-parent change operator (e.g. crossover) is required.

Standard crossover (applied to a binary bit-string representation) will preserve common components. In taking pieces from a random parent, something that is common to both parents will definitely be taken (see figure 1). However, this is not necessarily the case for non-standard crossover operators which work on different representations (e.g. permutation sequences used to represent TSP solutions). For some early TSP crossover operators which did not preserve common components (e.g. Order Crossover – Davis 1985), it has been shown that minor modifications to preserve common components can improve the performance of these operators (Chen and Smith 1998).

This previous research in genetic algorithms has shown that the benefits of preserving common components (in the crossover change operator) are independent of the effects of fitness-based selection. Since local search techniques all employ change operators (before applying

their own selection and control strategies), these techniques should all be able to benefit from the preservation of common components. However, identifying common components requires multiple solutions, and many local search techniques use point search and single-parent operators. One solution to this problem is to use parallel runs. Commonality-based mutation can then be used as the single-parent operator.

The above solution has been tested on simulated annealing. The two parallel runs provide the multiple solutions from which common components can be identified. With information about common components available, the normal change operator (e.g. two-opt) can be modified to focus on changing uncommon components. This single difference between BaseSA and SAGA has been designed to isolate any performance improvements to be solely caused by the benefits of preserving common components.

The two parallel SA runs in SAGA each use half of the computational effort as the single benchmark SA run. Therefore, the improvements are not due to using multiple runs and picking the best (see figure 5). With little or no use of commonality-based mutation, the parallel runs of SAGA cannot match the search performance of a single BaseSA run. To compensate for the reduced search effort of each parallel run, it is necessary to have a more focused search to achieve the same and/or better results. The preference to keep common components and to change uncommon components can help provide this search focus for problems with "big valley" fitness landscapes.

Conclusions

The results of SAGA demonstrate the benefits of preserving common components in crossover operators. In particular, the parallel SA runs in SAGA communicate no information about their uncommon components. Therefore, the effects of recombination (the traditionally perceived strength of crossover) have been completely eliminated. The effects of the selection and control strategies of genetic algorithms (e.g. fitness-based selection and population diversity) have also been eliminated by using the infrastructure of simulated annealing. The performance improvements in simulated annealing demonstrate that the preservation of common components is an important (and previously hidden) advantage of crossover operators and genetic algorithms.

References

Boese, K.D. 1996. Models for Iterative Global Optimization. Ph.D. diss., Computer Science Department, University of California at Los Angeles.

Chen, S. and Smith, S.F. 1998. Experiments on Commonality in Sequencing Operators. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*. Morgan Kaufmann.

Chen, S. and Smith, S.F. 1999. Introducing a New Advantage of Crossover: Commonality-Based Selection. In *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

Davis, L. 1985. Applying Adaptive Algorithms to Episatatic Domains. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*.

Eshelman, L.J., Mathias, K.E., and Schaffer, J.D. 1996. Convergence Controlled Variation. In *Foundations of Genetic Algorithms 4*, Belew, R. and Vose, M., eds. Morgan Kaufmann.

Fogel, L.J., Owens, A.J., and Walsh, M.J. 1966. *Artificial Intelligence through Simulated Evolution*. Wiley.

Glover, F. 1989. Tabu Search—Part I. In *ORSA Journal on Computing*, 1:190-206.

Goldberg, D. 1989. Genetic Algorithms in *Search, Optimization, and Machine Learning*. Addison-Wesley.

Hiroyasu, T., Miki, M., and Ogura, M. 2000. Parallel Simulated Annealing using Genetic Crossover. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*. ACTA Press.

Holland, J. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Johnson, D.S. and McGeoch, L.A. 2002. Experimental Analysis of Heuristics for the STSP. In *The Traveling Salesman Problem and Its Variations*, Gutin G. and Punnen A.P., eds. Kluwer Academic Publishers.

Kirkpatrick, S., Gelatt Jr., C.D., and Vecchi, M.P. 1983. Optimization by Simulated Annealing. In *Science*, 220:671-680.

Mühlenbein, H. 1991. Evolution in Time and Space--The Parallel Genetic Algorithm. In *Foundations of Genetic Algorithms*, Rawlins, G., ed. Morgan Kaufmann.

Radcliffe, N.J. 1991. Forma Analysis and Random Respectful Recombination. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann

Reeves, C.R. 1994. Genetic Algorithms and Neighbourhood Search. In *Evolutionary Computing: AISB Workshop*, Fogarty, T.C., ed. Springer-Verlag.

Reinelt. G. 1994. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag.

Schwefel, H.-P. 1981. *Numerical Optimization of Computer Models*. Wiley.

Wendt, O. and König, W. 1997. Cooperative Simulated Annealing: How much cooperation is enough? Technical Report, No. 1997-3, School of Information Systems and Information Economics at Frankfurt University.