Isolating the Benefits of Respect

Stephen Chen[1] and Gregory Pitt[2]

[1] School of Analytic Studies and Information Technology, York University
4700 Keele Street, Toronto, Ontario M3J 1P3
sychen@yorku.ca
http://www.atkinson.yorku.ca/~sychen
[2] Department of Mathematics, York University
4700 Keele Street, Toronto, Ontario M3J 1P3
greg.pitt@mathstat.yorku.ca

Abstract. The three mechanisms of crossover are transmission, assortment, and respect. Of these three mechanisms, assortment (i.e. recombination) is traditionally viewed as the primary feature and key advantage of crossover. However, respect (the preservation of common components) is also a feature that is unique to multi-parent operators like crossover – it takes two (or more) parents to have/identify common components. The effects of respect are isolated from all other aspects of genetic algorithms by using a parallel implementation of simulated annealing. In this implementation, the preservation of common components is used to focus the search process and this focus has improved the performance of simulated annealing on the TSP. Since only the mechanism of respect is transferred from genetic algorithms to simulated annealing, these experiments isolate and demonstrate the benefits of respect.

1 Introduction

The three primary features of a genetic algorithm (GA) are a population of solutions, fitness-based selection, and crossover [8][10]. Other heuristic search techniques (e.g. ant colony optimization [5] and evolution strategies [7][17]) may also use a population of solutions and/or fitness-based selection. Therefore, crossover is generally viewed as the distinguishing feature of a genetic algorithm [4][8].

The mechanisms of crossover include "respect", "transmission", and "assortment" [14]. The principle of respect is that the common components of two parent solutions should be preserved in the offspring. Transmission states that all components in the final offspring should have come from one of the two parents, and assortment is the equivalent of recombination – parents with two non-competing traits should be able to produce an offspring with both features.

Recombination has received much of the early attention of researchers (e.g. [4][8][18]), and its acclamation as "the overt purpose of crossover" [18] is still highly prevalent. An analysis of what gets recombined has led some researchers (e.g. [1][6]) to focus on transmission. The resulting systems (e.g. Convergence Controlled Variation [6]) can be viewed as employing population-based crossover operators – the role of transmission is to provide the components that get recombined. With this

general emphasis on recombination, the presence and preservation of common components has even been seen as a negative since they can cause premature convergence and genetic drift (e.g. [6][16]).

The previous research suggests that respect could be merely a (passive) side effect of transmission and assortment. To be legitimized as a design principle for crossover [14], it is important to demonstrate that its advantages (e.g. reduced implicit mutation) outweigh its disadvantages (e.g. premature convergence). Previous attempts to make this demonstration have failed to sufficiently isolate the effects of preserving common components from the remaining aspects of genetic algorithms (e.g. [3][14]). As a result, the improved performance of these GAs has not been clearly attributable to the mechanism of respect.

To isolate the effects of respect, this feature has been transferred into simulated annealing (SA) [12]. Simulated annealing is a point-search technique, so its search operators are normally unable to use respect and recombination – these mechanisms require a second solution from which to select common/uncommon components. In the following experiments, a parallel implementation of simulated annealing uses a modified search operator that can share information on common components. However, no information on uncommon components is exchanged, so the effects of respect are completely isolated from the effects of recombination. The resulting system which combines features from simulated annealing and genetic algorithms is called SAGA.

The results for SAGA are compared with the results from a similar, non-parallel implementation of simulated annealing for the TSP. Using the same number of two-opt swaps for both, the performance of SAGA is consistently and significantly better. These experiments isolate the effects of preserving common components, and the results demonstrate that respect is a beneficial mechanism that can be employed by multi-parent operators. Therefore, respect should be viewed as an important design principle for the development of crossover and other multi-parent operators.

2 Background

Genetic algorithms and simulated annealing can both be viewed as iterative improvement techniques – techniques that develop new candidate solutions by repeatedly making "small" changes to existing candidate solutions. These techniques can be analyzed by focusing on their "search operators" and "control strategies". A search operator creates new solutions by making some (usually small) changes to existing solutions. All of the remaining aspects of an iterative improvement technique can be included as part of its control strategy [19] – which search operator(s) to apply; which solutions to keep, change, or remove; and when to stop.

Using this model, genetic algorithms [8][10] are an iterative improvement technique based on Darwinian evolution and sexual reproduction. The search operator is crossover – take two parent solutions and create an offspring solution by exchanging/recombining solution components in a manner similar to genetic crossover. The control strategy is survival of the fittest – from a population of potential parent solutions, favour the fittest solutions for survival and reproduction.

