Parallel Search Strategies for TSPs using a Greedy Genetic Algorithm

Yingzi Wei[1] and Kanfeng Gu[2]
[1]*School of Information Science and Engineering, Shenyang Ligong University,*
[2]*Shenyang Institute of Automation, Chinese Academy of Science,*
China

1. Introduction

The Genetic Algorithm (GA) is an optimizing algorithm modelled after the evolution of natural organisms. GA was not originally intended for highly constrained optimization problems but were soon adapted to order-based problems like the TSP (Goldberg, D.E. etc. 1985, 1989). It has also been applied to a variety of combinatorial optimization problems. GA is an iterative procedure which maintains a population of candidate solutions. These solutions (instances or chromosomes) are encoded into strings of symbols. The initial population of instances, represented by their chromosomes, can be chosen heuristically or at random. During each iteration step, called a generation, a number of individuals selected from population solutions implement genetic operations. Some of the GA's merits are that it can be easily developed. GA does not require detailed knowledge about the problem, can search globally, and also adapt to the changing conditions in the problem. The traveling salesman problem (TSP) is defined as a very difficult task that seeks a shortest tour of N cities in such a way, that to visit all cities only once and return to the starting city. The TSP was chosen for many reasons: (i) it can be used to model many practical problems, (ii) it is a standard test-bed for new algorithmic ideas and a good performance on the TSP is often taken as a proof of their usefulness or effectiveness, and (iii) it is easily understandable, so that the algorithm behavior is not obscured by too many technicalities.
Despite of these merits, GA is often slower than conventional methods, such as heuristic searches. This is because GA does not utilize explicitly the knowledge of how to search for the solutions. Therefore, hybrid methods that combine GA with other techniques have been attempted (G. Andal Jayalakshmi etc, 2001). The TSP solver we suggested is one of the hybrid methods. It combines GA and greedy principles to construct the TSP solver. With the TSP, we can study the effect of using information about distances of the cities in genetic operators. We improved the genetic operator to guide the generation of new offspring genotypes. Owing to heuristics of greed, it is much faster than other TSP solvers based on GA alone.
This paper begins with a brief description of TSP and GA in general, followed by a review of key to design the GA for permutation problems and analysis of the probable difficulties therein. Then, the greedy selection principle is introduced. In the next a few sections, we present the greedy genetic algorithm (GGA), how we modify a genetic algorithm to solve TSP, our methodology, results, and conclusions.

Source: Advances in Greedy Algorithms, Book edited by: Witold Bednorz,
ISBN 978-953-7619-27-5, pp. 586, November 2008, I-Tech, Vienna, Austria

2. Population initialization

2.1 Encoding scheme

We use a path representation where the cities are listed in the order in which they are visited. In this technique, the N cities are represented by a permutation of the integers from 1 to N. For example, assuming there are 5 cities 1, 2, 3, 4 and 5, if a salesman goes from city 4, through city 1, city 2, city 5, city 3 and returns back to city 4, the chromosome will be {4 1 2 5 3}. For an *N* cities TSP, we initialize the population by randomly placing *1* to *N* into *N* length chromosomes and guaranteeing that each city appears exactly once. Thus chromosomes stand for legal tours.
When using the GA to solve TSPs, the absolute position of a city in a string is less important than the relative position of a city with respect to a tour. So the important information in a chromosome or city sequence is the relative positions of the cities, not the absolute position. Changing the relative positions of the cities may increase or decrease the amount of building blocks and thus result in greater or lesser fitness. For example, for a 5 cities tour, {4 1 2 5 3} and {3 4 1 2 5} mean the same tour. However, pairs of cities are now important. Shortly, highly fit subsets of strings (building blocks) play an important role in the action of genetic algorithms because they combine to form better strings (Goldberg, D.E. etc. 1985, 1989).

