Quantitative Analysis of Separate and Combined Performance of Local Searcher and Genetic Algorithm

M. Djordjevic,[1] A. Brodnik[2]

[1]Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Koper, Glagoljaska 8, 6000, Koper
milan.djordjevic@student.upr.si
[2]Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Koper, Glagoljaska 8, 6000, Koper
andrej.brodnik@upr.si

Abstract. In this paper an environment is established for a quantitative analysis of separate and combined performance of local searchers and standard genetic algorithm. Well researched and controlled Euclidean Travelling Salesman Problem examines the impact of grafting a 2-opt based local searcher into the standard genetic algorithm for solving the Travelling Salesman Problem with Euclidean distance. Standard genetic algorithms are known to be rather slow, while 2-opt search applied to the Travelling Salesman Problem quickly gives results that are far from optimal. We propose a strategy to graft a 2-opt local searcher into a genetic algorithm, after recombination, to optimize each offspring's genomes. Genetic algorithm provides new search areas, while 2-opt improves convergence. We tested our algorithm on examples from TSPLIB and proved that this method combines good qualities from both methods applied, significantly outperforming each of them.

Keywords. Genetic Algorithms, Grafted Genetic Algorithm, Traveling Salesman Problem (TSP), Memetic Algorithms (MA).

1 Introduction

Genetic Algorithms (GA) use some mechanisms inspired by biological evolution [8]. They are applied on a finite set of individuals called population. Each individual in a population represents one of the feasible solutions of the search space. Mapping between genetic codes and the search space is called encoding and can be binary or over some alphabet of higher cardinality. Good choice of encoding is a basic condition for successful application of a genetic algorithm. Each individual in the population is assigned a value called fitness. Fitness represents a relative indicator of quality of an individual compared to other individuals in the population. Selection operator chooses individuals from the current population and takes the ones that are transferred to the next generation. Thereby, individuals with better fitness are more likely to survive in the population's next generation. Recombination operator combines parts of genetic code of the individuals (parents) and that process brings codes of new individuals (offspring). Such a mixing of genetic material enables that well-fitted individuals or their relatively good genes give even better offspring. By a successive application of selection and crossover, the diversity of genetic material can be decreased which leads to a premature convergence in a local optimum which may be far from a global one. The components of the genetic algorithm software system are: Genotype, Fitness function, Recombinator, Selector, Mater, Replacer, Terminator, and in our system a Local searcher which is new extended component.

The **2-opt** is a simple local search algorithm for solving the Travelling Salesman Problem. The main idea behind it is to take a route that crosses itself and reorder it so that it does not. The basic step of 2-opt is delete two edges from a tour and reconnect the remaining fragments of the tour by adding two new edges. Once we choose the two edges to delete, we do not have a choice about which edges to add – there is only one way to add new edges that results in a valid tour. The 2-opt algorithm repeatedly looks for 2-opt moves that decrease the cost of the tour. A 2-opt move decreases the cost of a tour when the sum of the lengths of the two deleted edges is greater than the sum of the lengths of the two deleted edges. A 2-opt move is the same as inverting a subsequence of cities in the tour.

Here is a pseudcode for the 2-opt local search algorithm:

```
current_tour := create_random_initial_tour()
repeat
    modified_tour := apply_2opt_move(current_tour)
    if length(modified_tour) < length(current_tour)
```

```
        then current_tour := modified_tour
until no further improvement or a specified number of iterations
```

Although the 2-opt algorithm performs well and can be applied to Traveling Salesman Problems with many cities [4], it has a serious drawback since it can quickly become stuck in a local minimum.