Similarly, simulated annealing [12] is an iterative improvement technique based on the physical process of metal annealing. The search operator can be any reasonable neighbourhood search operator like an exchange operator – select some (random) parts of a solution and rearrange them. The control strategy is probabilistic selection based on a cooling schedule – accept all improving changes, and accept non-improving changes as a function of the "temperature".

Since exchange operators can be used with other control strategies (e.g. hill climbing, tabu search, etc), the primary advantage of simulated annealing is its cooling schedule-based control strategy. Further, other evolutionary computation techniques use fitness-based population search, so the primary advantage of genetic algorithms is the crossover-based search operator. To combine these two key features of simulated annealing and genetic algorithms, SAGA takes the mechanism of respect from crossover and adds it to an implementation of simulated annealing.

3 SAGA

SAGA starts with the control strategy for simulated annealing which is designed to allow probabilistic escapes 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.

This control strategy 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 increase with their quality, and the global optimum is in the "centre" of the cluster of local optima [2][13].

The standard control strategy for simulated annealing is not as 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 1.) Although the proof of convergence for simulated annealing does not require this insight, the practical (time-based) performance of an SA implementation may be affected.

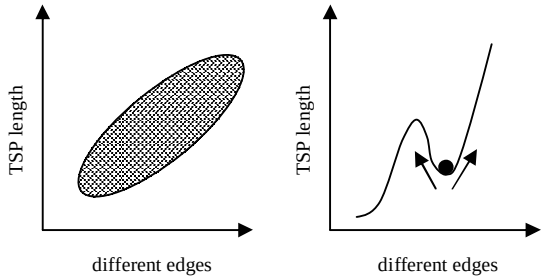

Fig. 1. In a typical distribution of local minima for the TSP, the best solutions have more common edges [2]. When escaping from a local minimum in simulated annealing, a change that produces additional different/uncommon edges may be moving away from the best solutions

In problems with big-valley fitness landscapes, changing a previously common component may increase the number of different edges and lead the search process to climb an exterior wall. Therefore, SAGA will use a "respectful" search operator to help direct the search process towards the centre of the big valley. To add the mechanism of respect to simulated annealing, multiple solutions are required. SAGA provides these solutions by using two parallel SA runs and storing the best solution after each "annealing stage".

SAGA	**Each parallel run**
	create initial solution
set initial temperature	
perform n annealing stages:	
• record common components of current solutions	
	• apply exchange operator m times
	• return to best found solution
• update temperature	
return best found solution	

Fig. 2. Operation of SAGA. The two parallel runs are synchronized at the beginning of each "annealing stage". Information on common components is shared at this time

To best isolate the effects of respect, SAGA uses "annealing stages" – the parallel SA runs are synchronized to use a single cooling schedule. Information on common components is shared before each annealing stage (see figure 2). With this information on common edges, a respectful search operator can favour changing solution components which are already different – changing these components cannot create an additional difference whereas changing a common component can. Compared to normal SA implementations, SAGA uses respect to concentrate its search efforts towards the centre of the big valley where the best solutions are found.

4 The Travelling Salesman Problem

GA and SA researchers often use the Travelling Salesman Problem (TSP) as a standard combinatorial optimization problem. 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 [15]. The following experiments use a popular smaller problem (PCB442) for parameter tuning, and testing is done on 5 symmetric TSPLIB instances of at least 1000 cities [11] (i.e. dsj1000, d1291, fl1400, fl1577, and pr2392) that offer a good range of sizes.

5 A Base SA Application for the TSP

The base SA application for the TSP (BaseSA) was developed with the following parameters. BaseSA starts from a random solution, uses a geometric cooling schedule ($\mu = 0.9$) with 500 temperature cycles/annealing stages, applies two million two-opt swaps per stage, and returns to the best found solution at the end of each temperature cycle. Since a fixed number of annealing stages are used, the temperature is not decreased if the best solution does not improve during that annealing stage. To determine the initial temperature, a preliminary set of 50 two-opt swaps is used where only improving steps are accepted. At the end of this set, the initial temperature starts at → the average length of the attempted backward steps / ln(0.5).

6 Developing SAGA for the TSP

SAGA has been developed for the TSP by using two parallel runs of BaseSA. Each instance of BaseSA uses all of the above parameters except that only one million two-opt swaps are now used during each annealing stage. This modification ensures that SAGA uses the same overall number of two-opt swaps (and thus a similar amount of computational effort) as the benchmark implementation of BaseSA.

