**Scholars Journal of Engineering and Technology**

OPEN ACCESS

Abbreviated Key Title: Sch J Eng Tech
ISSN 2347-9523 (Print) | ISSN 2321-435X (Online)
Journal homepage: https://saspublishers.com

# Parallel Genetic Algorithm Based on Construction of Gene Pool in the Ordinary Network for TSP

Xiaoqin Fan[1*]

[1]General Education Department, Guangzhou Panyu Polytechnic, Guangzhou 511583, China

**DOI:** 10.36347/sjet.2022.v10i05.002 | **Received:** 11.04.2022 | **Accepted:** 16.05.2022 | **Published:** 20.05.2022

***Corresponding author:** Xiaoqin Fan
General Education Department, Guangzhou Panyu Polytechnic, Guangzhou 511583, China

**Abstract** | **Review Article**

Though using parallel evolutionary algorithm to solve large-scale TSP problems is efficient, the parallel computer costs too much and the algorithm is not easy to expand. To address this issue, I propose a parallel genetic algorithm based on a gene pool under the existing network. To replace the group-genes in the evolutionary algorithm with the genes from the gene pool, the algorithm conducts greedy algorithm. The host process conducts greedy algorithm and improved evolutionary algorithm of Inver-over operator while the child process performs the improved hybrid genetic algorithms. Simulation results demonstrate that this algorithm achieves a better solution.

**Keywords:** TSP; Reverse; Greedy gene pool; Parallel algorithms.



## INTRODUCTION

TSP (Travelling Salesman Problem), as an ideal problem, its research findings are not to be applied directly, but widely translated into many combinatorial optimized problems, such as cargo distribution of large chain stores, PCB drilling, genetic detection and etc.. They can be all abstracted to TSP problem. Therefore, the study of TSP problems and their solutions are of great significance both theoretically and practically.

The perfect method to solve the TSP problem is global search method. Due to the limitation of computer operating ability, when n is relatively big, with global search method, it seems impossible to find out the accurate optimal solution but only approximate solution. So far, there is no effective algorithm to solve this kind of problem, thus any simplified method to solve TSP will attract much attention and evaluation. Many scholars have great interest in TSP and many methods solving TSP emerg, including evolutionary algorithm [1,2].

Genetic Algorithm for TSP is of higher efficiency [3, 4], but because of the increasing "n", the numbers of the cities, using genetic algorithm to solve TSP resulting worse solution and decreased convergence rate. Here I propose parallel genetic algorithm to address this issue. However, huge investment in large parallel computer system is the barrier for expansion. When the number of cities in a TSP is relatively big, a certain number of parallel computers are needed for the tasks.

So this paper focuses on the construction of local gene pool and greedy gene pool under the common computer network, by using greedy algorithm and the improved Inver-over operator [5] in the child process, implementing dynamic optimization to the gene pool. In the main process, hybrid genetic algorithm based on the gene pool is applied to dynamically update the elite groups, replacing genes from the gene-group with genes from gene-library to improve the speed of evolutionary algorithms and the quality of the result.

This paper is divided into five sctions. The first section introduces the research background, the main contents, the purpose and the significance of the research.

The Second section is background knowledge. TSP, genetic algorithm, the present situations and the existing problems of using genetic algorithm to solve TSP are briefly introduced.

The third section deals with the method to construct gene pool. It mainly introduces the method to

build partial gene pool and greedy gene pool, and the algorithm thought of greedy gene pool.

The fourth section describes improvement of Inver - over operator. The fifth section presents the thought of greedy gene pool and the method of implementing parallel algorithm.

Based on normal computer simulation experiment under the general network, the sixth section selects some samples from TSP case library (TSPLIB) which is internationlly agreed, to verify the validity of the algorithm described.

The last section comes to a conclusion, and points out the deficiency of this article and the direction of further research.

**Introduction of tsp and genetic algorithms**

TSP long-standing problem is a typical combinatorial optimization problem and has proven to be NP-complete problem. The study has attracted many scholars and there have been a large number of algorithms for solving TSP.

Among them is evolutionary algorithm [1, 2]. Known as the traveling salesman problem or Wayfaring Salesman Problem, TSP problem may be summarized as follows: There are n cities. Starting from a given city, how does a traveling salesman find out the shortest way back to the starting city after having visited all the n cities? Its mathematical model is as follows: Given a graph $G = (V, E)$, the edge $e \in E$, a non-negative weights $W(e)$, find $G's$ Hamiltonian cycle $C$, making $C$ the total weight $W(C)$ the smallest, where $W(C) = \sum_{e \in E \subset E} W(e)$ [3,4].