2.2 Initial population generation from gene bank

The initial solution plays a critical role in determining the quality of final solution in any local search. However, since the initial population has been produced randomly in most GA researches, it not only requires longer search time to obtain an optimal solution but also decreases the search possibility for an optimal solution. Evolution burden on the GA is especially obvious for TSP when GA starting from an original population with poor quality. For overcoming the difficulties forementioned, we use a gene bank to generate the initial population with good and diverse individuals in this paper.
The N cities are permuted and assembled to build a gene bank. For a TSP of *N* cities, *C* cities that are closer to the city *i* are encoded to construct a gene bank, where *C* is a number less than *N-1*. For simplification, *C* equals 3 in GGA. Gene bank is a matrix $A_{N \times C}$ whose size is $N \times C$. The element of A[*i*][*j*] is the jth closest city to city *i*. For example, A[*i*][*1*] and A[*i*][*2*] are the first and second cities closest to city *i*, respectively. The *C* closest cities constitute the whole ith row of gene bank for the city *i*.
When initializing the population, the first city code *i* is generated randomly. From the *i*th row of gene bank, city code *j* is then generated where *j* is the closest one in the unselected elements of the ith row. Then, city code *h* is selected from the *j*th row of gene bank. If all the city codes of the *j*th row have been selected, GGA produce randomly a city code not traveled before as the next traveling city. Following this method, city codes not traveled are generated to form a complete chromosome. The algorithm repeats the forgoing procedures multiple times. Many such chromosomes form the initial population of GGA.
Our algorithm always makes the choice that looks best when selecting a gene to assemble a chromosome based on the gene bank. This strategy for generating initial population is of a greedy method. The substring assembled based on gene bank is of above-average fitness and short defining length. These schemata with above-average fitness, low-order and short defining length tend to produce more offspring than others. For brevity, such schemata are called building blocks. As we known, building block hypothesis is that a genetic algorithm creates stepwise better solutions by recombining, crossing and mutating short, high-fitness

schemata(Goldberg, D.E. etc. 1985, 1989). So using these substrings is of great benefit to GGA getting an effective solver.

3. Operators of greedy genetic algorithm

A simple class of GAs always guides the algorithm to the solution by preferring individuals with high fitness over low-fitted ones. It can be deterministic, but in most implementation that it has random components. Greedy algorithms are introduced to our genetic operations. After genetic operation, such as crossover and mutation, only the better offspring will replace the parents. This policy is mainly to maintain its respective evolution direction of an individual and deduce the error of random operations.

3.1 Double-directional greedy crossover

Different crossover acts like the different environmental condition impacting on an individual. A different crossover operation changes the domain and procedure of search in order to enhance the possibility of finding a new solution. We adopt multiple crossover operators in this algorithm.

Crossover is a very powerful tool for introducing new genetic material and maintaining genetic diversity, but with the outstanding property that good parents also produce well-performing children or even better ones. Traditionally, combination has been viewed as the primary mechanism and advantage of crossover. However, there is no guarantee that crossover combines the correct schemata.

For crossover operation after several tests and researching, we use the double-directional greedy crossover which is similar to the greedy crossover invented. Greedy crossover selects the first city of one parent, compares the cities leaving that city in both parents, and chooses the closer one to extend the tour (Grefenstette 1985). If one city has already appeared in the tour, we choose the other city. If both cities have already appeared, we randomly select a non-selected city. Greedy crossover guides the searching direction by using local information. The TSP, we chose, is symmetric and its tour is a Hamiltonian cycle. So we propose an effective strategy to improve the greedy crossover operation aforementioned. The gene crossing of a double-directional greedy crossover is applied twice to a chromosome (e.g. to select from the first gene to the last and from the last gene to the first, respectively). This double-directional greedy crossover provides equivalent chances for gene segments located in different positions to reach a local optimum. The method is developed to form a suboptimal cycle based on more effective local searches.

3.2 Greedy mutation

In a GA, the mutation is the random deformation of one or more genes that occurs infrequently during the evolutionary process. The purpose of the mutation is to provide a mechanism to increase coverage of the search space and help prevent premature convergence into a local optimum. Given a permutation based individual of TSP, the mutation operator modifies the related traveling sequence. There are a lot of manners for doing sequence swapping operation. Easiest way is in using random swap. Unfortunately, such strategy unable to achieve an optimum quickly but can prevent convergence into a local optimum.

