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

Milan Djordjevic and Andrej Brodnik

Abstract The paper analyzes separate and combined performance of local searchers and standard genetic algorithm. On the well studied Euclidean *Travelling Salesman Problem* we examine the impact of grafting a *2-opt* heuristic based local searcher into the standard genetic algorithm for solving the Euclidean Travelling Salesman Problem. Genetic algorithm provides a diversification, while 2-opt improves intensification. Results on examples from *TSPLIB* show that this method combines good qualities from both methods applied and significantly outperforms each individual method.

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 (offsprings). The components of the genetic algorithm

Milan Djordjevic
UP FAMNIT, Koper, Slovenia e-mail: milan.djordjevic@student.upr.si

Andrej Brodnik
UP FAMNIT, Koper, Slovenia e-mail: andrej.brodnik@upr.si

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. 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 $\{C_1, C_2, ...C_N\}$ 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) [10].

2 Grafted GA for the TSP

Grafting in botanic is when the tissues of one plant are affixed to the tissues of another. 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], see Algorithm 1. This form of optimization is of a local kind. It alters the genome by heuristically changing the solution. Edge map crossover [5] is an implementation of the recombination operator. It makes use of a so called edge map. Distance preserving crossover is another implementation of the recombination operator. It attempts

to create a new tour with the same distance to both parents [7]. 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. 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.

Algorithm 1 Pseudocode

```
1:  t = 0
2:  initialize (P(t))
3:  evaluate (P(t))
4:  while not terminate (P(t)) do
5:      sel = select (P(t))
6:      mat = mate (sel)
7:      rec = for each mated collection m ∈ mat do recombination(r)
8:      loc = for each genome g in each recombined collection r ∈ rec do local search
9:      rep = replace(loc, P(t))
10:     P(t+1) = select(rep)
11:     evaluate (P(t+1))
12:     t = t+1
```

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. 1)

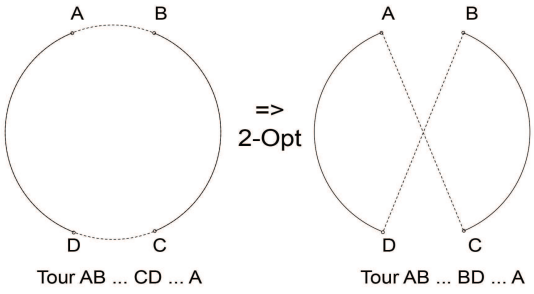

Fig. 1 Exchange step of 2-opt algorithm

3 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.

4 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 [1], 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 [9]. 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 sixth 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 seventh 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 0.32percent. 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.

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. In our experiment we compared two direct techniques, a genetic algorithm and a 2-opt heuristic with our grafted genetic algorithms. Solutions from Concorde and greedy algorithm were added for better comparison. Quantitative results on test cases from *TSPLIB* show that grafted algorithms have new advantages. Even when both components have serious drawbacks, their grafted combinations exhibits a very good behaviour. Results on examples from *TSPLIB* show that this method combines good qualities from both methods applied and significantly outperforms each individual method. Further calibration of this system will include measuring the optimal blend of two components for larger test cases.

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

Table 1 Five techniques for solving Euclidean TSP

References

1. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: TSP cuts which do not conform to the template paradigm. In: Computational Combinatorial Optimization. pp. 261–304 (2001),
2. Djordjevic, M.: Influence of Grafting a Hybrid Searcher Into the Evolutionary Algorithm. In: Proceedings of the Seventeenth International Electrotechnical and Computer Science Conference. pp. 115–118. Slovenian Section IEEE (2008)
3. Djordjevic, M., Tuba, M., Djordjevic, B.: Impact of Grafting a 2-opt Algorithm Based Local Searcher Into the Genetic Algorithm. In: Proceedings of the 9th WSEAS international conference on Applied informatics and communications. pp. 485–490. World Scientific and Engineering Academy and Society (WSEAS) (2009)
4. Engels, C., Manthey, B.: Average-case approximation ratio of the 2-opt algorithm for the tsp. Operations Research Letters **37**, 83–84 (2009)
5. Freisleben, B., Merz, P.: New genetic local search operators for the traveling salesman problem. In: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature. pp. 890–899. PPSN IV (1996),
6. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman & Co, New York (1979)
7. Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research **126**, 106–130 (2000)
8. Holland, J.: Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor (1975)
9. Hoos, H., Stutzle, T.: Stochastic local search: Foundations and applications. Morgan Kaufmann (2005)
10. Merz, P., Freisleben, B.: Memetic algorithms for the traveling salesman problem. Complex Systems **13**, 297–346 (2001)