Proceedings of the GECCO-2003 Late Breaking Papers, 235-239, Chicago, IL, July 12-16, 2003.

GeneRepair - A Repair Operator for Genetic Algorithms

George G. Mitchell **Diarmuid O'Donoghue** **David Barnes** **Mark McCarville**

Department of Computer Science
National University of Ireland Maynooth,
Co. Kildare, Ireland.
georgem@cs.may.ie

Abstract

In this paper we present the outcome of two recent sets of experiments to evaluate the effectiveness of a new adjunct genetic operator GeneRepair. This operator was developed to correct invlaid tours which may be generated following crossover or mutation of our particular implementation of the genetic algorithm. Following implementation and testing of our genetic algotihm with GeneRepair we found a significant positive side in our results. Using GeneRepair along side traditional corsover and mutation operators we have been able to travers the search space of a problem and generate very good results in an extremely efficent manner, in both time and number of evaluations required.

1 INTRODUCTION

In this paper we present a novel approach to solving permutation problems that uses only standard crossover and standard mutation. We isolate the problem constraints in a separate operator, which operates as an adjunct operator to the standard set of genetic operators.

This approach is applicable to any problem domain where the solution constraints can be identified in the gene string. In this paper we explore two different types of permutation problems. We look at the Traveling Salesman Problem (TSP), which is a well-known NP-Complete problem, and the Vehicle Routing Problem (VRP), which is an NP-Hard problem (Garey and Johnson, 1979). The TSP involves visiting all cities on a map, generating the shortest total tour distance. The VRP involves finding the delivery schedule for N cities using M trucks of finite capacity, again for the shortest total distance traveled by all trucks.

As Mitchell (1999) points out "*some type of encoding require specially defined crossover and mutation operators... like the Traveling Salesman Problem in which the task is to find a correct ordering for a collection of objects*".

2 REPRESENTATION AND OPERATORS

The natural choice of representation for the TSP and VRP is an Order-based representation. These have been successfully applied to the TSP and VRP problems by Fogel (1988, 1993 and 1993a), Banzhaf (1990), Ambati (1991) and Pereira *et al* (2002). Additionally, the genetic operators employed must also be Order-based. If either the representation or the operators do not respect the Order based nature of the problem, then invalid solutions will be generated.

First, we looked at the crossover operators that respect the Order-based nature of permutation problems, and prevent the introduction of errors such as invalid tours (Mitchell, 1999). The order preserving crossover operators that have been developed include: Order Crossover (Syswerda, 1991), Modified Crossover (Davis, 1985), Partially Mapped Crossover (Goldberg and Lingle, 1985), Cycle Crossover (Oliver *et al.,* 1987), 2-quick / 2-repair (Gorges-Schleuter, 1989), plus a number of less frequently used crossover operators (Crawford and Wainwright, 1996).

Secondly, we looked at Order-based mutation operators developed for Order-based problems. These include: Displacement Mutation (Michalewicz, 1992), Exchange Mutation (Banzhaf, 1990), Insertion Mutation (Fogel 1988 and Michalewicz 1992), Simple Inversion Mutation (Holland 1975 and Grefenstette *et al.,* 1985), Inversion Mutation (Fogel, 1993 and 1993a) and other order preserving mutation (Larrañaga, 1999).

We present a solution for Order-based problems that uses only standard crossover and standard mutation. To counteract the invalid tours that occur as a result, we introduce GeneRepair - a genetic repair operator that has a number of positive effects: It allows the use of standard GA libraries, with the addition of a single repair operator for permutation problems. It simplifies the understanding of the GA, by allowing the use of standard crossover and mutation for Order-based problems. Finally, it removes problem specific activities from the genetic operators themselves, and isolates it in a single intra-generation operation.

Proceedings of the GECCO-2003 Late Breaking Papers, 235-239, Chicago, IL, July 12-16, 2003.

3 GENEREPAIR

The GeneRepair enhanced genetic algorithm operates in the manner of traditional genetic algorithms, and can be summarized as follows:

1. Generate the initial population P(0) at random and set i = 0;
2. Evaluate the fitness of each individual in P(i);
3. Select parents from P(i) based on their fitness.
4. Apply standard crossover
5. Apply standard mutation.
6. Apply GeneRepair.
7. Repeat until convergence.

Although the VRP is NP-Hard and TSP is NP-Complete, they may be characterized by two separate facets: Optimization and Permutation. Responsibility for optimization lies with the standard genetic algorithm, which effectively remains unchanged from Holland (1975). Responsibility for only allowing valid permutation in the population lies solely with the GeneRepair operator.