We use a new mutation operator, greedy-swap of two cities positions. The basic idea of greedy-swap is to randomly select two adjacent cities from one chromosome and swap them if the new (swapped) tour length is shorter than the elder. For the use of the gene bank when initializing the population, the neighboring coding is often constituted of building block. This strategy is mainly to decrease the possibility of breaking the building block. GGA keep the new tour only when getting a shorter-length tour after not more than 3 trials of swap. So the greedy mutation operation is a procedure of local adjustment and improvement for the chromosome.

3.3 Immigration

The "goodness" of the genetic population depends both on the average fitness (that is corresponding to the objective function value) of individuals and the diversity in the population. Losing on either count tends to produce a poor GA. In the beginning, the potentially good individuals sometimes fill the population so fast that can lead to premature convergence into local maxima. Mutation means to increase diversity in the population by introducing random variations in the members of the population. However, the mutation in the end phase can be too slow to improve population since the individuals have similar fitness values. These problems can be overcome by using the immigration in place of mutation.
Immigration refers to the process of replacing poor members of the current population by bringing in new individuals. For our implementation of the immigration, the population is doped with immigrant individuals for a few of generations. After the midterm phase of evolution, we use the same method to generate immigrants as the method we adopt to generate the initial population. We found that these immigrants not only introduce new genetic material into the population but also bring an open competition plaza for GGA and hence force the algorithm to search newer regions of solution space. Immigrants can also remedy the shortage of small population because the population size is limited for too heavy computation. Figure 1 illustrates the transitional process between consecutive generations of GGA.

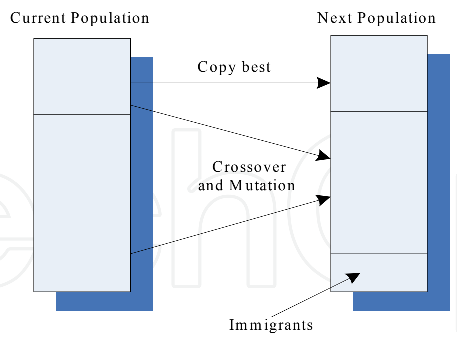

Fig. 1. Transitional process between consecutive generations

4. Evolutionary dynamics of GGA

Genetic algorithms mimic nature evolution using the principles of survival of the fittest. Reproduction operation, or called selection, is the impulsion of GA evolution. A simple GA selects the better individuals from the population into the next generation based on the roulette wheel selection. This always affects the diversities of population for that super-individual(s) will take over most of the population in a few generations.

By comparing the qualities of parents to their offspring's, the individuals of GGA realize the population evolution. Only better offspring will replace its parent place. We abandon the traditional selection operator in our GGA so that the population's diversity is kept very well all along. Each individual runs in its own evolution direction, respectively. Different individuals search different domains. The greedy genetic algorithm takes on the parallelization nature due to its parallel searches.

We haven't employed special fitness function for TSP problems. The length of tour is calculated and directly used for evaluating the fitness of each individual. We leave out the transformation procedure between the objective function and fitness function so as to deduce the computation amount.

The genetic operators of GGA make the most of the heuristic information to achieve local optima. The evolution of whole population fulfils the distributed and parallelized search. So the GGA search is a perfect combination of local and global search for optimal solution keeping from the premature convergence.

By comparing the qualities of parents to their offspring's, the individuals of GGA realize the population evolution. Only better offspring will replace its parent place. We abandon the traditional selection operator in our GGA so that the population's diversity is kept very well all along. Each individual runs in its own evolution direction, respectively. Different individuals search different domains. The greedy genetic algorithm takes on the parallelization nature due to its parallel searches.

We haven't employed special fitness function for TSP problems. The length of tour is calculated and directly used for evaluating the fitness of each individual. We leave out the transformation procedure between the objective function and fitness function so as to deduce the computation amount.