It is well-known combinatorial optimization problems of mathematics field.

TSP was first mentioned in 1800. Between 1920s and 1950s of the 20th century, people began to realize that TSP is an NP problem [5, 6]; in 1954, optimal solution to TSP of 42 cities is obtained. Since 1954, the scale of optimal solutions to TSP is larger and larger. Instances with up to 13509 towns were managed to be exactly solved in the United States in 1998. An optimal tour through a 15,112-town instance in Germany is computed in 2001. Nevertheless, the cost of the project is huge. According to the report, to solve the TSP problem between 15112 towns in the United States, 110 computers with 500 MHZ compaq Ev6Alpha processor from Rice University and Princeton University were put into the process. These 110 connected computers spent 22.6 years in total. In May 2004, the Swedish obtained the optimal solution of 24978 towns.

TSP problem became increasingly popular in scientific circles in Europe and the USA. Notable contributions were made by George Dantzig, Delbert Ray Fulkerson and Selmer M. Johnson at the RAND Corporation in Santa Monica, who expressed the problem as an integer linear program and developed the cutting plane method for its solution. With these new methods they solved an instance with 49 cities to optimality by constructing a tour and proving that no other tour could be shorter. In the following decades, the problem was studied by many researchers from mathematics, computer science, chemistry, physics, and other sciences [6].

Richard M. Karp showed in 1972 that the Hamiltonian cycle problem was NP-complete, which implies the NP-hardness of TSP. This supplied a mathematical explanation for the apparent computational difficulty of finding optimal tours.

Great progress was made in the late 1970s and 1980, when Grötschel, Padberg, Rinaldi and others managed to exactly solve instances with up to 2392 cities, using cutting planes and branch-and-bound.

In the 1990s, Applegate, Bixby, Chvátal, and Cook developed the program Concorde that has been used in many recent record solutions. Gerhard Reinelt published the TSPLIB in 1991, a collection of benchmark instances of varying difficulty, which has been used by many research groups for comparing results. In 2006, Cook and others computed an optimal tour through an 85,900-city instance given by a microchip layout problem, currently the largest solved TSPLIB instance. For many other instances with millions of cities, solutions can be found that are guaranteed to be within 2-3% of an optimal.

The total number of possible paths of TSP and the number of cities are increased by factorial number; therefore, it is difficult to find out the optimal solution. As for this problems, no matter the traditional dynamic programming, branch and bound method, greedy method or other recent methods, like intelligent optimization algorithms (tabu search, simulated annealing, genetic algorithm and artificial neural networks, ant algorithm) and their hybrid algorithm are of lower efficiency, and higher cost.

Genetic algorithm is a stochastic simulation of biological evolutionary mechanisms global search and optimization methods [6]. It automatically obtains and optimizes the search space, and adaptively control the search process in order to achieve the optimal solution. General procedures for genetic algorithm optimization problems are as follows:

**1) Initialization**

The population size depends on the nature of the problem, but typically contains several hundreds or

thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found [7].

**2) Selection**

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity.

**3) Genetic operators**

The next step is to generate a second generation population of solutions from those selected through a combination of genetic operators: crossover (also called recombination), and mutation.

**4) Termination**

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

1. A solution is found that satisfies minimum criteria.
2. Fixed number of generations is reached.

As genetic algorithm is not affected by the limitation of search space, requirements such as continuity, conductivity and unimodality are not necessary. Its robustness and implicit parallelism make it is widely used in solving the complex problems that are difficult to solve with the traditional methods [7], such as combinatorial optimization, pattern recognition, computer network optimization, etc. People have been using genetic algorithm to solve large-scale traveling salesman problem [6].

The most efficient way to solve TSP is to use genetic algorithm and other similar algorithm. Compared to the traditional algorithms, genetic algorithm doesn't take the process into consideration but directly focus on the shortest distance so as to obtain the solution as soon as possible. But its large search space takes long time, and it is sensitive to the initial value, so genetic algorithm effects slowly on the large-scale TSP problem [8]. Local optimization algorithm is very efficient when applied in local optimal TSP. It solves instances with up to hundreds of cities in a very short period of time, but it is easy to be trapped in local optimal solution.