A two-opt swap changes two edges in a solution. To implement the mechanism of respect so that common components can be preserved, SAGA records the common edges of the two parallel BaseSA runs at the beginning of each annealing stage (see figure 2). The respectful two-opt operator will then select one uncommon edge and one random edge. This preference to change uncommon components will help preserve common components which should help direct the overall search process towards the centre of the "big valley" of local minima.

The only parametric difference between SAGA and BaseSA is an additional parameter to specify how often to apply the respectful two-opt operator versus the standard two-opt operator. This parameter was chosen from the 11 probabilities shown in table 1 by conducting a set of tuning experiments on PCB442. Measuring the percent difference above the known optimal for the final solution in ten SAGA trials, it was determined that the respectful two-opt operator should be used 90% of

the time (with normal two-opt swaps being used for the remaining 10%). This parameter is used in all of the experiments presented in section 7.

Table 1. Probability that the first-edge is specifically selected to be an uncommon edge. Values represent average percentage above optimal for 10 SAGA trials – 300 generations, 60 million total two-opt swaps. Average performance of 10 BaseSA trials is provided for comparison

Probability	PCB442
0.0	4.21
0.1	3.60
0.2	3.92
0.3	3.68
0.4	3.76
0.5	3.49
0.6	3.01
0.7	3.17
0.8	3.08
0.9	**2.79**
1.0	3.40
BaseSA	3.79

The results of the above tuning experiments also indicate that the improvements in SAGA are not due strictly to the use of two parallel runs. With a 0% probability of using the respectful two-opt operator, SAGA is essentially two (smaller and less effective) independent runs of BaseSA. Compared to these two independent runs, the performance of SAGA tends to improve with the increased use respect. The key exception to this trend is for a probability of 100%. It appears that SAGA benefits from divergent changes just like traditional genetic algorithms benefit from mutation.

7 Experiments

Using the above parameters, thirty trials of BaseSA and SAGA were run on each of the 5 test TSP instances. The results for the experiments are shown in table 2. The respectful two-opt operator in SAGA has led to consistent and significant improvements compared to the benchmark implementation of BaseSA.

Table 2. Average percentage above optimal for 30 trials of BaseSA and SAGA with one billion total two-opt swaps performed. One-tailed t-tests show that the chance of the SAGA results being the same as the BaseSA results is less than 0.01% for all five instances

Instance	BaseSA		SAGA	
	avg.	std. dev.	avg.	std. dev.
dsj1000	4.54	0.74	2.27	0.39
d1291	8.87	1.41	3.12	1.12
fl1400	3.07	1.04	2.00	0.88
fl1577	6.47	1.58	0.64	0.55
pr2392	9.24	1.37	6.53	0.56

It is important to note that only the relative results or differential performance should be considered from the above experiments. Despite the large number of iterations/two-opt swaps performed, the absolute results of BaseSA and (to a lesser extent) SAGA are still moderate. Attempts to tune BaseSA (e.g. increasing μ, the number of temperature cycles, the number of two-opt swaps per temperature cycle, etc) have led to only minimal improvements. However, the performance of BaseSA is a controlled parameter in these experiments – any strengths or weaknesses in BaseSA should be reflected equally in the performance of both BaseSA and SAGA. Therefore, the superior performance of SAGA as compared to BaseSA is a clear demonstration of the benefits of respect.

8 Discussion

The crossover operator consists of three mechanisms: respect, transmission, and assortment. Negative arguments against crossover (and genetic algorithms) have often targeted only assortment in isolation. However, since a crossover operator is more than just recombination, these previous arguments are weakened by the demonstration that respect also provides important benefits to a crossover operator.

For example, other researchers (e.g. [9][20]) have attempted to combine the benefits of genetic algorithms and simulated annealing by using (only) recombination to coordinate parallel SA runs. These implementations have had mixed results and produced comments like "crossover can be compared to the very unpromising effort to recombine animals of different species" [20]. However, medical experiments for humans are often done on animals that we would not likely receive any benefit if we were to recombine with them (e.g. rats, monkeys, etc). These experiments exploit our common features with these animals, and SAGA exploits the common components of two solutions to coordinate parallel SA runs.