The genetic operators of GGA make the most of the heuristic information to achieve local optima. The evolution of whole population fulfills the distributed and parallelized search. So the GGA search is a perfect combination of local and global search for optimal solution keeping from the premature convergence.

5. Experimental results

We used standard TSP benchmarks (G. Reinelt, 1996) whose optimal solutions (or the current best solutions) are compiled, too. For all the problems, we use the same double-directional greedy crossover and greedy mutation possibilities of 0.8 and 0.02, respectively, but use different population sizes, immigrant possibilities and various number of generations for different problems, as Table 1 shows. Because the template based crossover operation is of the random operation, low possibility of template based crossover is adopted.

We run the GGA 10 times with 10 different random seeds contrast with GA so as to compare the average performance between GGA and GA. For comparison, we also experiment on the different effects between the greedy crossover operator (G. Andal Jayalakshmi, 2001) known and our double-directional greedy crossover operator. We list real number solutions, not the integral ones. From figure 2 to figure 7, we illustrate the best tour routes provided and the best solution that we calculate out with GGA for problem eil51, eil76, eil101, respectively. For problem eil51, we get a new better solution, shown in figure 3, than the provided one (G. Reinelt, 1996). We try to use a higher immigration possibility and less population size for problem eil101 in order to decrease the computation amount, where we get a solution shown in figure 6.

As illustrated in figure 8, for problem eil76, the average tour length of initial population generated from gene bank is 1012 in GGA. However, the average tour length of initial population generated randomly is 2561 in GA. From figure 8, the curve of average tour length declines straightly with the increase of evolution generation, especially in the start phase. But we notice the occurrence that the curve of GGA fluctuates slightly after the midterm phase of evolution. That is because, after evolving half of the whole generation number, the population is mixed with immigrants that lead the average population fitness to decrease. With sacrificing the high fitness a little, the population retrieves its diversities to some extent. However, the curve of GGA is still in the decline tendency generally. The immigration operation manifests its effect of inhibiting from premature convergence.

Problem instance	Control parameter of GGA			Solutions of GGA and solutions provided			
	Population size	Immigrant possibilities	Number of generations	Average tour length	Best tour length	Best tour length (G. Reinelt, 1996)	Quality of tour
eil51	150	0.15	2000	433.05	428.98	429.98	0.9977
eil76	200	0.15	2000	562.93	553.70	545.39	1.0152
eil101	105	0.2	5000	689.67	665.50	642.30	1.0353