**CONSTRUCTION OF GREEDY GENE POOL**

Set that $P_k = V_1^k V_2^k \cdots V_n^k$ is a feasible path for point $V_1, V_2, \cdots, V_n$, the total length of the loop $P_k$ is $f(P_k) = \sum_{i=1}^{n-1} d(V_i^k, V_{i+1}^k) + d(V_n^k, V_1^k)$, where, $V_j^k$ is the j[th] point, $d(V_i^k, V_{i+1}^k)$ is the Euclidean distance between point $V_i^k$ and point $V_{i+1}^k$. By using this algorithm, the total length $f(P_k)$ of the loop $P_k$ can be used to evaluate the individual [9].

Set $G = (V, E), V = \{1,2,\cdots,n\}, E = \{(i,j) \mid i, j \in V\}$, the coordinates of point i and point j are $(x_i, y_i)$ and $(x_j, y_j)$ respectively, then the Euclidean distance between i and j is $d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Set that $D = \{d(i,j)\}$, then D is the square of $n \times n$.

Set $A = (a_{ij})_{n\times(n-1)} = \begin{bmatrix} 2 & 3 & 4 & \cdots & n \\ 1 & 3 & 4 & \cdots & n \\ 1 & 2 & 4 & \cdots & n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2 & 3 & \cdots & n-1 \end{bmatrix}_{n\times(n-1)}$. For each point $i$, according to the size of $d(i,j)$, sort the $i$th line of the corresponding elements in $A$ in accordance with the order from small to large, and insert $[1,2,3,\cdots,n]^T$ into the first column in $A$, expand $A$ to a phalanx $n \times n$, calling the phalanx local gene pool $A_1$. After the generation of local gene pool $A_1$, the first m columns were selected from $A_1$ to compose n-6 new matrixes

$B_m, m = 6,7,\cdots,n-1,$ this matrixes $B_m$ vary in the number of points, for example, there are m total different points in $B_m$, which respectively reflects the local situation in the name of a certain point as the center, while the $i$ th line in $B_m$ constitutes point i's neighbor set points. In the algorithm, each $B_m$ can be optimized by using greedy algorithm, and according to the computer distribution, it can be separately placed on different computers to run independently, and thus get a new $B_m$, that is, the greedy gene pool. This n-6 matrix point number which varies, for example, a total of 6 $B_6$ in different points 7 points different from $B_7$, They are represented in the center of which is a point of local conditions, such as matrix Bm constitute the first line of the point i i neighbor point set. Algorithm runs, for each matrix greedy algorithm implementation, and according to the actual situation of the computer running the distribution will be different on different computers Bm run independently greedy algorithm, matrix $C_m$, where $C_m$ is greedy gene pool. TSP problem with the greedy algorithm, the time required does not exceed $O(m^2)$. In general, solution obtained with greedy algorithm is not the optimal for TSP problem. It is necessary to optimize the greedy gene pool in the child process, and to get the best genes from the main process for dynamic update of the local gene pool.

Since in the evolvement process, the genes involved in genetic operators promoter is mainly from the individuals, thus the quality of the evolved individual determines the efficiency of the algorithm. If individual's fitness values are poor, the overall performance of the algorithm will be affected, especially for TSP. Although the greedy algorithm does not guarantee optimal solution, we can use it to generate relatively good initial population, and thus greatly improve parallel TSP algorithm performance of the sub-process evolution, solving the speed, quality solution [10].

Paper [11] presents the evolution algorithm for TSP by constructing the gene pool which improves both the accuracy and the speed. But its point-centered gene pool constructs gene chip from the near points, without considering the relationship among the points which are among the gene fragments. So the algorithm is running fast early, but in the latter part of the evolution of computing, the gene pool almost had no effect on group.

Because the front part of the gene constructed in the algorithm is of fine locality, only the first m individual genes are extracted to construct local gene library, reducing the length of the gene, thereby improving the efficiency of the algorithm.

**IMPROVED INVER-OVER OPERATOR**

Guo's algorithm [12-14] is one of the fastest algorithms for solving TSP which proposed Inver-over operator. Inver-over operator has characteristics of both crossover and mutation which takes full advantage of the information community, and it is much higher in speed and quality than other simple crossover. But the experiment also shows that when the number of cities becomes relatively larger, its global optimization capability dramatically declines. As a result, many scholars have studied Inver-over operator and try to improve it [5, 10]. This paper improves Inver-over operator by labeling the all points near every vertex as its neighboring point set, initializing each point set, choosing the next search space instructively [11]. The main steps are as follows:

```
Initialize the group P using the neighbor set of points
  While (stop condition not satisfied)
  {
    For (each individual Si of groups)
      {
        S'= Si;
        Select a vertex c from S randomly;
        While (true)
          {
            Generate a random number p (0 ≤ p ≤ 1);
            if (p<pc)     /*pc is a constant*/
              {
                Select vertex c' from the remaining selected vertices from S';
              }
            else
              {
                Randomly choose an individual from P;
                Mark the selected-individual c's next vertex as c;
              }
            if (the vertices c and c' in S' are adjacent)
              {
```

```
                    break;
                }
            else
            {
                Flipped vertex c to the next vertex to the vertices between c', c = c';
            }
        }  /* end while*/
    if (the fitness value of S'≤ Si)
        {
            Si = S';
        }
    }   / * end for*/
}  / * end while*/
```

