Impact of Grafting a 2-opt Algorithm Based Local Searcher into the Genetic Algorithm

MILAN DJORDJEVIC
FAMNIT/PINT
University of Primorska Koper
Glagoljaska 8, Koper
SLOVENIA
milan.djordjevic@student.upr.si

Abstract: - This paper 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. Pure 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 genetic algorithm, after recombination and mutation, 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 applied methods, significantly over performing each of them.

Key-Words: - Genetic Algorithms (GA), Grafted Genetic Algorithm (GGA), Travelling Salesman Problem (TSP), Memetic algorithms (MA)**.**

1 Introduction

Genetic Algorithms (GA) use some mechanisms inspired by biological evolution [1]. They are applied on a finite set of individuals called population. Each individual in population represents one of the feasible solutions of the search space. Function 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 genetic algorithm. Each individual in the population is assigned with the 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. Crossover operator combines parts of genetic code of the individuals (parents) and that process brings codes of new individuals (offspring). Such mix of genetic material enables that well-fitted individuals or their relatively good genes give even better offspring. By successive application of selection and crossover, the diversity of genetic material can be decreased which leads to premature convergence in local optimum which may be far from the global one. Mutation operator contributes to restoring lost regions of the search space by random changing of some genes. 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.

In the **traveling salesman problem (TSP)** a set {C1, C2, ⋯ CN) of cities is considered and for each pair {Ci, Cj} of distinct cities a distance dCj, Ci is given. The goal is to find an ordering π of the cities that minimizes the quantity

$$i=1N-1d(C\pi i,\quad C\pi i+1)+dC\pi N\ ,\ C\pi 1. \qquad (1)$$

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 dCi, Cj =dCj, Ci for 1≤i, j≤N and more specificaly to the Euclidean distance. While the TSP is known to be NP-hard, even when many restrictions are applied, the case with Euclidean distance is well researched and there are many excellent algorithms which perform well even on very large cases.

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 [2].

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 over itself and reorder it so that it does not. The basic step of 2-opt is to 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 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 [3], it has a serious drawback since it can quickly become stuck in local minima.

The Greedy Heuristic gradually constructs a tour by repeatedly selecting the shortest edge and adding it to the tour as long as it doesn't create a cycle with less than *N* edges, or increases the degree of any node to more than 2. The same edge must not be added twice. The running time for this heuristic is O $(n^2 \log^2(n))$ and the pseudo code is:

1. Sort all edges.
2. Select the shortest edge and add it to our tour if it doesn't violate any of the above constraints.
3. Do we have N edges in our tour? If not, repeat step 2.

The Greedy algorithm normally keeps within 15-20% of the Held-Karp lower bound.

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 to the conventional genetic algorithm as it need not make use of genetic components. It facilitates the optimization of individual genomes outside the evolution process. There are many implementations of local searchers [4], [5], some even in hardware [6]. In our algorithm, after both the Recombination and the Mutation have 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. 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. This are very needed 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.

2 Grafted GA for TSP

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. The fitness values are then rated better when they are smaller. The fitness function defines a mapping from the solution space to the real number and it plays a role of the environment in the optimization problem and holds information on the coordinates of the locations or the distances between them. For each location it`s coordinates are stored within this FF, these locations are in the two dimensional space of real numbers.

Edge map recombination makes use of a so called edge map. This edge map is a table in which each location is placed. For each location there is a list in which the neighbouring location are listed if this location within the two parents. 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, chooser 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

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 (the winner of the tournament). This is repeated until the required amount of genomes is selected. Parameters programmed: Selection size – 400, tournament size – 2. The Random Mater is like 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 random behaviour prohibits the creation of the same groups of parents over and over again. Parameters programmed: Grouping size – 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. This is the replacement strategy that will suffice for most genetic algorithms. The equality terminator four 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 both the Recombination and the Mutation have 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 localsearch(I)
  loc=for each genome g in each recombined collection
          r∈re do local search(I)
  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 well known 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)

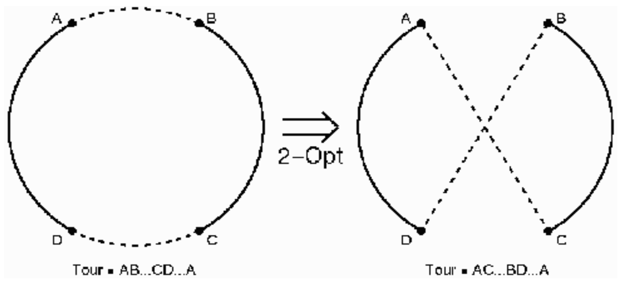

Figure 2 Exchange step of 2-opt algorithm

The following figure (Fig. 3) gives an example of an application of the 2-opt heuristic to a two dimensional geometric TSP:

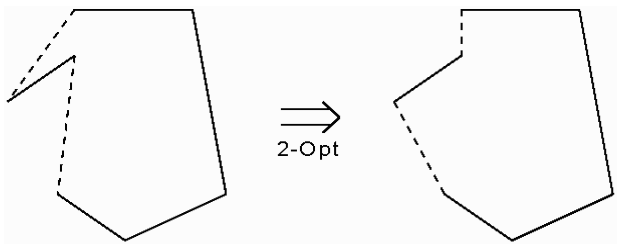

Figure 3 Example of an application of the 2opt to the TSP

3 Tests

For testing our strategy and comparing it to other solutions we used all the instances of symmetric traveling salesman problem which can be found on TSPLIB [7] that have less than 100 nodes. 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 the better algorithm for the symmetric TSP, but rather to compare on well controlled environment impact of grafting genetic algorithm. Altogether 14 instances have been tried out, with different complexity and range from 14 to 99 cities per instance.