In the **Traveling Salesman Problem** (TSP) a set $\{C_1, C_2, \cdots C_N\}$ of cities is considered and for each pair $\{C_i,C_j\}$ of distinct cities a distance $d(C_i,C_j)$ is given. The goal is to find an ordering π of the cities that minimizes the quantity

$$\sum_{i=1}^{N-1} d(C_{\pi(i)}, C_{\pi(i+1)}) + d(C_{\pi(N)}, C_{\pi(1)}). \quad (1)$$

This quantity is referred to as the tour length since it is the length of the tour a salesman would make when visiting the cities in the order specified by the permutation, returning at the end to the initial city. We will concentrate in this paper on the symmetric TSP in which the distances satisfy $d(C\_i,C\_j)=d(C\_j,C\_i)$ for $1\leq i,j\leq N$ and more specificaly to the Euclidean distance. While the TSP is known to be NP-hard [6] even under substantial restrictions, the case with Euclidean distance is well researched and there are many excellent algorithms which perform well even on very large cases [6].

Genetic algorithms have been successfully applied to the TSP, but for restricted versions of the TSP, such as one with the Euclidean distance, they are very slow in convergence and more efficient methods are known [5]. The genetic algorithm`s considered in this paper are hybrid evolutionary algorithms incorporating local search which have been referred to as memetic algorithms (MA) [13], [14], [16], genetic local-search methods [17], Lamarckian genetic algorithms [15], Lamarckian search, and Baldwian search [12].

2 Grafted GA for the TSP

Grafted genetic algorithm. Grafting in botanic is when the tissues of one plant are affixed to the tissues of another. To speed maturity of hybrids in fruit tree breeding programs, hybrid seedlings may take ten or more years to flower and fruit on their own roots. Grafting can reduce the time to flowering and shorten the breeding program.

Local Searcher is an extension of the conventional genetic algorithm as it does not need to make use of genetic components. It facilitates the optimization of individual genomes outside the evolution process. There are many implementations of local searchers [5], [6], some even in hardware [9]. In our algorithm, after the Recombination has been applied, a Local Searcher is used to optimize every single offspring genome. Because of the usage of such external optimizer the genetic algorithm is no longer "pure" and therefore we then speak of a grafted genetic algorithm [2], [3]. This form of optimization is of a local kind. It alters the genome by heuristically changing the solution. When approximating a TSP instance, a 2-opt local optimization technique is applied to make modifications to a genome so as to create better genomes at a higher rate. These are much desired because the evolution process can be quite slow with respect to the desired results. Furthermore it has always been the case in optimization that incorporating problem specific knowledge (not only through local optimizations, but also in defining the evolutionary operators) is required to gain better results.

A genome represents a potential solution to a problem. How the solution information is coded within a genome is determined by the genotype. TSP Numbered List is a representation of a tour in the TSP by means of a list in which the locations are identified by numbers.

The fitness function (FF) has a specific task in a genetic algorithm and plays a specific role in terms of the optimization problem description. The fitness function rates the genomes and therefore the solutions according to their fitness. Solution for our TSP problem is a Hamiltonian cycle and the fitness value is the sum of the weights of the edges contained in the cycle.

Edge map crossover is an implementation of the recombination operator. It makes use of a so called edge map. Edge map is a table in which each location is placed. For each location there is a list in which the neighbouring locations are listed with this location. Recombination is then established as follows:

1. Choose the first location of one of both parents to be the current location.
2. Remove the current location from the edge map lists.
3. If the current location still has remaining edges, go to step 4, otherwise go to step 5.
4. Choose the new current location from the edge map lists of the current location as the one with the shortest edge map list.
5. If there are remaining locations, choose the one with the shortest edge map list to be the current location and return to step 2.

Example: Parents: 1-2-3-4-5-6; 2-4-3-1-5-6

Edge map: 1) 2 6 3 5; 2) 1 3 4 6; 3) 2 4 1; 4) 3 5 2; 5) 4 6 1; 6) 1 5 2 6

1. Random choice: 2.
2. Next candidates: 1 3 4 6, choose from 3 4 6 (same #edges), choose 3.
3. Next candidates: 1 4 (edge list 4 < edge list 1), choose 4.
4. Next candidate: 5, choose 5.
5. Next candidate: 1 6 (tie breaking) choose 1
6. Next candidate; 6, choose 6.