**Parallel algorithm basing on the greedy gene pool**

Based on the greedy evolution of the gene pool, parallel algorithm sets a computer as the mainframe of the entire system, which runs the main process and as the center of data exchange in the evaluation process. The main process optimizes the elite by using the hybrid genetic algorithm based on gene pool and Guo Tao algorithm [11]. The basic steps are as follows:

(1) to generate an initial local gene pool and $n-6$ greedy gene pool, and to deposit these genes into the database for sharing;(2) to write the initial control information into an asynchronous control information database, to prohibit the work machines to read $A_1$ and $B_m$ when the algorithm starts; (3) to generate $A_1$ and $B_m$, and deposit them into their corresponding database;(4) to divide the working machines based on the n-size of the TSP problem so that it can deal separately with one or more lines in $A_1$ and $B_m$;(5) to write control information into an asynchronous control information database, allowing working machine to read $A_1$ and $B_m$ ; and store the best individuals in the elite group into the database for the sharing of the working machines;(6) to read from the database the evolution elite groups and generate the optimal solution, and then deposit it into its database s after setting every other algebra.

Other computers in the network as a working machine run sub-process and read the TSP from the mainframe and obtain the corresponding gene pool. Sub process only solves point-centered domain.

**The main steps are**

(1) to copy all the data from the mainframe to the working machine and extract genes from the corresponding local gene pool according to the task assigned or set (2) to generate a number m randomly, implement the greedy algorithm on $B_m$, so that it becomes greedy gene pool $C_m$;(3) to implement improved Inver-over algorithms on $C_m$;(4) to store $C_m$ in the mainframe to obtain the optimal solution;(5) to obtain the gene chip with the length of m and $C_m$ for hybridization [15]; store optimal gene in the mainframe, turn to (3).

**SIMULATION**

Multiple TSP problems are selected for testing from the paper[16] and international general TSP instance library TSPLIB[17]. Algorithm uses the C # programming language in Microsoft Visual Studio 2019 Preview platform, which is configured to host CPU Intel Core i7-1165G7@4.70GHz, memory is 16 GB. The operating system is Windows10. The database software is Microsoft SQL Server 2019. The working machine connects mainframe database through ADO.NET, carrying out the process on 50 computers in LAN. Table 1 lists the optimal value. The optimal paths for the first 6 problems are as in Figure 1-6, while the optimal paths for problem pr2392 are too many to be listed.

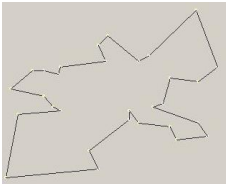

**Fig-1: Oliver30 optimal path**

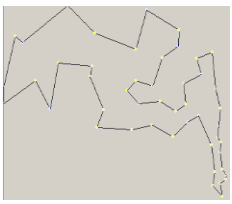

**Fig-2: Att48 optimal path**

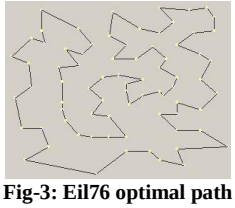

**Fig-3: Eil76 optimal path**

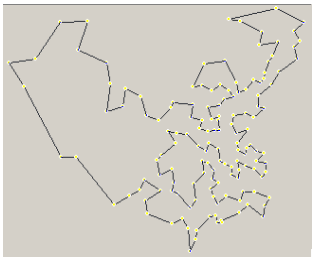

**Fig-4: Chn144 optimal path**

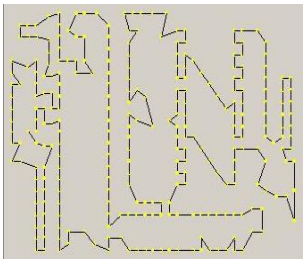

**Fig-5: A280 optimal path**

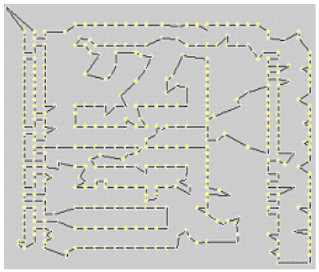

**Fig-6: TSP pcb442 optimal path**

Judging from the simulation, the proposed algorithm is particularly suitable for handling large-scale TSP. As in Table 1, the algorithm for smaller-scale TSP is of less advantage. Many papers propose optimal solution to small-scale TSP problem, but the algorithm proposed now is appropriate for large-scale TSP problem. Nevertheless, the results of large-scale TSP is rare, they are not listed in Table 1.