In evolutionary computation, the advantages and disadvantages of sexual reproduction (i.e. crossover) versus asexual reproduction (i.e. mutation) are often discussed. It is clear that the offspring producible by crossover are a subset of those producible by mutation. Therefore, it can be argued that crossover is merely a restricted (and weakened) form of mutation, and that genetic algorithms should subsequently be less effective than mutation-based evolutionary computation methods.

This reduction to the set of offspring producible by crossover is caused by respect – specifying that common components must be preserved reduces the variety of offspring that can be produced. However, this restriction can improve the chances that a change/mutation will lead to an improved solution. In figure 3, it is shown for the OneMax problem that a random mutation has a lower probability of improving the solutions than a restricted (respectful) change/mutation. This restriction caused by respect has also been used in SAGA to help direct the search process towards the better solutions found at the centre of the big valley.

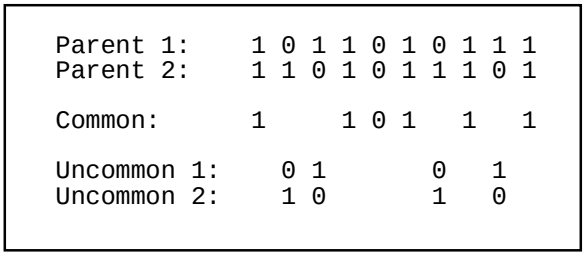

Fig. 3. In the OneMax problem, the objective is to have all 1's. A beneficial change/mutation occurs when a 0 is turned into a 1. In the above example, an unrestricted mutation applied to either parent has only a 30% chance of being beneficial. However, a "respectful" mutation that is applied to only the uncommon components of a parent has a 50% chance of being beneficial [3]

9 Conclusions

The three mechanisms of crossover have been presented as respect, transmission, and assortment. However, respect (the preservation and accumulation of common components) is also associated with negative aspects such as premature convergence and genetic drift. The experiments with SAGA isolate and demonstrate the benefits of respect – another feature that gives crossover operators and genetic algorithms an advantage over single-parent operators and point search techniques.

References

1. Baluja, S.: An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics. Carnegie Mellon Technical Report CMU-CS-95-193 (1995)
2. Boese, K.D.: Models for Iterative Global Optimization. Ph.D. diss., Computer Science Department, University of California at Los Angeles (1996)
3. Chen, S., Smith, S.F.: Introducing a New Advantage of Crossover: Commonality-Based Selection. In GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann (1999)
4. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold (1991)
5. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents. In IEEE Transactions on Systems, Man, and Cybernetics – Part B, Vol. 26-1 (1996) 29-41
6. Eshelman, L.J., Mathias, K.E., Schaffer, J.D.: Convergence Controlled Variation. In: Belew, R., Vose, M. (eds.): Foundations of Genetic Algorithms 4, Morgan Kaufmann (1996)
7. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence through Simulated Evolution. Wiley (1966)
8. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)

9. Hiroyasu, T., Miki, M., Ogura, M.: Parallel Simulated Annealing using Genetic Crossover. In Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems. ACTA Press (2000)
10. Holland, J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press (1975)
11. Johnson, D.S., McGeoch, L.A.: Experimental Analysis of Heuristics for the STSP. In: Gutin G., Punnen A.P. (eds.): The Traveling Salesman Problem and Its Variations. Kluwer Academic Publishers (2002)
12. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. In Science, Vol. 220 (1983) 671-680
13. Mühlenbein, H.: Evolution in Time and Space--The Parallel Genetic Algorithm. In: Rawlins, G. (ed.): Foundations of Genetic Algorithms. Morgan Kaufmann (1991)
14. Radcliffe, N.J.: Forma Analysis and Random Respectful Recombination. In Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann (1991)
15. Reinelt. G.: The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag (1994)
16. Schaffer, J.D., Mani, M., Eshelman, L.J., Mathias, K.E.,: The Effect of Incest Prevention on Genetic Drift. In: Banzhaf, W., Reeves, C. (eds.): Foundations of Genetic Algorithms 5, Morgan Kaufmann (1998)
17. Schwefel, H.-P.: Numerical Optimization of Computer Models. Wiley (1981)
18. Syswerda, G: Uniform Crossover in Genetic Algorithms. In Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann (1989)
19. Talukdar, S.N., de Souza, P.: Scale Efficient Organizations. In Proceedings of 1992 IEEE International Conference on Systems, Man and Cybernetics (1992)
20. Wendt, O., König, W.: Cooperative Simulated Annealing: How much cooperation is enough? Technical Report, No. 1997-3, School of Information Systems and Information Economics at Frankfurt University (1997)