3.1 SOLUTION CONSTRAINTS

Combinatorial problems like the TSP and VRP place constrains on the valid solutions. Solutions are only valid when all N cities in the problem are present in the solution. Thus, we use a fixed-length chromosome to represent our tours. Furthermore, a solution is only considered valid when all cities are represented once only in the solution, and no cities are absent. These constraints act as a trigger for the application of the GeneRepair operator.

Non order-preserving crossover (above) can cause a violation of the validity constraint, by combining parent strings, which result in invalid offspring. See Figure 1.

Similarly, non order-preserving mutation operators can also generate invalid solutions. This happens when mutation randomly inserts a city that already exists in the solution.

Parent 1	0 1 2 3 4 | 5 6 7 8 | 9
Parent 2	8 4 1 6 3 | 7 9 2 0 | 5
Child 1	0 1 2 3 4 | 7 9 2 0 | 9
Child 2	8 4 1 6 3 | 5 6 7 8 | 5

Figure 1: Constraint violation by 2-Point Crossover.

In practice, GeneRepair examines each tour in turn, enforcing the following:

1. Correct number of cities in the tour
2. No duplicate cities
3. No missing cities

These constrains invoke the GeneRepair operator, and identifies the string the location of duplicate cities (see Figure 2).

Detection of invalid cities:

Child 1	0 1 2 3 4 7 9 2 0 9

Figure 2: GeneRepair- Invalid cities identified.

3.2 REPAIR

Knowing the location of the offending cities, GeneRepair replaces these cities iteratively with valid cities retrieved from a corrective template. The first strategy investigated was to replace the duplicate cities with the missing cities, according to a pre-determined template (see Figure 3).

GeneRepair Template	0 1 2 3 4 5 6 7 8 9
(i) Child 1	0 1 2 3 4 7 9 2 0 9
(ii)Child $1_{\text{GeneRepaired}}$	0 1 2 3 4 7 9 5 6 8

Figure 3: GeneRepair- correction of tour.

The majority of GeneRepair replacements were performed in a left-to-right manner - replacing the left-most duplicate city first. Additionally, the replacement city was retrieved from the template also in a left-to-right manner. However, brief evaluation of a random replacement technique, randomly selecting the replacement city from the template was also evaluated. Initial results show no identifiable difference between the two techniques.

The replaced city is selected according to a corrective template. Three different types of template were investigated:

1. Static template. This consisted of a preset valid tour, and remained constant throughout.
2. Parent-based Template. Select the fitter parent, and use that as the corrective template. This template varied for every corrected individual.
3. Random Template. For each corrected individual a new template of random numbers was generated, within the validity constraints of the TSP or VRP problem.

Each of these techniques was tested on a select number of VRP and TSP problems. The parent-based solution produced the worst results. Both random and fixed template solutions produced good results, with the randomly generated template producing marginally better results.

4 EXPERIMENT 1 - VRP

We now evaluate the performance of GeneRepair on a number of VRP benchmark problems selected from the Augerat Set-A (A-n32-k5, A-n33-k6, A-n34-k5, A-n36-k5, A-n39-k6). These include the best-known solutions to each problem. These problems range from 32 cities to 39 cities, using either 5 or 6 trucks for the solution.

We firstly compare two implementation, one with GeneRepair and the other without. The allowed us assess the performance of the GeneRepair operator in conjunction with the standard genetic operators.

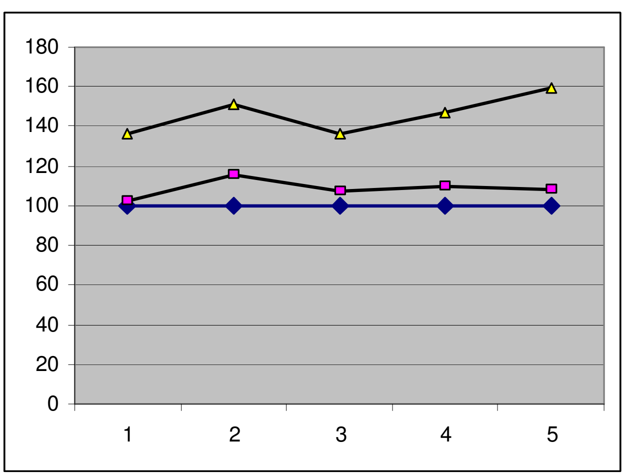

Figure 4: Comparison to best-known solutions

In Figure 4 we see that the solutions produced with GeneRepair, are significantly better than those produced without. Furthermore, the GeneRepair solutions consistently approach the best-known solutions.

Our experiments were purely a proof of concept, and no special effort was made to optimize the genetic parameters in order to achieve short tours. Specifically, we only used truncation selection with just two different truncation parameters. Additionally, only two mutation rates were investigated. We expect that significant improvements can be made to the shortest tours we produce, by optimizing the genetic parameters.