Offspring: 2-3-4-5-1-6

Distance preserving crossover is another implementation of the recombination operator. It attempts to create a new tour with the same distance to both parents. In order to establish this, the content of the first parent is copied to the offspring and all edges that do not occur in the second parent are removed. The resulting fragments are reconnected without making use of non-overlapping edges of the parents. If edge (*i, j*) has been destroyed, the nearest available neighbor *k* of *i* from the remaining fragments, is selected and the edge (*i, k)* is added to the tour [7].

Example: Parents: 5-3-9-1-2-8-0-6-7-4; 1-2-5-3-9-4-8-6-0-7

Fragments: 5-3-9|1-2|8|0-6|7|4

Offspring: 6-0-5-3-9-8-7-2-1-4

Tournament Selector places groups of genomes from the population together, creating the groups from top to bottom with respect to the enumerative ordering of the genomes in the population and selects the best of the genomes within this group. This is repeated until the required amount of genomes is selected. The selection size is 400, and tournament size is 2. The Random Mater is a simple way of mating parents. It mates the parents as enumerated in the population at random using the mating size to create groups until no more groups can be created. The grouping size is 2. The new offspring only replacer is the implementation of the classical replacement strategy that simply only allows the offspring to survive. Thus the genomes from the next generation replace the entire current population. The equality terminator for all equal genomes implements the termination condition specifying that the genetic algorithm should terminate when all genomes in the population are identical-all equal genomes.

The Local Searcher is an extension to the conventional genetic algorithm as it need not make use of genetic operators. It facilitates the optimization of individual genomes outside the evolution process. After the Recombination has been applied, a Local Searcher is used to optimize every single offspring genome. The Local Searcher has no further knowledge on the execution of the genetic algorithm in the larger setting. The system will provide it with the genome it needs to locally optimize when needed. Fig. 1 presents the pseudo code for the algorithm.

```
t=0
initialize(P(t))
evaluate(P(t))
while(not terminate(P(t))) do
   sel=select(P(t))
   mat=mate(sel)
   rec=for each mated collection m∈mat do
recombination(r)
   loc=for each genome g in each recombined collection
r∈rec do local search(l)
   rep=replace(loc, P(t))
   P(t+1)=select(rep)
   evaluate(P(t+1))
   t=t+1
```

Figure 1 Algorithm Code

The 2-opt Hybrid searcher is a local optimizer for the TSP that has been grafted into the standard genetic algorithm. This local optimizer performs the 2-opt heuristic that exchanges edges to reduce the length of a tour. An exchange step consists of removing two edges from the current tour and reconnecting the resulting two paths in the best possible way. (Fig. 2)

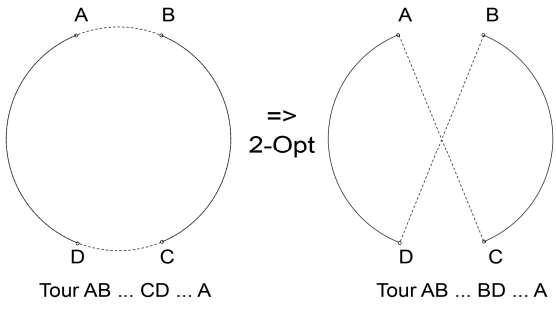

Figure 2 Exchange step of 2-opt algorithm

3.1 Experiment

For testing our strategy and comparing it to other solutions we used the instances of symmetric traveling salesman problem which can be found on TSPLIB. We deliberately used well known problem (TSP) and relatively small instances for which best solutions are known since the goal of this research is not to find a better algorithm for the symmetric TSP, but rather to compare on well controlled environment the impact of grafting a genetic algorithm. Altogether 20 instances have been tried out, with different complexity and range from 14 to 150 cities per instance.