Table 1. Empirical results

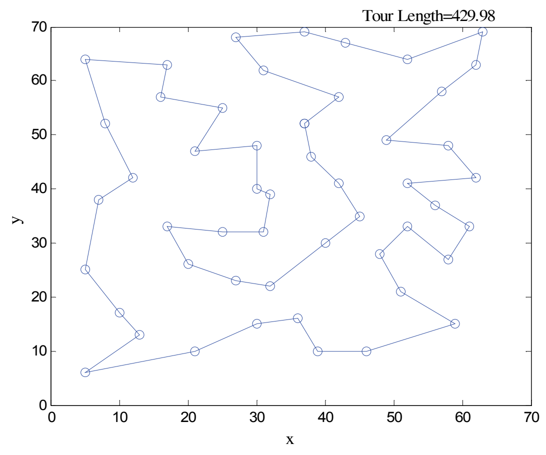

Fig. 2. Best solution of problem eil51 (G. Reinelt, 1996). Tour Length=429.98

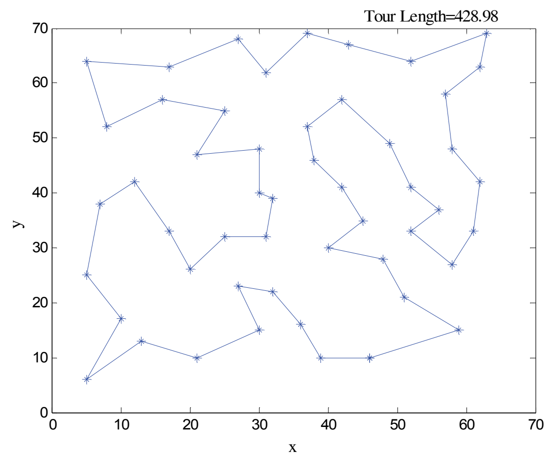

Fig. 3. Our best solution of problem eil51. Tour Length=428.98

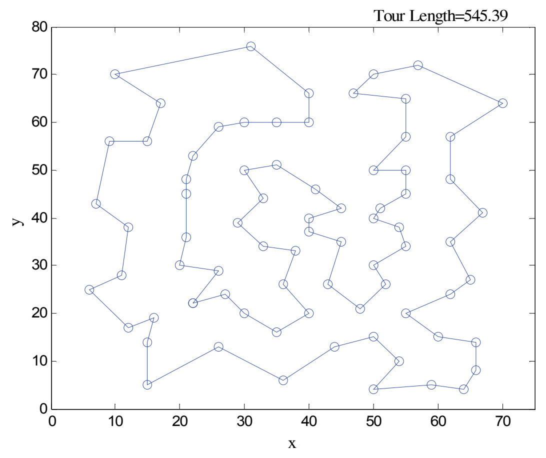

Fig. 4. Best solution of problem eil76 (G. Reinelt, 1996). Tour Length=545.39

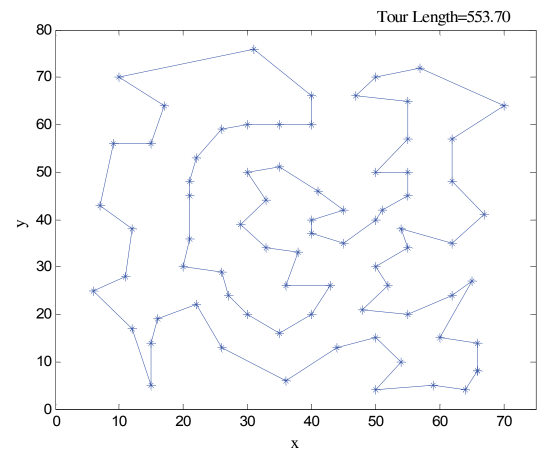

Fig. 5. Our best solution of problem eil76. Tour Length=553.70

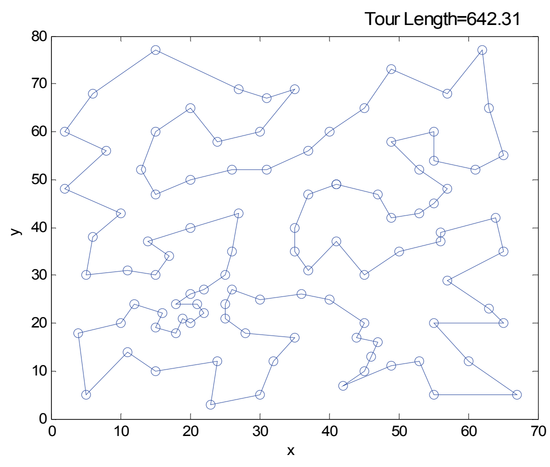

Fig. 6. Best solution of problem eil101 (G. Reinelt, 1996). Tour Length=642.31

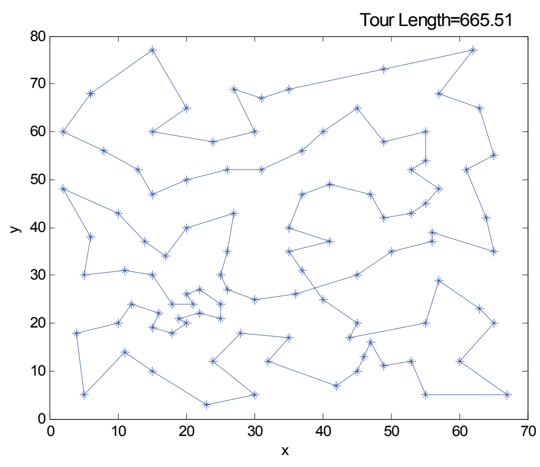

Fig. 7. Our best solution of problem eil101. Tour Length=665.51

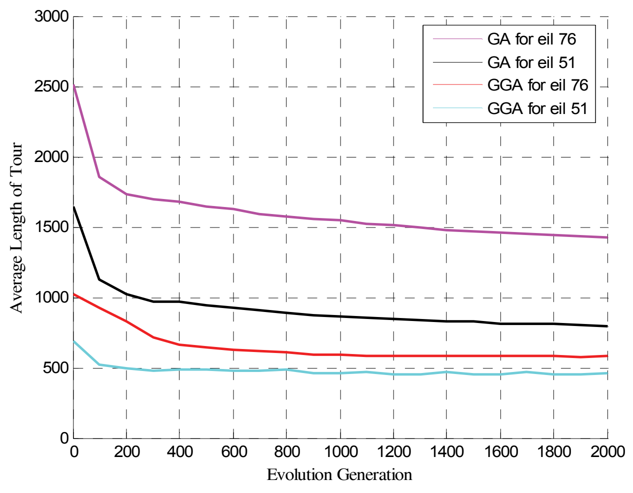

Fig. 8. Performance comparison of different algorithms

6. Conclusion

GA is unable to guarantee to achieve the optimal solution of problems. Compared to the GA, the greedy genetic algorithm with improved genetic operations has been presented for the global optimization of TSPs. The GGA is a parallel-searching algorithm based on TSP-oriented methodologies. Powerful heuristics developed in the corresponding field of TSPs can significantly increase the performance of the genetic algorithm. It is vital for GGA application to engineering practice that GGA works very efficiently in the start phase. A suit of benchmark test has been used to illustrate the merits of the modified genetic operations in GGA. Both the solution quality and stability are improved. GGA demonstrates its promising performance.

7. References

Andrzej Jaszkiewicz, (2002) Genetic local search for multi-objective combinatorial optimization, *European Journal of Operational Research*, Vol. 137, No. 1, pp. 50-71.