**Table-1: TSP algorithm running results**

| TSP | optimum | Size of the group | Average time |
|---|---|---|---|
| oliver30 | 423.740563133203 | 50 | 0.2617 |
| att48 | 33523.7085074356 | 50 | 0.372s |
| eil76 | 544.369052670828 | 50 | 1.374s |
| chn144 | 30353.4474810516 | 100 | 2.836s |
| a280 | 2587.80879066408 | 100 | 83s |
| pcb422 | 50935.5635917108 | 100 | 128s |
| pr2392 | 386606.457689425 | 100 | 1993.064s |

## CONCLUSION

This paper proposes an algorithm to solve the problems using computers in general network instead of parallel computer, which is economic convenient and fast.

Simulation shows that the proposed algorithm is feasible. When the computer configuration is low, it is an effective way to solve tough problems. When the scale of TSP is too large, the algorithm converges relatively slow and the algorithm needs further improvement.

## REFERENCES


1. Baraglia, R., Hidalgo, J. I., & Perego, R. (2001). A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on evolutionary computation*, *5*(6), 613-622.
2. Wen, Yi., PanDa-zhi. (2016). Improved Genetic Algoithm for Traveling Salesman Problem [J]. Computer Science, (S1): 90-92.
3. Sun Wen-bin,Wang, J. (2016). An Algorithm for TSP Problem Based on Genetic Algorithm and Multi-optimization Operration [J]. *Geography and Geo-Information Science*, *32*(4); 1-4.
4. Hui, Y., Li-shanf, K., & Yu-Ping, C. (2003). A gene-pool based genetic algorithm for TSP. *Wuhan University Journal of Natural Sciences*, *8*(1), 217-223.
5. Zhai, F., Xie Xian-hua. (2020). Study on optimal robot task scheduling based on genetic algorithms [J]. Mathematics in practice and thory, *50*(15); 143-154.
6. Clarke. (1964). G, Wright J W. Scheduling of vehicles from a central depot to a number of delivery points. *Opera 鄢 tions Research*, *12*(4), 568.
7. Dan, L. M. K. J. L., Quan, L. S., Ming-Qiang, L., & Song, K. J. (2004). The basic theory and application of genetic algorithms. *The publisher of science. Beijin*.
8. Simin, Y., Fengjun, W. (2020). Research on Solving TSP with Genetic Algorithm or Branch and Bound Method[J]. Computer Science and Application, *10*(9); 1609-1617.
9. Luo, Z., Feng, S., Liu, X. (2020). Method of area coverage path planning of multiunmanned cleaning vehicles based on step by step genetic algorithm [J]. Journal of Electronic Measurement and Instrumentation, *32*(8); 43-50.
10. TAN, N. B., WANG, J., & MOU, L. M. (2009). Improved genetic algorithm for solving TSP problem. *Journal of Jiamusi University (Natural Science Edition)*.
11. Chen, S., LIN Piyuan, HUANG, P. (2020). Pointer Network Improved Genetic Algorithm for Solving

Traveling Salesmen Problem [J]. *Computer Engineering and Applications*, *56*(19):231-236.

12. Hassanat, A. B., Prasath, V. B., Abbadi, M. A., Abu-Qdari, S. A., & Faris, H. (2018). An improved genetic algorithm with a new initialization mechanism based on regression techniques. *Information*, *9*(7), 167.
13. Fu, C., Zhang, L., Wang, X., & Qiao, L. (2018, May). Solving TSP problem with improved genetic algorithm. In *AIP Conference Proceedings* (Vol. 1967, No. 1, p. 040057). AIP Publishing LLC.
14. Agrawal, M., & Jain, V. (2020, July). Applying Improved Genetic Algorithm to Solve Travelling Salesman Problem. In *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 1194-1197). IEEE.
15. Akter, S., Nahar, N., ShahadatHossain, M., & Andersson, K. (2019, February). A new crossover technique to improve genetic algorithm and its application to TSP. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)* (pp. 1-6). IEEE.
16. http://www.iwr.uniheidelberg.de/groups/co mopt /software/TSPLIB95/