We compared our method (grafted genetic algorithm – GGA), separately in one case with edge map crossover (GGAemc) and in another case with a distance preserving crossover (GGAdpc) with four other methods. As the upper bound for the quality of solution we used the above mentioned Greedy Heuristic. For the lower bound for the quality of solution we used exact solutions, global minima, obtained by Concorde. Then we compared our grafted method with a pure 2-opt algorithm and pure genetic algorithm.

For Greedy Heuristic and the pure 2-opt Heuristic the running time is in a range from 0.5 to 1.5 seconds. All tests were conducted on a laptop computer with AMD 2GHz processor, with Windows 7. In this research absolute times were not of crucial importance, we were only interested in relative performance of tested algorithms.

3.2 Results

All the results are summarized in Table 1. Twenty well known cases from TSPLIB were used for testing. The names of these cases are in the first column and the name always contains the size of the problem, i.e. the number of cities (which are between 14 and 150).

The last two columns are exact solutions (global minima) obtained by Concorde, together with execution times. A well known problem with moderate sized examples and tools to get optimal solutions were selected, recall that a goal of this research is not to improve solutions for difficult problems but to compare and quantitatively examine the effects of grafting local searches (in this case 2-opt based) to standard genetic algorithm. Such knowledge can be used to fine tune and calibrate a hybrid system which can then be used on large cases. These last two columns are used as a reference for all other tests.

The second column in Table 1 represents lower bound for the quality of solution. It is a simple Greedy Heuristic described in Section 1. It is fast, but very unsophisticated and any reasonable algorithm should do better than that. The Greedy Heuristic gives results that are about 15% (except for some very small cases) worse than the optimal solution. The column titled *quality* shows by how many percent is the solution produced by this algorithm worse than the optimal solution. 0% in this column means that the algorithm found the best solution. The running times of the algorithm are from 0.2 to 2.3 in seconds.

The third column in the Table 1 corresponds to the pure 2-opt algorithm. As expected, it also gives quick but far from optimal solutions. It quickly finds a local minimum, but it is unable to broaden the search to find another local minimum. However, 2-opt algorithm is superior to Greedy algorithm, the quality of the solution, with the similar running times from 0.2 to 2.5 seconds, is on average about 8% worse than optimal.

The fourth column in the Table 1 corresponds to the pure Genetic Algorithm. The running time, as expected, is significantly increased. While our GGA algorithm reached optimal solution below one second or few seconds (0.6 to 17.1 seconds), the running time for pure genetic algorithm was from 3.4 seconds to 100 seconds which was time-limit. In 12 out of 20 cases no optimal solution was found within that time limit, but in 8 cases an optimal solution was found and the middle column indicates in which generation that happened. For 12 cases where optimal solution was not found, the quality of found solution is expressed as for previous cases in percents above the optimal solution.

The fifth column in Table 1 describes results obtained by our grafted algorithm, which is programmed with edge map crossover as recombination operator (GGAemc). In 17 out of 20 considered cases an optimal solution was found. Remaining three instances differ from optimal solution in 0.01, 0.18 and 0.22 percent. The solutions were found in relatively few generations and very fast. Execution times were 0.6 to 17.1 seconds.

The sixth column in Table 1 corresponds to our grafted genetic algorithm which contains a distance preserving crossover as recombination operator (GGAdpc). In 11 out of 20 considered cases an optimal solution was found. In remained 9 cases, delivered solutions differ from optimal in range from 0.13 to 0.32 percent. The running time and number of generations of GGAdpc, in comparison with GGAemc, are slightly lesser, particularly in the lowermost part of the table which represents more complex instances. The difference in the quality on the other side is in the hand of GGAemc for the same considered cases.