We compared our method (grafted genetic algorithm – GGA) with four other methods. As the lower bound for the quality of solution we used above mentioned Greedy Heuristic. For the upper bound for the quality of solution we used exact solutions, global minima, obtained by Concorde. Then we compared our grafted method with pure 2-opt algorithm and pure genetic algorithm (GA).

For Greedy Heuristic and pure 2-opt Heuristic we limited running time to the time needed by our GGA to reach optimal solution. All tests were conducted on laptop computer with AMD 2GHz processor. In this research absolute times were not important, we were only intersted in relative performance of tested algorithms.

4 Results

All the results are summarized in Table 1. As mentioned before, 14 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 from 14 to 99).

The last two columns are exact solutions (global minima) obtained by Concorde, together with execution times. Well known problem with moderate sized examples and tools to get optimal solutions were selected since the 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 hybrid system that 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 bond 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 12% (except for some very small cases) worse than the optimal solution. The column titled *quality* shows by how many percent is the solution for this algorithm worse than the optimal solution. 0% in this column means that the algorithm found the best solution.

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 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 same running time, is on average about 6% worse than optimal.

The fourth column in the Table 1 corresponds to pure Genetic Algorithm. The running time, as expected, is significantly increased. While our GGA algorithm reached optimal solution (the same time was allowed to previous two quick algorithms) below one second or few seconds (0.6 to 13 seconds), the running time for pure genetic algorithm was from 3.4 seconds to 100 seconds which was a time-limit. In 6 out of 14 cases the optimal solution was not found within that time limit, but in 8 cases optimal solution was found and middle column indicates in which generation that happened. For 6 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 most important is the fifth column (in red color). It describes results obtained by our grafted algorithm. In all 14 considered cases optimal solution was found. It was found in relatively few generations and very fast. Execution time was 0.6 to 13 seconds. It shows that grafting introduces new quality and gives results much better than any of its components.

Name	Greedy	2-opt	GA			GGA			Concorde	
	quality	*quality*	*quality*	*gen.*	*time*	*qual.*	*gen.*	*time*	*opt*	*time*
burma14	7.12%	3.14%	0%	74	3.4	0%	7	0.6	3323	0.1
ulysses16	8.12%	6.45%	0%	136	4.1	0%	9	0.7	6859	0.19
ulysses22	10.34%	6.57%	0%	1267	14.7	0%	8	0.6	7013	0.24
bayg29	11.57%	4.24%	0%	1345	19.4	0%	13	1.3	1610	0.25
bays29	11.42%	3.41%	0%	2185	29.2	0%	12	1.2	2020	0.31
dantzig42	12.87%	6.75%	0%	4704	79.8	0%	10	1.3	699	0.52
att48	12.34%	7.44%	0%	4807	85.2	0%	22	2.2	10628	0.45
eil51	12.87%	6.95%	4.21%	5482	100.0+	0%	33	6	426	0.31
berlin52	13.12%	6.92%	0%	2037	33.7	0%	15	1.7	7542	0.41
st70	11.32%	6.91%	5.12%	5259	100.0+	0%	20	5.1	675	0.47
eil76	13.80%	7.15%	6.56%	5347	100.0+	0%	53	9.1	538	1.27
pr76	11.87%	6.95%	4.18%	5218	100.0+	0%	42	7.4	108159	1.17
gr96	12.92%	5.10%	4.98%	5191	100.0+	0%	73	12	55209	1.62
rat99	12.49%	6.11%	5.31%	5114	100.0+	0%	74	13	1211	1.68

Table 1: Five techniques for solving Euclidean TSP

5 Conclusion

The goal of this paper was to investigate the impact of grafting a 2-opt based local searcher into the standard genetic algorithm 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 restricted version of the problem with Euclidean distance is considered. In that controlled environment we compared two pure techniques, genetic algorithm and 2-opt algorithm with our grafted genetic algorithm. 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 algorithm has new quality. Even when both components have serious drawbacks, their grafted combination exhibits excellent behaviour. Further calibration of this system will include measuring the optimal blend of two components for larger test cases.

References:

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems* , University of Michigan Press Publishing House, 1975.

[2] Concorde TSP Solver softvare, William Cook http://www.tsp.gatech.edu/concorde.html

[3] Engels C, Manthey B: Average-case approximation ratio of the 2-opt algorithm for the TSP, *Operations Research Letters,* Volume 37, Issue 2, 2009, pp. 83-84

[4] Weiqi Li: Attractor of Local Search Space in the Travelling Salesman, *WSEAS Transactions on Systems*, Issue 3, Vol. 3, May 2004, pp 1114-1119

[5] Jian L, Peng C, Zhiming L: Solving Traveling Salesman Problems by Genetic Differential Evolution with Local Search, *Workshop On Power Electronics And Intelligent*

Transportation System Proceedings, 2008, pp. 454-457

[6] TSPLIB library of sample instances for the TSP http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html