Bryan A. Norman & James C. Bean, (1999) A genetic algorithm methodology for complex scheduling problems, *Naval Research Logistics*, Vol. 46, No. 2, pp. 199-211.

Chatterjee S., Carrera C. & Lynch L., (1996) "Genetic algorithms and traveling salesman problems," *European Journal of Operational Research* vol. 93, No. 3, pp. 490-510.

Forbes J. Burkowski, (2004) "Proximity and priority: applying a gene expression algorithm to the Traveling Salesperson Problem," *Parallel Computing*, Vol. 30, No. 5-6, pp. 803-816.

G. Andal Jayalakshmi & S. Sathiamoorthy. (2001) "A Hybrid Genetic Algorithm: A New Approach to Solve Traveling Salesman Problem," *International Journal of Computational Engineering Science* Vol. 2, No. 2, pp. 339-355.

G. Reinelt, (1996) TSPLIB, University of Heidelberg, http://www. iwr. uni-heidelberg.de/iwr/comopt/soft/ TSPLIB95/TSPLIB.html.

Goldberg, D.E. & Lingle, R.J. (1985) Alleles, loci, and the traveling salesman problem, *Proceedings of the International Conference on Genetic Algorithms*, London, pp. 154-159.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

J. Grefenstette, R. Gopal, R. Rosmaita, & D. Gucht, (1985) Genetic algorithms for the traveling salesman problem, *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Eribaum Associates, Mahwah, NJ.

Paul K. Bergey & Cliff Ragsdale, (2005) "Modified differential evolution: a greedy random strategy for genetic recombination," *Omega*, Vol. 33, No. 3, pp. 255-265.

Whitley D., Starkweather, T. & Fuquay, D., (1989) "Scheduling problems and traveling salesmen: the genetic edge recombination operator," *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA, pp. 133-140.