Next we show that GeneRepair develops better solutions faster. In Figure 5 we see that the GeneRepair implementation converges on the better solutions significantly faster than one without GeneRepair.

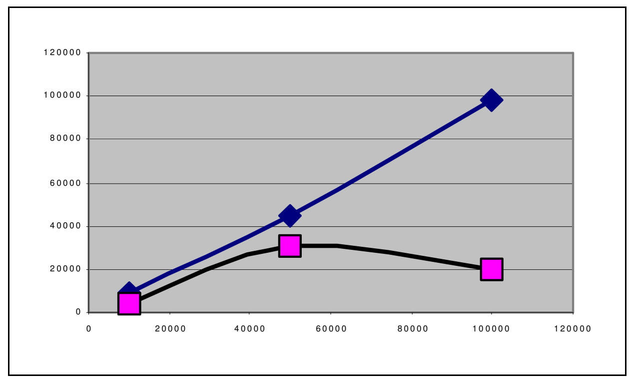

Figure 5: Solution with and without GeneRepair

Thus, GeneRepair has shown itself to be very promising and worthy of further investigation - particularly with regard to the use of only truncation selection.

5 EXPERIMENT 2 - TSP

Next we evaluate GeneRepair on the TSP benchmark problems from the Heidelberg TSPLIB problem set (Reinelt, 1991). For these experiments we investigated the potential of the GeneRepair based solution, without reference to a non-GeneRepair implementation.

We optimized the genetic parameters of crossover and mutation in order to produce the best solutions on selected TSP problems. This investigated the ability of GeneRepair to generate optimal solutions, as the benchmark solutions are assured optimal solutions.

We conducted approximately 5 experiments on each of the 3 following problem sets. Throughout all experiments the population size was the square of the number of cities in the problem set. Tournament and roulette wheel selection (but not truncation selection) were used. Only 1-point crossover was investigated. Exchange mutation was used exclusively, with rates varying between 0% and 10%.

The first problem set involved a 16-city TSP problem. Tests revealed the optimal mutation rate to be 2%. The optimal solution to this TSP problem was repeatedly found in approximately 25 generations.

The second problem involved 22 cities and again optimality was found with mutation at 2%. The optimal solution to this TSP problem was found in approximately 3200 generations.

The final problem involved 51 cities and a mutation rate of 2% was used. Only one test was complete at the time of writing with less than 10,000 generations. The shortest we produced in this test was 433 compare with the optimal solution of 429 - approximately 1% off the best solution.

6 EXPLANATION FOR GENEREPAIR

GeneRepair is composed of two distinct tasks: *fault detection* and *fault correction*. To help identify the exact reason for GeneRepairs' improvement in performance, we analyzed each phases in turn.

First we measure the frequency with which GeneRepair was invoked. GeneRepair repaired approximately 11% of the alleles, while solving the benchmark VRP problems. Additionally, some of these alleles required multiple repair operations. (As may be expected, these figures are higher during the first 100 epochs). For comparison, we recorded the number of invalid tours generated by our solution without GeneRepair. Here, approximately 15% of individuals were found to violate the VRP validity constraint.

In general, GeneRepair does increase the number of generated individuals that form part of the valid search space. However, this relatively modest increase in the search space does not adequately account for the significant increase in performance obtained. For example, increasing the population size to allow for this 11% wastage, had little effect on the quality of the results generated.

Next we investigated the *fault correction* part of GeneRepair. First, we analyze how errors are introduced. Crossover introduces the majority of errors as it is always applied. It does this by combining incompatible sections of tours. (See figure 1)

N-point-Crossover preserves the identicallity between both parents. Thus, the GeneRepair operator is invoked more during early evolution than it is when we reach convergence.

Secondly, the replacement strategy replaces invalid (i.e. duplicate) genes with missing genes, according to the replacement strategy described above. So, in conclusion, GeneRepair is a multi-point mutation operator, that is applied heavily during early evolution and rarely applied when convergence is achieved.

1-point mutation tends to introduce errors and, GeneRepair will Fix the error, but it does So randomly. Either the mutation will remain unaffected by GeneRepair and another duplicate city will be replaced. This has tie effect of causing 2-point mutation. Alternatively, the mutation itself will be repaired, which Reduces the level of mutation. Importantly, the mutation introduced by GeneRepair is Not an alternative to standard mutation, as standard mutation is still required when near-optimal convergence is reached. Initial results seem to indicate that the reduction in mutation is (at least partly) counteracted by GeneRepair's introduction of its own mutogenic effect, but investigations are ongoing.