Table 1. Five techniques for solving Euclidean TSP

Name	**Greedy**	**2-opt**	**GAemc**			**GAdpc**			**GGAemc**			**GGAdpc**			**Concorde**	
	quality	*quality*	*quality*	*gen.*	*time*	*quality*	*gen.*	*time*	*qual.*	*gen.*	*time*	*qual.*	*gen.*	*time*	*opt*	*time*
burma14	8.32%	5.71%	0%	74	3.4	0%	81	3.5	0%	7	0.6	0%	6	0.5	3323	0.1
ulysses16	10.42%	7.15%	0%	136	4.1	0%	125	4.4	0%	9	0.7	0%	9	0.7	6859	0.2
ulysses22	12.54%	7.87%	0%	1267	14.7	0%	1328	16.4	0%	8	0.6	0%	8	0.7	7013	0.2
bayg29	13.37%	6.38%	0%	1345	19.4	0%	1137	17.6	0%	13	1.3	0%	14	1.4	1610	0.3
bays29	12.87%	5.37%	0%	2185	29.2	0%	2643	34.1	0%	12	1.2	0%	12	1.2	2020	0.3
dantzig42	14.06%	7.11%	0%	4704	79.8	0%	4232	74.6	0%	10	1.3	0%	9	1.3	699	0.5
att48	13.98%	8.47%	0%	4807	85.2	0%	5213	91.3	0%	22	2.2	0%	23	2.3	33522	0.6
eil51	15.24%	7.67%	4.21%	5482	100.0+	5.23%	5489	100.0+	0%	33	6	0%	30	6.1	426	0.3
berlin52	14.82%	7.45%	0%	2037	33.7	4.92%	5021	100.0+	0%	15	1.7	0%	15	1.7	7542	0.4
st70	13.17%	7.84%	5.12%	5259	100.0+	5.72%	5198	100.0+	0%	20	5.1	0%	19	5.1	675	0.5
eil76	14.47%	8.15%	6.56%	5347	100.0+	7.24%	5298	100.0+	0%	53	9.1	0.19%	49	9.1	538	1.3
pr76	13.96%	9.95%	4.18%	5218	100.0+	5.36%	5191	100.0+	0%	42	7.4	0%	43	7.4	108159	1.2
gr96	16.32%	7.14%	4.98%	5191	100.0+	5.71%	5090	100.0+	0%	73	12	0.13%	73	12	55209	1.6
rat99	14.79%	7.41%	5.31%	5114	100.0+	7.12%	5011	100.0+	0%	74	13	0.17%	70	13	1211	1.7
kroA100	12.37%	8.07%	5.12%	5072	100.0+	6.58%	4971	100.0+	0%	24	12	0.18%	22	12	21282	1.7
kroB100	16.58%	7.19%	6.14%	5041	100.0+	5.92%	4816	100.0+	0%	39	13	0.21%	36	13	22141	1.7
kroC100	10.47%	11.19%	4.87%	5121	100.0+	6.78%	4923	100.0+	0.18%	34	13	0.19%	28	13	20749	1.8
kroD100	14.81%	7.74%	5.07%	4976	100.0+	8.12%	4951	100.0+	0%	31	13	0.29%	25	13	21294	1.5
lin105	16.60%	9.85%	6.72%	4756	100.0+	6.51%	4803	100.0+	0.01%	26	9.9	0.17%	25	9.7	14379	1.3
ch150	19.62%	11.72%	7.22%	4512	100.0+	8.77%	4460	100.0+	0.22%	88	17	0.32%	82	17	6528	7

1.5 Conclusions

The goal of this paper was to investigate the impact of grafting a 2-opt based local searcher into the standard genetic algorithm, GGAemc and GGAdpc, for solving the Travelling Salesman Problem with Euclidean distance. It is known that genetic algorithms are very successful when implemented for many NP-hard problems. However, they are much more effective if some specific knowledge about particular problem is utilized. The TSP is well researched problem with many such improvements, especially when the restricted version of the problem with Euclidean distance is considered. In that controlled environment we compared two direct techniques, a genetic algorithm and a 2-opt algorithm with our grafted genetic algorithms. Exact solution from Concorde and lower bound on quality, Greedy algorithm were added for better comparison. Quantitative results on test cases from TSPLIB show that grafted algorithms have new quality. Even when both components have serious drawbacks, their grafted combinations exhibits excellent behaviour. Further calibration of this system will include measuring the optimal blend of two components for larger test cases. The future research will focus on a new heuristic algorithm for making an initial tour of Lin-Kernighan heuristic, which is known to be one of the most successful heuristics for the TSP.