This may account for our improved performance as it effectively prohibits the problem of premature convergence. Furthermore, it is applied less frequently during final convergence, allowing an optimal to be achieved. (This seems to mimic the operation of a Boltzman machine on simulated annealing problems.) However, investigations are at a relatively early phase, and research is ongoing.

7 FUTURE WORK

The experiments performed so far highlight the need for a number of further investigations. Future work is necessary to compare the effectiveness of GeneRepair against the order-preserving crossover and mutation operators. We will also conduct experiments to evaluate the effectiveness of GeneRepair on large problems with more than 1000 cities. Finally, we will explore the interplay between standard mutation and the mutogenic effects of GeneRepair. This may involve the use of an adaptive mutation rate in conjunction with GeneRepair.

8 CONCLUSION

We solved two permutation problems by combining standard genetic operators with a novel genetic repair operator - GeneRepair. Validity constraints that originate in the problem domain are thus centralized in a single repair operator. We explored the use of GeneRepair on the TSP and VRP, using the fitness function to optimize the solution while GeneRepair ensures the validity of solutions. This approach is potentially applicable to any domain where the solution constraints can be separated from the fitness function. Results produced so far have either reached global optimal solutions, or have been close to optimal solutions. Furthermore, solutions appear to be produced in a relatively small number of generations. We examined the higher levels of early mutation that result from GeneRepair operations, as one possible explanation for the results produced so far.

References

M.R Garey and D.S. Johnson (1979). *Computers and Intractability. A Guide to the Theory of NP-Completeness*. New York, NY: W. H Freeman and Company.

M. Mitchell (1999). *An Introduction to Genetic Algorithms*, Cambridge USA, London UK: MIT Press.

D.B. Fogel (1993), Empirical Estimation of the Computation Required to Discover Approximate Solutions to the Travelling Salesman Problem Using Evolutionary Programming, Proceedings of 2nd Annual Conference on Evolutionary Programming, 56-61.

D.B. Fogel (1993a), Applying Evolutionary Programming to Selected Travelling Salesman Problems, *Cybernetics and Systems: An International Journal*, 24 : 27-36

D.B. Fogel (1988), An Evolutionary Approach to the Travelling Salesman Problems , *Biological Cybernetics*, 60 : 139-144.

W. Banzhaf (1990), The “Molecular” Travelling Salesman, *Biological Cybernetics*, 64 : 7-14.

B.K. Ambati, J. Ambati and M.M. Mokhtar (1991), Heuristic Combinatorial Optimisation by Simulated Darwinian Evolution: a Polynomial Time Algorithm for the Traveling Salesman Problem, *Biological Cybernetics*, 65 : 31-35.

F. B. Pereira, J. Tavares, P. Machado, and E. Costa (2002), GVR: a New Genetic Representation for the Vehicle Routing Problem, Proceedings of the 13th Irish Conference on Artificial Intelligence and Cognitive Science, 95-102.

G. Syswerda (1991), Schedule Optimization Using Genetic Algorithms, *Handbook of Genetic Algorithms*, New York NY, Van Nostrand Reinhold, 350-372.

L. Davis (1985), Applying Adaptive Algorithms to Epistatic Domains, Proceedings of the International Joint Conference on Artificial Intelligence, 162-164.

D.E. Goldberg and R. Lingle (1985), Alles, Loci and the TSP, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 154-159.

I.M. Oliver, D.J. Smith and J.R.C. Holland (1987), A Study of Permutation Crossover Operators on the TSP, Genetic Algorithms and Their Applications: Proceedings of the Second International Conference, 224-230.

M. Gorges-Schleuter (1989) ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy, Proceedings of the Third International Conference on Genetic Algorithms, 422-427.

Augerat Set-A available at http://www.branchandcut.org

Crawford, K. D., R. Wainwright (1996), Research Question: How does one go about developing a new crossover operator with an a priori expectation of its merit? (A Survey of Crossover Operators for Genetic Algorithms), Technical Report UTULSA-MCS-96-2, The University of Tulsa, USA.

Z. Michalewicz (1992), Genetic Algorithms + Data Structures = Evolution Programs, Berlin Germany, Springer Verlag.

J. Holland (1975), Adaptation in Natural and Artificial Systems, Ann Arbor USA, University of Michigan.

J. Grefenstette, R. Gopal, B. Rosmaita and D. Van Gucht (1985), Genetic Algorithms for the TSP, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 160-65.

P. Larrañaga, C.M.H Kuijpers, R.H. Murga, I. Inza and S. Dizdarevic (1999), Genetic Algorithms for the Travelling Salesman Problem A Review of Representations and Operators, *Artificial Intelligence Review*, 13 : 129 – 170.

G. Reinelt. (1991), TSPLIB: A traveling salesman problem library. ORSA Journal on Computing, 3:376—384.