References

1. D. Applegate, R. Bixby, V. Chvatal, and W. Cook: *Finding tours in the TSP. Technical Report 99885* (Research Institute for Discrete Mathematics, University of Bonn, 1999).
2. M. Djordjevic: Influence of grafting a hybrid searcher into the evolutionary algorithm. In B. Zajc and A. Trost (eds.): *Proceedings of International Electrotechnical and Computer Science Conference*, **17** (IEEE, Ljubljana, 2008), 115-118.
3. M. Djordjevic: Impact of grafting a 2-opt algorithm based local searcher into the genetic algorithm. In N. E. Mastorakis, M. Demiralp, V. Mladenov and Z. Bojkovic (eds.) *: Recent Advances In Applied Informatics And Communications* (Wseas press, Moscow, 2009), 485-490.
4. C. Engels, B. Manthey: Average-case approximation ratio of the 2-opt algorithm for the TSP. *Operations Research Letters*, **37** (2009), 83-84.

5. B. Freisleben, P. Merz: New Genetic Local Search Operators for the Traveling Salesman Problem. In H. Voigt, W. Ebeling, I. Rechenberg and H. Schwefel (eds.): *Parallel Solving from Nature – PPSN IV* (Springer-Verlag, Berlin, 1996), 890-899.
6 M. R. Garey and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, New York, 1979).
7. K. Helsgaun: An effective implementation of the Lin-Kernighan travel salesman heuristic. In R. Slowinski, J. Artalejo, J. Billaut, R. Dyson and L. Peccati (eds.): *European Journal of Operational Research*, **126** (Elsevier, Amsterdam, 2000), 106- 130.
8. J. H. Holland: *Adaptation in natural and artificial system (*MIT Press, Cam bridge, 1992).
9. H. Hoos and T. Stützle: *Stochastic Local Search, Chapter 8 (TSP)* (Morgan Kaufmann Publishers, San Francisko, 2004).
10. D. S. Johnson, L. A. McGeoch: The travelling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra (eds.): *Local Search in Combinatorial Optimization* (John Wiley & Sons, New York, 1997), 215–310.
11. D. S. Johnson, L. A. McGeoch: Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen (eds.): *The Traveling Salesman Problem And Its Variations* (Kluwer Academic Publishers, Boston, 2002), 369–443.
12. B. Julstrom: Comparing Darwian, Baldwian, and Lamarckian search in a genetic algorithm for the 4-cycle problem. . In L. D. Whitley and D. E. Golberg (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference*, (Morgan Kaufmann, Orlando, 1999), 134-138.
13. N. Krasnogor and J.Smith: A memetic algorithm with self-adaptive local search: TSP as a case study. In L. D. Whitley and D. E. Golberg (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference* (Morgan Kaufmann, Vegas, 2000), 987-994.
14. P. Merz and B. Freisleben: Memetic algorithms for the traveling salesman problem. In S. Wolfram and T. Rowland (eds.): *Complex Systems* **13**, (Complex Systems Publications, Champaigne, 2001), 297–345.
15. G. M. Morris, D. S Goodsell, R. S. Halliday, W. E. Hart, R. K. Belew, and A. J. Olson (1998) Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. In C. L. Brooks III, G. Frenking, S. Sakaki and P. R. Schreiner (eds.): *Journal of Computational Chemistry*, **19** (Wiley, New York, 1998), 1639-1662.
16. P. Moscato: *On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms* (California Institute of Technology, Pasadena, 1989).
17. T. Yamada and C. Reeves: Solving the C_{sum} permutation flowshop scheduling problem by genetic local search. *International Conference on Evolutionary Computation-WCCI98*, (1998), 230-234.